New Hybrid Genetic Algorithms for Parameter Search

Tarek A. El-Mihoub

A thesis submitted in partial fulfilment of the requirements of Nottingham Trent University for the degree of Doctor of Philosophy

April 2006

Contents

List of Figures

Abstract

A genetic algorithm is a computational optimisation algorithm that is inspired by the principles of natural selection and genetic dynamics. Hybridising other optimisation techniques within the genetic algorithm framework can enhance the search performance. The chances of improving the performance of a hybrid depend on the details of its design.

This thesis aims to employ learning to utilise the genetic information that is readily available to maximise the effectiveness and the efficiency of a hybrid. Learning can be incorporated in different ways to achieve this goal. It can utilise solution-specific knowledge to improve its contribution in the search process. It can also utilise the direct and indirect influence of the design choices on the exploration-exploitation trade-off to adapt the search control parameters to a problem without external control.

This thesis examines the different hybrid design issues and their effect on the hybrid's performance. It contributes towards discovering the nature of the relations between the hybrid design choices and the hybrid's performance. The research demonstrates that the two main drawbacks of the basic learning models, which are the hindering effect and the diversity limitation, can be alleviated through adjusting the duration and the probability of local search. Some of the search method's features that enable it to be incorporated within a genetic search to accelerate finding high quality solutions are defined. They empower the search method to learn the nature of the search space through using the genetic information. An effective model with such features is also presented.

This thesis also investigates different ways of achieving a balance between exploration and exploitation. Utilising learning to make use of the direct influence of the details of the local method on this balance can help to find an optimal setup for this method. The investigation demonstrates the effectiveness of applying co-evolution for such utilisation. It also analyses the effect of using the fitness as productivity metric on the search's behaviour. It also illustrates the impact of the hindering effect on this mechanism and the possible ways to combat it. The research shows the effectiveness and the efficiency of employing evolution to use the indirect influence of the learning model on the utilisation of the search time for online learning of the effectiveness of the different learning strategies. It also explains the slow convergence of the evolutionary self-adaptive algorithms that adjust more than one control parameter based on the probabilities of introducing epistasis.

A novel form of hybridisation between ant-based and genetic-local hybrid algorithms is proposed. This work has demonstrated the ability of ant-based algorithms for reinforcement

learning. They have been shown to be capable of finding, without any form of human intervention, an optimal setup for a genetic-local hybrid algorithm that can effectively and efficiently solve a given problem.

List of publications

El-Mihoub, T., Hopgood, A.A., Nolle, L., Battersby, A.: Performance of Hybrid Genetic Algorithms Incorporating Local Search, Proceedings of the 18th European Simulation Multiconference ESM 2004, Magdeburg, Germany, 13-14 June 2004, pp 154-160, ISBN 3-936150-35-4.

El-Mihoub, T., Hopgood, A.A., Nolle, L., Battersby, A.: Hybrid Genetic Algorithms: a review to appear in the Special Issue "Hybrid Intelligent Systems using Neural Networks, Fuzzy Logic, and Genetic Algorithms" of Engineering Letters journal.

El-Mihoub, T., Hopgood, A.A., Nolle, L., Battersby, A.: Self-adaptive Baldwinian Search in Hybrid Genetic Algorithms, to appear in Proceedings of the 6th Fuzzy Days International Conference on Computational Intelligence, 2006

El-Mihoub, T., Nolle, L., Schaefer, G., Nakashima, T., and Hopgood, A.: A Self-Adaptive Hybrid Genetic Algorithm for Color Clustering, to appear in Proceedings of the 2006 IEEE International Conference on Systems, Man, and Cybernetics.

Acknowledgments

First, I would like to thank and praise Allah. Without his help this thesis would have not been possible.

I would like to thank my academic supervisors Adrian Hopgood, Lars Nolle, and Allan Battersby for their boundless and instructive help, criticism and patience. I am grateful to them for always having an open door to discuss, listen and help with my research.

I am also grateful to my colleagues and friends who have provide help, advice and companionship during my study.

I should also acknowledge the encouragement and the support of my brothers and sisters.

Gratitude is also due to my sons Mohamed Algasim and Mayyar who make my life and my study enjoyable.

Finally, my wife deserves special mention for her constant support and understanding.

To my parents

and my sister Amal

Chapter 1 Introduction

This dissertation investigates the degree to which learning can extend the effectiveness and efficiency of hybrid genetic search algorithms. It describes a set of experiments that provide detailed insight into the effect of design choices for the hybrid on the search's behaviour and how they relate to each other. It proposes a probabilistic local search method as a solution to the problem of interference between genetic schema processing and the use of local knowledge. It also describes and evaluates hybrid genetic algorithm solutions that can adapt to a wide range of optimisation problems.

A genetic algorithm is a population-based search and optimisation technique that mimics the process of natural evolution. The two main concepts of natural evolution, which are natural selection and genetic dynamics, inspired the development of this method. The basic principles of this technique were first laid down by Holland (Holland 1975) and are well described, for example, in (De Jong 1975) (Goldberg 1989a).

1.1 Hybrid genetic algorithms

A genetic algorithm evolves a set of candidate solutions through examining their fitness to select the most promising ones, and manipulating those using genetic operators in order to find optimal solutions. Genetic algorithms are able to find acceptable solutions to a wide verity of problems (De Jong 2005). However, they are likely to be outperformed by problem-specific techniques in both speed and accuracy of the final result (Areibi et al. 2001). Genetic algorithms are not well suited to fine-tune the solutions quality (Reeves 1994), but on the other hand they are effective at exploring the search space (Ibaraki 1997).

The incorporation of a local search method or a problem-specific technique into a genetic algorithm is essential if an effective and efficient algorithm is desired (Areibi et al. 2001). Combining global and local search is a strategy used by many successful global optimisation approaches (Talbi 2002), and hybrid genetic algorithms have in fact been recognised as a powerful algorithmic paradigm for evolutionary computing. In particular, the relative advantage of hybrid genetic algorithms over genetic algorithms is quite consistent on complex search spaces (Lobo and Goldberg 1997).

The idea of improving the performance of a genetic algorithm by combining it with local search methods for solving complicated optimisation problems has been investigated extensively during the past decade, and different forms of hybridisation have been proposed (Preux and Talbi 1999). A genetic algorithm can be used to capture a global view of the

entire search space of the optimisation problem while the local search method can be utilised to incorporate some of the domain knowledge by performing local exploitation (Lobo and Goldberg 1997). Because of the complementary properties of genetic algorithms and local search methods, the hybrid approach often outperforms either method when they are used individually (Goldberg and Vosser 1999).

The two common approaches for combining a genetic algorithm with a local search method are Lamarckian evolution (the active form) and the Baldwin effect (the passive form) (Anderson et al. 1997). Both forms use the metaphor that an individual learns during its lifetime. In Lamarckian evolution, direct learning passes the improved characteristics of each individual from one generation to another. This means both the change in the genotypic information and the improved phenotypic fitness are passed to the next generation as genotypic information at the end of the learning process. On the other hand, in the Baldwin effect, only the improved phenotypic fitness is passed at the end of learning process (Hinton and Nowlan 1987). Lamarckian is discredited in nature, but useful as an algorithm. Although the Baldwin search strategy is slower than the Lamarckian strategy, it is better at avoiding premature convergence by maintaining diversity (Whitley et al. 1994).

Search algorithms need to balance exploration across the search space with exploitation of the optimum region. As local search in a hybrid genetic algorithm enhances exploitation, the balance needs to be redressed to achieve optimal performance. Achieving this balance is strongly dependent on the settings of a hybrid genetic algorithm's control parameters, which have significant impact on its performance (Hart 1994). Choosing suitable values for these parameters is one of the main difficulties when building a practical pure or hybrid genetic algorithm (Deb 1997).

In addition to parameter adjustment, there are several issues that need to be taken into consideration when designing a hybrid search algorithm. These issues include decision-making between global and local search to maximise effectiveness and efficiency based on their cost, reliability and ability to use knowledge of problem domain (Lobo and Goldberg 1997). Incorporating some knowledge of the problem domain and some knowledge of the strengths and weakness of the search methods can guide hybrid genetic algorithms to make these decisions effectively. The way of utilising local information, i.e. the learning approach, within a hybrid algorithm is also an important issue that faces the designers of hybrid genetic algorithms due its effect on its performance.

1.2 Research Perspective

The focus of this dissertation is on gaining an understanding of the hybrid genetic algorithm's design issues and their influence on the hybrid's performance. Hybrid design choices related to the incorporation of a local search method are analysed to uncover key features and relations that make a hybrid's search effective and efficient without the need of choosing the settings for the control parameters manually. The proposal, here, is to investigate these design issues by designing, developing and testing hybrid genetic algorithms that employ learning to direct their operations, and to adapt their control parameters to find high quality solutions to a wide range of optimisation problems efficiently. This investigation will focus on a decision-making problem and learning strategies in order to get the most out of the genetic algorithm, as a global search method, and the problem-specific technique, as a local search method. It will also examine possible methods of controlling the hybrid genetic algorithm's vast parameter space to make the algorithm's performance more effective without the need for any forms of human intervention. The question this research seeks to address is "to what extent can the incorporation of learning help a hybrid genetic algorithm search a solution space more efficiently?"

The existing theoretical models of genetic algorithms are limited in use and applicability. Therefore, the majority of theoretical work has been derived from experimentation. The approach taken in this thesis is also based on the careful design, collection and analysis of experimental results.

1.3 Dissertation Overview

This dissertation is divided into 9 chapters, beginning with this introduction. A literature review of global optimisation, which is concentrated on genetic algorithms and their hybrids, follows in chapter 2. Chapter 3 investigates the impact of the duration of local search on the hybrid's performance, whereas chapter 4 studies the effect of the probability of local search and learning strategies on the hybrid's performance. A probabilistic search method is proposed in chapter 5 as a solution to reduce interference between genetic schema processing and utilising local knowledge within a hybrid. Applying evolution to self-adapt the duration of local search as a way to optimise the utilisation of hybrid's time is explored in chapter 6. The use of an adaptive technique to decide on the learning strategy is investigated in chapter 7. Chapter 8 studies the use of ant colony optimisation to fully self-adapt genetic-local hybrids. Chapter 9 draws the thesis to an end by providing a discussion and recommendations.

In the following, the content of each of these chapters is outlined in greater detail.

Chapter 2 introduces single-point and population-based optimisation techniques. It discusses the difficulties they face when used to solve global optimisation problems. An in-depth examination of genetic algorithms and the different ways of hybridising them with other search and optimisation methods follows in order to shed some light on the effectiveness and efficiency of hybridising genetic algorithms. Two major genetic-local hybrid design's issues are then discussed. These issues include the different approaches for employing local search information and various mechanisms for achieving a balance between a global genetic algorithm and a local search method. This chapter emphasises on the importance of combining genetic algorithms with other techniques to build competent genetic algorithms that solve hard problems quickly, reliably and accurately without the need for any forms of human intervention.

The third chapter investigates the influence of the duration of local search on the performance of hybrid genetic algorithms. Its interactions with the learning strategy and their combined effect on the sampling ability of the global genetic algorithm are studied. It, then, analyses the effects of the duration of local search on its role in a hybrid to provide insight into the expected behaviour of a hybrid depending on the duration of its local search. The results shed some light on the combined effect of the duration of local search and the learning strategy on the hybrid's performance.

In the fourth chapter, the effects of the learning strategy and the probability of local search on the performance of two hybrids with different mechanisms for deciding between global and local search are explored. The way that both the learning strategy and the probability of local search interact with each other and their combined effect on the hybrid's performance is analysed. The effect of both these factors on the population size requirements, convergence speed, and solution quality is investigated. The results emphasise the importance of the relation between the probability of local search and the learning strategy on the hybrid's performance.

A simple probabilistic search method is developed in the fifth chapter as a secondary search method within a hybrid genetic algorithm. The developed method aims to make use of some of the available genetic information to reduce any conflict with the genetic algorithm schema processing in order to utilise the efficiency of the Lamarckian learning approach to produce an effective search. This chapter evaluates this method as a secondary method in a hybrid and as a stand-alone optimisation algorithm using three test functions with different

marginal fitness contribution of their genes. The results of the evaluation illustrate the ability of the developed method to reduce disrupting the genetic schema processing.

The advantages and disadvantages of applying evolution to self-adapt the control parameters associated with the utilisation of the local search within a hybrid genetic algorithm are explored in the sixth chapter. The effect of this form of adaptation on the hybrid's performance on different test functions is studied. The impact of the implicit use of the productivity metric as a measure to decide on local search control parameter values on the self-adaptive hybrid's performance is also analysed. The influence of the interactions between learning strategy and local search method on the self-adaptation behaviour and the possible ways to improve this form of adaptation are also studied. The performance of the developed algorithm is compared with another adaptive algorithm.

The aim of the seventh chapter is to investigate the use of an adaptive approach to decide on the learning mechanism. Assigning different learning strategies for the population's individuals over the course of the run via some intelligent means is investigated through applying evolution to self-adapt the learning mechanism within a hybrid genetic algorithm. This chapter examines the effect of this form of adaptation on the hybrid's performance in order to get some insight into its advantages and disadvantages. It also investigates the interactions between this form of adaptive learning and two different adaptive hybrid genetic algorithms.

In chapter 8, a novel form of hybridisation between an ant-based algorithm and a genetic-local hybrid algorithm is proposed. An ant colony optimisation algorithm is used to monitor the behaviour of a genetic-local hybrid algorithm in order to dynamically adjust the probabilities of using the genetic operators, the local search operator, its duration, and the learning strategies to adapt the hybrid's performance to a given problem. The effectiveness of using ant-based algorithm as a reinforcement learning approach is compared with the effectiveness genetic algorithms in self-adapting hybrid genetic algorithms.

This thesis is drawn to an end by chapter 9, where a summary of the research findings is given and the main contributions of the thesis are evaluated in some depth. Next, this thesis is evaluated, suggesting where some more experiments are needed and where some methods need further development. Finally, a section on further work describes key directions of interesting further study to the research in the thesis.

Chapter 2 Literature Review

Optimisation is concerned with the computation and characterisation of the minimum or maximum of mathematical functions (Hopgood 2001 pp.164). Optimisation problems are widespread in the mathematical modelling of real life systems for a very broad range of applications (Horst and Pardalos 1995). Such applications include economies of scale (Joborn et al. 2004), allocation and location problems (Houck et al. 1996) (Joines and Kay 2002), operation research (Tsang and Voudouris 1997), structural optimisation (Striz and Sobieszczanski-Sobieski 1996) (Kim and de Weck 2004), engineering design (Deb 1999), network and transportation problems (Dorigo et al. 1999), chip design (Areibi and Yang 2004), database problems (Bommel and der Weide 1992), nuclear (Bäck et al. 1996 cited in Bäck et al. 1997) and mechanical design (Giraud-Moreau and Lafon 2002) (Deb and Goel 2001), chemical engineering design (Lin and Miller 2004a) and control (Wang et al. 2003), and molecular biology (Morris et al. 1998) (Shmygelska and Hoos 2005). These applications include also a number of other combinatorial optimisation problems such as integer programming (Rudolph 1994) and related graph problems (Grefenstette et al. 1985) (Julstrom 1999).

Researchers in different fields have suggested a huge number of optimisation techniques. Unlimited refinements have made these techniques work on specific types of applications. All these procedures are based on some common ideas and are furthermore characterised by a few additional specific features. This chapter reviews some of these optimisation methods and concentrates on solving the following general global optimisation problem (f)

$$\max f(x) : x \in S \tag{2.1}$$

where $f(x)$ is a continuous function on S (S here refers to the search space or the set of all possible solutions), and $S \subseteq \Re^d$ is a compact body. Some of the methods that will be described require additional assumptions on the objective function $f(x)$ or the feasible region S. They will be noted whenever necessary. The optimal global solution value

$$f^* \equiv \max_{x \in S} f(x) \tag{2.2}$$

is assumed to exist and is attained, i.e. the set

$$S^* \equiv \{x \in S : f(x) = f^*\} \tag{2.3}$$

is not empty. It is also assumed one or more local optimum values can exist with

$$f_{local} \equiv \max_{x \in s} f(x) \leq f^* \tag{2.4}$$

where $s \subset S$.

This chapter starts with an overview of single point search methods, the difficulties they face when used to solve global optimisation problems, and the different techniques used to improve their efficiency and effectiveness. Then, ant algorithms and genetic algorithms, as population-based search methods, are reviewed. After that, the difficulties genetic algorithms face when globally optimising problems and a brief description of the techniques to overcome these difficulties are given. The hybridisation of single-point search methods and the genetic algorithms, as choice to overcome the obstacles of global optimisation, is discussed. Some issues that affect the hybrid performance and design are reviewed.

2.1 Single-point search and local search methods

The simplest form of search is single-point search. Its procedure can be summarised in the following four steps:

Step 1: Choose a potential solution from the search space and evaluate its merit or fitness. Define it as the current solution.

Step 2: Modify the current solution to generate a new solution and evaluate the new solution merit or fitness.

Step 3: If the new solution is better than the current solution, then exchange it with the current solution; otherwise discard the new solution.

Step 4: Repeat the two previous steps until no modification can improve the current solution.

The effectiveness of the single-point search depends on the modification applied to the current solution. If the modification operator returns a potential solution from the search space selected uniformly at random, the search becomes an essentially exhaustive search (Michalewicz and Fogel 2000 pp.58) with a probability of re-sampling the same solution more than once. However, if the modification operator uses neighbourhood information to generate a new potential solution, the algorithm will be biased by the current solution. The search for an optimum will be concentrated around the current solution and can be easily trapped in a local optimum. This form of search is known as local search method or neighbourhood search.

From the previous description of the local search algorithm, it is clear that any local search has the following three basic components:

Neighbourhood: represents the subset of potential solutions that are immediately reachable from a potential solution. If the size of this set is small then the neighbourhood can be searched very quickly, but the search algorithm can be easily trapped at a local optimum. In contrast, if the size of the neighbourhood is very large there is less chance of being trapped, but the efficiency may suffer (Michalewicz and Fogel 2000 pp.58).

Modification operator: represents the way of generating the next potential solution from the current potential solution and determines the size of the neighbourhood (Michalewicz and Fogel 2000 pp.58).

Accept function: represents the policy used for accepting moves suggested by the modification operator.

The simplest form of local search is the random walk method, where the modification operator chooses randomly a solution in the neighbourhood of the current potential solution. The new solution then becomes the current solution regardless of the difference between the merits of both solutions. The algorithm locates the optimum by keeping the overall best solution encountered during the random walk. This algorithm, as many other algorithms, does not have a well defined stopping criterion.

The hill-climbing search differs from the random walk method in the accept function, where the current solution is replaced by the new solution only if it is better than the current solution. Searching large neighbourhoods for an improving potential solution can be very time consuming. This problem faces optimising continuous functions where the number of members of the neighbourhood is infinite. Probabilistic sampling from the neighbourhood is usually used to solve this problem. Soils and Wet (1981 cited in Hart 1994) proposed a random local search for continuous functions. They used a normal distribution with zero mean to modify every dimension of the current solution and depending on the rate at which better solutions are found, the variance of the normal distribution is modified. The algorithm is halted after a fixed number of function evaluations or when the step size becomes smaller than a given threshold.

2.1.1 Improving efficiency of local search

More advanced forms of local search methods have been used to improve the efficiency of local search in finding the local optimum. These advanced forms make use of additional local information to accelerate the search. These methods include bracketing search methods (Press et al. 1993 pp. 397), gradient methods (Press et al. 1993 pp. 405), and Fast Local Search (FLS) (Tsang and Voudouris 1997).

2.1.1.1 Bracketing search methods

Bracketing search methods use the relative merits of additional potential solutions compared to the current potential solution in order to produce a better solution and to reduce the number of function evaluations needed to reach the local optimum. Among the efficient bracketing methods are the *golden section method*, and *Brent's method.*

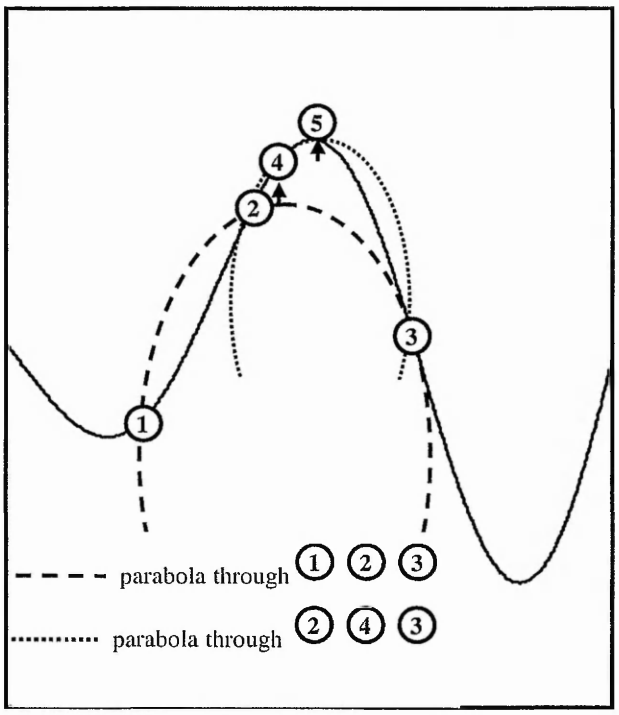

Figure 2.1: Convergence to a Maximum by Brent's Method.

The *golden section* method is similar to the bisection method. It starts with two solutions that bracket the local optimum. This method selects the next potential solution so that it is within a distance of 0.38917 from one solution and 0.61803 from the other (Press et al. 1993 pp.399) in order to minimise the worst-case possibility. After evaluating the merit of the new produced solution, the two best potential solutions are selected as the new bracketing solution. The process is repeated until the best solution converges at the optimum.

On the other hand, *Brent's method* starts with three potential solutions. The method fits a parabola to those solutions. It then uses the maximum of the parabola as the next potential

solution of the overall function. The method iteratively narrows the bracket based on the position of the new solution relative to the others (figure 2.1).

2.1.1.2 Gradient methods

These methods use the gradient of the objective function at the potential solution, to direct the search towards an improved solution in order to speed the convergence. There are many methods that rely on the gradient and for detailed information on variants of these methods the reader can refer to (Press et al. 1993).

If the objective function is sufficiently smooth at the current potential solution, the algorithm can use the directional derivative to proceed in the direction of the steepest ascent in order to reach the optimum. This can be achieved through finding the angle around the potential solution for which the magnitude of the derivative of the objective function with respect to some step size is maximised. The maximum occurs in the direction of the gradient $-\nabla f(x)$. The steepest ascent method can generate a new potential solution from the current one using the following formula:

$$x_{k+1} = x_k + \alpha_k \nabla f(x_k) \tag{2.5}$$

where $k \geq 0$, $\nabla f(x_k)$ is the gradient at x_k and α_k is the step size.

The right step size is critical to guarantee the best rate of increase in the objective value of potential solutions over several iterations. The *Newton method* incorporates second-order derivative information into the above formula to find the optimum of a quadratic basin in a single step. The following formula is used in the Newton method to generate new solutions.

$$x_{k+1} = x_k + (\nabla^2 f(x_k))^{-1} \nabla f(x_k) \tag{2.6}$$

where $\nabla^2 f(x_k)$ is the Hessian matrix

$$\nabla^2 f(x_k) = \begin{bmatrix} \frac{df^2}{dx_1^2} & \frac{df^2}{dx_1 dx_2} & \cdots\cdots & \frac{df^2}{dx_1 dx_n} \\ \frac{df^2}{dx_2 dx_1} & \frac{df^2}{dx_2^2} & \cdots\cdots & \frac{df^2}{dx_2 dx_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{df^2}{dx_n dx_1} & \frac{df^2}{dx_n dx_2} & \cdots\cdots & \frac{df^2}{dx_n^2} \end{bmatrix}_{x_k} \tag{2.7}$$

The Newton method requires calculation of the inverse Hessian matrix, which is a time consuming task. *Quasi-Newton methods* estimate the inverse Hessian instead of calculating it using different techniques. The *Newton-Gauss method*, for example, uses Gaussian elimination to generate the inverse Hessian.

2.1.1.3 Fast local search

Tsang and Voudouris (1997) proposed the Fast Local Search (FLS) algorithm as a refinement of a local search that adapts its neighbourhood. It can speeds up the search by shadowing less promising parts of the neighbourhood. The neighbourhood is divided into a number of smaller sub-neighbours, which can be either active or inactive. Initially all sub-neighbourhoods are active. FLS visits the active sub-neighbourhoods in some order. If a sub-neighbourhood is examined and does not contain any improving solution, it becomes inactive. Otherwise, it remains active and the improving solution is accepted. This may cause some sub-neighbourhoods to be reactivated, if they are thought to contain improving solutions because of the change in the current potential solution. As the potential solution improves, more and more sub-neighbours become inactive, and when all sub-neighbourhoods have become inactive, the best solution found is the local optimum.

2.1.2 Avoiding the trap in a local optimum

The probability of being trapped in a local optimum is high for search methods that use a neighbourhood information as a basis for its search. This probability can be increased further by utilising speed up techniques that use additional local information. Variants of the model of local search have been proposed which reduce this probability. Variations can happen at different phases of the search. It can happen during neighbourhood generation, in the course of formulating the rule of accepting for new potential solutions or the stopping criterion.

The simplest variant of local search is multi-start where the search is repeated from a new starting potential solution. By choosing a new starting potential, the search algorithm is allowed to continue and locate a potentially different local optimum. The result of this algorithm is the best local optimum encountered over all runs of the algorithm, which can be the global optimum. Dynamic Hill Climbing (DHC) (Yuret and de la Maza 1993) is a multi-start local search, which uses a diversity-based distance metric to ensure locating a new local optimum.

Simulated Annealing (SA) (Kirkpatrick 1983), Tabu Search (TS) (Glover 1989) and Guided Local Search (GLS) (Tsang and Voudouris 1997) utilise different techniques to escape local optima. These search methods, which can be considered as neighbourhood search methods, aim to avoid local optima by using heuristic methods that allow non-improving moves to be made. When examining the neighbourhood around the current potential solution, the next candidate potential solution is not necessarily the neighbour with a better fitness value.

The basic ingredients of these variants are described in brief in the following subsections.

2.1.3 Dynamic hill climbing

Dynamic Hill Climbing (DHC) algorithm (Yuret and de la Maza 1993) is a multi-start local search that is designed to force the local search method to explore the search space uniformly. The algorithm chooses the starting points of the local search independently and as far as possible from the points that are already evaluated to maintain the diversity of the search to avoid local optima. The local search in a DHC method tries to find a local optimum in the neighbourhood of the starting point. The direction of the move is selected randomly with a pre-selected step length. If the move improves the current solution then the step length is doubled and the move is pursued in the same direction. On the other hand, the step length is halved in cases where the move worsens the current solution. The next longest step is then tried in the same direction. The reduction in step size increases the accuracy of the optimum when the search approaches a local optimum. The process iterates until no further improvement in the current solution is possible. The DHC continually seeds the local search with new points to start from. The process is repeated until the stopping criteria are satisfied.

DHC algorithm has found practical applications ranging from medical imaging (De la Maza and Yure 1995) to the energy minimisation problem for determining the shape of organic molecules, the travelling salesman problem, and the problem of assessing the structure of sedimentary deposits using seismic data (Yuret 1994) .

2.1.4 Guided local search

Guided Local Search (GLS) algorithm (Tsang and Voudouris 1997) extends local search methods with the concept of features, i.e. a set of attributes, which characterise a solution to the problem. It assumes that any solution can be described by a set of features owned by a specific solution. Each feature is assigned a cost value. Features should be defined so that the presence of a feature in a solution affects the value of the objective function through the feature's cost. A feature with a high cost is not attractive. Initially, all the features have a zero feature cost. Whenever the search method reaches a local optimum, the GLS modifies the cost of all the features owned by the local optimum in order to induce the search to explore other regions which do not exhibit costly features. A GLS has been successfully applied to a number of hard combinatorial optimisation problems, and real world problems. They have also been applied to difficult continuous optimisation problems (Voudouris and Tsang 1995) (Voudouris 1998).

2.1.5 Tabu search

Building upon his previous work, Glover proposed in 1989 a new heuristic search method, which he called Tabu Search (TS). Glover described it as a meta-heuristic optimisation method whose role is to guide and orient the search of another heuristic specially tailored to the problem at hand. Many computational experiments have shown that the tabu search has become an established optimisation technique, which can compete with most known techniques (Glover 1990).

The basic principle of TS is to pursue the search even when it encounters a local optimum by accepting non-improving solutions. At the same time, the systematic use of memory helps to avoid cycling back to already sampled solutions in the neighbourhood of current potential solution. This short-term memory, called tabu lists, records the recent history of the search. The use of short-term memory helps TS to perform extensive exploration. It also makes the structure of the neighbourhood dependant on the current state of search. For this reason TS can be described as a dynamic neighbourhood search technique (Hertz et al. 1995)

The TS in its basic form, is a combination of a local search method with short-term memory. The elements of the memory are called tabus (disallowed moves). Tabus, as mentioned above, are stored in the tabu list. Usually, only a limited quantity of neighbourhood information is recorded. Unfortunately, a large memory, which is computationally expensive to search, is required to store all solutions, therefore it is seldom used. The most commonly used tabus involve recording the last few modifications performed on the current solution. The probabilistic TS can significantly reduce the tabus list length and the computational cost of checking, however excellent solutions maybe missed (Hertz et al. 1995).

Tabu lists are usually implemented as circular lists of fixed length (Gendreau 2003). Some authors, however, have proposed varying the tabu list length during the search (Glover 1989) (Joines et al. 2000a).

When there is no danger of cycling, aspiration criteria that allow revoking tabus are used to guarantee that a tabu list is not prohibiting attractive moves. The most commonly used aspiration criterion is one that allows a move regardless of being in the tabu list, if that move can produce a solution better than the current best solution. More complicated aspiration criteria have been used (Lin and Miller 2004a).

The TS may use intermediate-term memory to gather information about promising regions of the search space. In the case that the TS discovers such an area, normal search halts and an intensive search is performed to find the best solutions in the current region. On the other hand, a TS algorithm may use long-term memory to record the total number of iterations that various solution components have been involved in. The aim is to prevent the search from spending most of its time in a restricted portion of the search space and guide it to more interesting parts of the search space. When unexplored parts are noticed, the TS algorithm uses a diversification mechanism, such as restart and continuous diversification, to force the search into these areas. The TS integrates intensification and diversification to explore the search space efficiently.

TS are well adapted to discrete optimisation for which a finite set of moves can be used to reach any solution from any other solution in a finite number of moves. The TS can be straightforwardly applied to continuous functions by choosing a discrete encoding of the problem. It can also be modified to suit continuous problems without the need for discrete encoding. Instead of forbidding already visited solutions, the algorithm prevents visiting solutions that are close to already visited ones (Rolland 1997) (Lin and Miller 2004b).

2.1.6 Simulated annealing

Simulated Annealing (SA) (Kirkpatrick et al. 1983) is a stochastic optimisation method that avoids the trap in a local optimum by accepting non-improved solution based on principles of condensed matter physics. This technique has been successfully applied to different optimisation problems.

The concept of SA is based on the manner in which liquids freeze or metals re-crystallize during the process of annealing. In an annealing process, a melt, initially at high temperature and disordered, is carefully and slowly cooled so that the system at any time is approximately in thermodynamic equilibrium. As cooling proceeds, the system becomes more ordered and approaches a frozen ground state at $T=0$. Hence, the process can be though as an adiabatic approach to the lowest energy state.

Two conditions are essential for the system to attain the ground state. The initial temperature must not be very low and it must be decreased at a sufficient slow rate. The annealing time must be long enough to allow any necessary transformations to take place and prevent the system from forming defects or freezing out in meta-stable states (trapped in a local minimum energy state).

In any heated metal sample and at temperature T, the probability of a cluster of atoms to exist at configuration i, which is defined by a set of atomic positions $\{r_i\}$ and an energy state $E\{r_i\}$, is defined by the Boltzmann probability factor:

$$P(E\{r_i\}) = e^{\frac{E\{r_i\}}{k_B T}} \tag{2.8}$$

where k_B is Boltzmann's constant. Cooling the metal slowly makes atoms move between relatively higher and lower energy levels and allows them to equilibrate at each temperature T. The material will reach the ground state (global optimum), a highly ordered form in which the probability of the existence of a high energy state throughout the material is very little.

The slow movement towards an ordered ground state of the physical system is similar to progression to a global optimum in a system where the energy function is replaced by the objective function, $f(x)$ to be optimised. To simulate the annealing behaviour, a control parameter T that determines the stability of a potential solution must be specified and the criteria of accepting a new potential solution in the standard iterative local search method should be modified. If there is no improvement in the merit or the fitness of the current generated solution, x_t , compared to the current potential solution, x_{tp} , the probability of accepting it as a potential solution is determined by the Boltzmann probability distribution as given in equation 2.9.

$$P(x_t) = e^{\frac{f(x_t) - f(x_{tp})}{T}} \tag{2.9}$$

This probability is compared against a randomly generated number over the range [0, 1]. If this probability is greater than the generated number, the current generated solution is accepted as the current potential solution. Repeating this iterative improvement many times at each value of the control parameter T, the methodical thermal rearrangement of atoms within a metal at temperature T is simulated. The value of the control parameter T is initially set high and is periodically reduced according to a cooling schedule.

One of the difficulties in using simulated annealing is that it is very difficult to choose the rates of cooling and the initial temperatures (Davis and Steenstrup 1987). The selection of these parameters depends on heuristics and varies with the function to be optimised. In addition, the number of function evaluations required to slightly improve the current solution dramatically increases near the global optimum and as the temperature becomes low (Davis and Steenstrup 1987).

2.2 Population-based search methods

In contrast to single-point search methods, which keep refining a single solution until no further improvement can be achieved, population-based algorithms maintain a population of candidate solutions. Using a population of candidate solutions helps search algorithms to avoid being trapped in a local optimum and consequently can often find global optimal solutions. They are also well fitted for multi-objective problems.

Two of the most successful and widely recognised population-based optimisation methods are reviewed. Ant Colony Optimisation (ACO) and Genetic Algorithms (GA) are discussed in the following sections.

2.2.1 Ant colony optimisation

Real ants are capable of finding the shortest path from their nest to a food source without visual sensing. Ants deposit a substance called pheromone while walking, forming a pheromone trail. Ants can smell pheromone and when choosing a route, they probabilistically tend to follow paths which are rich in pheromone. The pheromone trail allows the ants to find their way back to the food source or to the nest. It can also be used by ants to find the location of food sources found by their nest mates.

The pheromone trail following behaviour enables the ants to discover the shortest paths. The ants that happen to pick the shorter path will create a strong trail of pheromone faster than the ones choosing a longer path. This stimulates successive ants to choose the shorter path until eventually all ants have found the shortest path. The pheromone trail following behaviour also explains the ants' ability to adapt to changes in the environment, such as new obstacles interrupting the currently shortest path.

Ant colony members exchange information using a simple form of indirect communication mediated by pheromone formation, known as stigmergy (Dorigo et al. 1999). This form of stigmergtic communication plays a crucial role in ant foraging behaviour. The combined effect of an autocatalytic (positive feedback) mechanism (Dorigo et al. 1991) and implicit solution evaluation complements the stigmergtic communication role in the emergent of shortest path-finding behaviour. Implicit solution evaluation is based on the fact that shorter paths will be completed earlier than longer ones and therefore they will receive pheromone reinforcement more quickly. Implicit solution evaluation together with an autocatalytic mechanism can be very effective (Dorigo et al. 1996). The shorter the path, the sooner the pheromone is released resulting in more ants using the shortest path.

Ant foraging behaviour is a kind of distributed optimisation mechanism in which each single ant contributes to find the shortest path to food sources. Although a single ant is capable of finding a path between nest and food source, it is the ant colony which finds the shortest path.

Dorigo et al. (1991) proposed that stigmergtic communication can be applied to solve difficult optimisation problems. Based on this, they proposed the Ant System (AS) to solve the Travelling Salesman Problem (TSP). Several improvements have also been applied to this algorithm (Dorigo and Di Caro 1999). These improved versions of the AS can be described as population-based optimisation algorithms that are inspired by the behaviour of natural ant colonies, in the sense that they solve their problems by population cooperation using indirect communication through modifications in the environment.

Ant algorithms were first proposed as a multi-agent approach to difficult optimisation problems such as the TSP (Dorigo et al. 1991) (Dorigo and Di Caro 1999) and the quadratic assignment problem (Maniezzo et al. 2004). There is currently much ongoing activity to extend and apply ant-based algorithms to many different discrete optimisation problems (Dorigo et al. 1999). Recent applications include problems such as vehicle routing (Maniezzo et al. 2004) (Dorigo et al. 1999), sequential ordering (Maniezzo et al. 2004), graph colouring (Shawe-Taylor and Zerovnik 2001) and routing in communications networks (Di Caro and Dorigo 1998) (Dorigo and Di Caro 1999). A variety of other applications also exist.

2.2.1.1 Ant colony optimisation approach

Ant foraging behaviour can be easily applied to solve discrete optimisation problems by simulating behaviour through artificial means. In addition to being an abstraction of real ants, artificial ants can be enriched with some additional capabilities in order to make them more effective and efficient.

A population or a colony of artificial ants collectively searches for good solutions to the optimisation problem under consideration. According to the assigned notion of neighbourhood, each artificial ant performs a sequence of local moves in order to build a solution. It starts from an initial state selected according to some problem dependent criteria. It then continues to move through adjacent states until a solution is build.

An ant selects the next state from its adjacent states using a probabilistic decision policy. This policy makes use of local information, which can include in addition to the amount of

pheromone laid on the paths connecting adjacent states, problem specific knowledge and the ant past actions. Usually, the decision policy does not look ahead to predict future states.

Pheromone variables associated to problem states are used to represent the laid pheromone. These variables are used to store numeric information that represents the colony's current and past performance. This information is only available for ants that are accessing the state.

An artificial ant keeps an internal state which stores its past actions (local moves). The internal state can also store useful information to compute the quality of the global solution or the contribution of each executed move. Moreover, it can play a fundamental role in managing the feasibility of the solutions.

Initially, there is no pheromone on all solutions paths. Therefore, the probability of visiting adjacent states depends entirely on the ant's internal state and problem specific knowledge. An ant walks through adjacent states using the decision policy until it builds a solution.

The algorithm simulates pheromone trail formation by modifying numeric information stored in the pheromone variables of the path followed. In this way, the artificial ants modify their environment to reflect the past history of the whole ant colony. This form of stigmergetic communication plays a major role in the utilisation of collective knowledge. It changes the problem landscape according to past history of the ant colony.

Artificial ants can use online step-by-step, online delayed or a combination of both to release pheromone. In online step-by-step, ants release pheromone while they are building the solution. However, in online delayed, they deposit pheromone after building a solution by moving back to all visited states. The decision about which strategy should be used to release pheromone is problem dependent.

In general, the amount of pheromone deposited is made proportional to the quality of solution an ant has built (or is building) in order to induce the ants toward promising search regions. In this way, if a move contributes to the generation of a high quality solution, the amount of pheromone deposited will be proportional to its contribution. Ant algorithms can also use implicit solution evaluation to stimulate successive artificial ants into follow shorter paths.

As with any population-based optimisation algorithm, using autocatalysis can lead the ant colony search to premature convergence (Dorigo et al. 1999). For this reason, pheromone trail evaporation and stochastic state transition are usually employed within the optimisation process. An evaporation mechanism modifies pheromone information over time allowing the ant colony to slowly forget its past history, directing the search into new regions of the search space.

Once an ant has accomplished its task, which includes building a solution and depositing pheromone information, the ant dies. It contributes to the ant colony search by modifying the problem landscape according to the quality of the solution it found.

The search continues by creating a new ant which, in turn, builds a new solution and modifies the problem representation accordingly. This process continues until a termination condition is satisfied.

In addition to online updating of pheromone, ant algorithms can update offline. The algorithm can use global information, to deposit additional pheromone information, at the end of the algorithm iteration in order to bias the search from a global perspective. It can allow a "daemon" to observe the ant's behaviour in order to collect global information. It can also apply problem specific local optimisation methods, to deposit additional pheromone offline, based on the observation of all solutions generated by the ants.

It is possible to enrich ant algorithms with extra capabilities such as look ahead (Michel and Middendorf 1998) and backtracking (Di Caro and Dorigo 1998) in order to improve efficiency. Ants can also be hybridised with local search methods (Dorigo and Di Caro 1999) (Shmygelska and Hoos 2005).

Ant colony optimisation algorithms, as a consequence of their concurrent and adaptive nature, are particularly suitable for distributed stochastic problems where the presence of exogenous sources determines a non-stationary in the problem representation in terms of costs and/or environment.

2.2.2 Genetic algorithms

In nature, individuals, who have "good" genetic structures, have better chances of winning limited resources than their rivals. As a result, they have more chances to mate. The genetic structures of produced offspring are mixtures of the genetic structures of their parents. These structures are usually as good as that of their parents and can sometimes be better since there is a possibility of combining the parents' structures to produce new good

structures. The result of these operations and other genetic operations is the propagation of good genetic structures in the following generations and the gradual death of less successful structures. The goodness of a genetic structure depends on the features controlled by it. If these features are enabling an individual to survive in its environment, the related structure can be described as good structure; otherwise it is regarded as being less successful.

Natural evolution is an optimisation process (Fogel 1997) in which the quality of the species is maximised. It is, however, an open-ended dynamic process in which the quality of an individual can only be defined in relation to the environment in which it exists. Natural evolution optimises the functionality of individuals.

From an information science point of view, natural evolution can be regarded as a huge information processing system. Each individual carries its genetic information, which is referred to as the genotype. The interactions between the individual's genotype and its environment cause development of its character, which constitutes the phenotype, while an individual grows up. The genetic information is eventually passed on to the next generation if the individual shows traits that enable it to reproduce before it dies. The individual can be regarded as the mortal survival machines of potentially immortal genetic information (Corno et al. 1998).

The two main concepts of natural evolution, which are natural selection and genetic dynamics, have inspired the development of a population based search and optimisation technique. This method is known as a genetic algorithm. The basic principles of this technique were first laid down by Holland (Holland 1975) and are well described, for example in (De Jong 1975) (Goldberg 1989a).

2.2.2.1 Genetic algorithm basics and some variations

Genetic algorithms start with an initial population of individual structures or chromosomes. Each of these chromosomes represents a potential solution to a given optimisation problem. Each individual is assigned a fitness score based on its observed performance in solving a given problem. A high fitness score reflects a good characteristic that a specific solution exhibits. Individuals with a high fitness have more opportunity to become part of the mating pool, where some of the individuals are probabilistically selected for reproduction based on their fitness. Next, the genetic operators (usually mutation and crossover) are applied to the individuals in the mating pool producing offspring. The rates at which these operators are applied are an implementation decisions. If the rates are low enough, it is more likely that some of the offspring produced will be identical to their

parents. The two populations of children and parents are then merged to create a new generation. The result of applying a set of genetic operations that mimics natural selection and genetic dynamics is a new generation of individuals, which contain a higher proportion of good characteristics than the previous generation. Over many generations, good characteristics are spread though the population. They are mixed and exchanged with other favourable characteristics as the search progresses. The population will eventually converge to an optimal solution, if the genetic algorithm is well-designed (Beasley et al. 1993a).

Pseudo code for a genetic algorithm is shown in figure 2.2.

The first step involves the generation of an initial population of chromosomes which represent the potential solutions to the optimisation problem. The chromosomes represent genotypes that are manipulated by the genetic algorithm. Each chromosome consists of several genes and every gene or group of genes has some phonotypical meaning such as a parameter in the problem search space. The genes of each chromosome control the location of an associated solution in the problem search space. Normally, the initial population of chromosomes is generated randomly, although problem-specific knowledge can be used to influence its generation (Reeves 1993). In the canonical genetic algorithm, the potential solutions were encoded as binary strings, each gene consists of a group of bits and each gene represents a parameter in the problem search space. The initial population of the canonical algorithm is generated randomly.

In the evaluation part of a genetic algorithm, each individual is assigned a fitness score that reflects how far that individual is from the optimum compared to other individuals. The fitness assignment is performed by mapping the genetic structure to a point in the phenotype domain and then evaluating this point using the function to be optimised.

The selection mechanism probabilistically selects the fittest chromosomes from the population to survive and mate. These individuals represent near-optimal solutions. A variety of selection schemes has been used. Holland proposed a fitness proportional selection mechanism, where the probability of selecting any chromosome for mating is calculated by dividing the chromosome fitness by the total fitness assigned to all the chromosomes in the population.

```
Begin
   t = 0
   initialise(Population(t))
   evaluate(Population(t ))
   while termination criteria not satisfied
   Begin
      t = t + 1
      MatePool(t) =select(Population(t-1))
      MatePool(t) =crossover (matePool(t))
      MatePool(t) =mutate(matePool(t))
      evaluate( MatePool(t) )
      Population(t)=merge (MatePool(t),Population(t-1))
   End
End
```

Figure 2.2: Pseudo Code of a Genetic Algorithm.

Fitness scaling, fitness ranking (Baker 1985) (Whitley 1989), and tournament selection (Goldberg and Deb 1991) techniques have been proposed as alternatives to the proportional selection mechanism in order to overcome associated problems (Beasley et al. 1993a) (Hopgood 2001 pp.180). The first problem is that the selection pressure of this technique becomes very weak as the population converges upon a narrow range of values. Fitness scaling techniques have been proposed to avoid this problem where the relative fitness of the individual is used in calculating the selection probability instead of the absolute fitness. Fitness scaling, however, can aggravate another problem associated with a fitness proportional selection mechanism. The selection pressure of the fitness proportional selection mechanism with the existence of a highly fit individual (but not the optimal) in the population can make that individual rapidly take over the population and lead to premature convergence (Beasley et al. 1993a). Rank-based selection, where the chromosomes in the population are ranked and the probability of selection is a function of rank rather than fitness, was proposed to solve this problem. If the selection probability of a chromosome is proportional to its rank, the rank is referred to as linear, otherwise it is nonlinear. Nolle et al. (2000) proposed a nonlinear rank strategy with the ability of adapting its selection pressure online through controlling a specific control parameter. Tournament selection is another proposed scheme, where a small set of chromosomes is chosen at random and the best chromosome is selected for mating. This technique is less susceptible to premature convergence and its selection pressure can be adjusted by controlling the size of the set. Ranking and tournament selections are the natural choices for problems in which

it is difficult to precisely specify a fitness function (Grefenstette 1997). Boltzmann selection mechanisms (Mahfoud 1997) control the selection pressure of the genetic algorithm based on principles from simulated annealing to indefinitely prolong the search in order to locate better final solutions.

Elitist strategy is a scheme to bias the selection, where the best chromosome in the parent population is chosen and all but one of the children's population (Eshelman 1997). It biases the search to exploit the genetic information of the best chromosome found so far.

Due to the selection mechanism's important role in guiding the genetic search and maintaining a high genotype diversity (Bäck and Hoffmeister 1991), several researchers have studied how different selection schemes affect the algorithm's performance (Goldberg and Deb 1991) (Bäck and Hoffmeister 1991) (Zhang and Kim 2000)(Goldberg and Sastry 2001).

The aim of a crossover operator is to recombine the good features, which are scattered through the mating population, into better chromosomes (Eshelman 1997). There is no guarantee that crossover will always produce better chromosomes, however the existence of the selection operator eventually discards children with less favourable features. The crossover is probabilistically applied to randomly selected pairs from the parent mating pool. Crossover is not usually applied to all chromosomes of the mate pool. However, a high fraction of them undergo crossover. Holland originally proposed the one-point crossover, where a randomly selected point is picked in the genetic structure and new children are generated by swapping the segment at that point between the two parents. Other crossover operators have been devised, often involving more than just one point. Using a two-point crossover operator can improve the performance of the genetic algorithm compared to a single point operation. More than two, however, can reduce the performance (De Jong 1975 cited in Goldberg 1989a). Multi-point crossover can disrupt the building-blocks and at the same time can explore the search space more thoroughly. Uniform crossover is another type of crossover where each gene is created by copying the corresponding gene from a parent according to a randomly generated crossover mask. Uniform crossover has the advantage that the ordering of genes is entirely irrelevant (Syswerda 1989).

Eshelman et al. (1989) investigated the effect of the different crossover operators on the genetic algorithm performance theoretically, in terms of positional and distributional bias, and empirically, using several problems. They found no overall winner. Reduced surrogate

crossover (Booker 1987) was introduced to increase the crossover productivity as the population converges. The productivity of uniform crossover makes it more suitable for small populations (De Jong and Spears 1992). However, reduced surrogate two-point crossover is more suitable for large populations.

Mutation is another genetic operator, which is applied to each of the offspring independently. In contrast to crossover, a mutation operator alters one or more genes of the chromosome to produce a new chromosome. This operator is typically applied with a low probability.

The mutation operator has a great influence on the genetic algorithm search. Mutation guarantees that no point in the search space has a zero probability of being visited. It helps to prevent the permanent loss of useful gene values that may be accidentally lost during the search. In addition, the mutation plays an important role in making small refining moves that are not efficiently made using crossover and selection alone (Rosin et al. 1997). Mutation can be very effective in solving many optimisation problems even when used without crossover (Eshelman 1997).

The order of genes within a chromosome is critical in order for the building-block hypothesis (Goldberg 1989a) to work effectively. An inversion operator (Holland 1975), which works by reversing the order of genes between two randomly chosen positions within a chromosome, was suggested in attempts to find gene orderings which have better evolutionary potential (Goldberg 1989a p166). The reorder process expands the search space. In addition to the genetic algorithm search for good sets of gene values, it is simultaneously optimising the gene ordering too (Beasley et al. 1993b). The use of uniform crossover can eliminate the need for reordering (Syswerda 1989).

After new offspring have been created, the genetic algorithm uses a replacement mechanism to merge both parents and children populations in order to produce the next generation. The most commonly used replacement techniques are generational and steady state. In generational genetic algorithms, the population of parents is completely replaced by the children population to produce the new generation. If this mechanism is used without elitist strategy there is a risk of losing good building blocks for ever. In the steady state mechanism, only one mating per cycle is allowed to replace a pair of parents. This gives the genetic structures of the parents the chance to compete and mate with that of their children. Steady state genetic algorithms suffer a higher gene loss than do their generational counterparts (De Jong and Jayshree 1992). The parent selection and replacement strategies

must complement each other in terms of the overall effect they have on the exploration-exploitation balance (De Jong and Spears 1993). The replacement technique can be biased toward the fittest chromosomes and can lead to premature convergence if combined with biased reproduction. An unbiased version can protect the search from being trapped in local optima (Eshelman 1997). Other merge biased techniques have been borrowed from evolution strategies (Bäck et al. 1991). For example, Eshelman (1991) combined the $(\mu + \lambda)$ replacement selection technique with unbiased reproductive selection, to select the best individuals from both parents and children populations. Mühlenbein and Schlierkamp-Voosen (1993) used (μ, λ) in their breeder genetic algorithm to produce λ offspring $(\lambda > \mu)$ and the best μ offspring are chosen to replace the parents population.

2.2.2.2 Schema theorem, implicit parallelism and building blocks hypothesis

While each genetic operator is simple to understand independently, the resulting behaviour of the genetic algorithm can be quite complex. The notion of schema processing is used to explain the behaviour of genetic algorithms and to justify their ability to search.

A schema is a pattern of gene values which may be represented (in binary code) by a string of characters in the alphabet {0, 1, #}. A particular chromosome is said to contain a particular schema if it matches that schemata, with the '#' symbol matching anything. So for example, the chromosome '1101' contains among others, the schemata '1##1','#1#1','11#1','110#'. The order of a schema is the number of '0' and '1' symbols it contains (2, 2, 3, 3 respectively in the example). The defining length of schema is the distance between the outermost non-# symbols (4, 3, 4, and 3 respectively in the example).

The notion of schema processing is used to illustrate the property of implicit parallelism where a large quantity of schemata is being processed simultaneously while processing a relatively small quantity of chromosomes. The property can be explained as follows. Since each chromosome contains many different schemata (2^L where L: chromosome length), the processing of a single chromosome is, in fact, a processing of all the schemata that are contained in that chromosome.

Assuming that an individual's high fitness is due to the fact that it contains good schemata, the power of a genetic algorithm can be explained according to the schema theory. This theory states that a particular schema receives trails according to the ratio of schema fitness to population average fitness as long as the schema is not disrupted by crossover or mutation. As a consequence short, low-order, above average schemata receives

exponentially increasing trials in subsequent generations of a genetic algorithm. Such schemata are called building blocks.

Goldberg (1989a) believes that the primary source of the genetic algorithms search power is their ability to find good building blocks that when combined together can produce high-order schemata with better fitness. The previous hypothesis is known as the building block hypothesis. Goldberg imposed two conditions on a genetic algorithm in order to be effective as predicted by schema theory. The related genes should be close together in a chromosome and the interactions between genes should be little. These two conditions are known as the recommendation of the building block hypothesis.

The schema theorem provides good bounding advice on how to assure the growth of good schema and how growth can be sustain to takeover the population (Goldberg and Sastry 2001). Bridges and Goldberg (1987) tried to derive an exact formula for schemata propagation under specific assumption by extending an exact formula for the expected propagation of chromosomes in genetic algorithm, under selection and crossover. Attempts have been done to derive a generalised schema theorem (Goldberg 1987) (Goldberg and Deb 1991) (Whitley et al. 1992). Goldberg and Sastry (2001) show that the schema theorem works with different selection schemes and genetic operators. They explore its ramification for the choice of selection operator and parameterisation of the algorithm.

2.2.2.3 Performance of genetic algorithms

The performance of a pure genetic algorithm as any global optimisation algorithm depends on the mechanism for balancing the two conflicting objectives, which are exploiting the best solutions found so far and at the same time exploring the search space for promising solutions. The power of genetic algorithms comes from their ability to combine both exploration and exploitation in an optimal way. Holland (Goldberg 1989a, pp.36) draws an analogy between the behaviour of genetic algorithms and a k-armed bandit problem with unknown payoff distribution. The similarity then was used to show that the exponential allocation of trials according to observed performance offers genetic algorithms with near-optimal sampling.

However, although this optimal utilisation may be theoretically true for a genetic algorithm, there are problems in practice. These arise because Holland assumed that the population size is infinite, the fitness function accurately reflects the suitability of a solution, and the interactions between genes are very low (Beasley et al. 1993a).

In practice, the population size is finite, which influences the sampling ability of a genetic algorithm and as a result affects its performance. The consequences of using limited population sizes on the performance of genetic algorithms, the problems associated with it and the impact of hybridising a genetic algorithm with other search techniques to alleviating these problems are discussed in this section.

Stochastic sampling is used to alleviate the consequences of finite population size. As a result, the performance of a genetic algorithm will be subject to stochastic errors. The accumulation of stochastic errors causes the population to converge at a single point in the search space, even in the absence of selection pressure. This problem is known as genetic drift (Thierens et al. 1998). The rate of genetic drift provides a lower-bound on the rate at which a genetic algorithm can converge towards the optimal solution. The rate of convergence of the genetic algorithm must be sufficiently large to counteract any genetic drift. The genetic drift can be slowed down by increasing the mutation rate which also can slow down the convergence to the global optimum. The solution to resist the genetic drift is to find a way to maintain diversity in the population without decelerating the search.

Incorporating a local search method within the global genetic algorithm can be a solution for combating the effect of the genetic drift. Applying a local search method to a solution can introduce new genes into the population without decelerating the search. The proper use of a local search method can also accelerate the search towards the global optimum (Hart 1994). This, in turn, can guarantee that the convergence rate is large enough to obstruct any genetic drift.

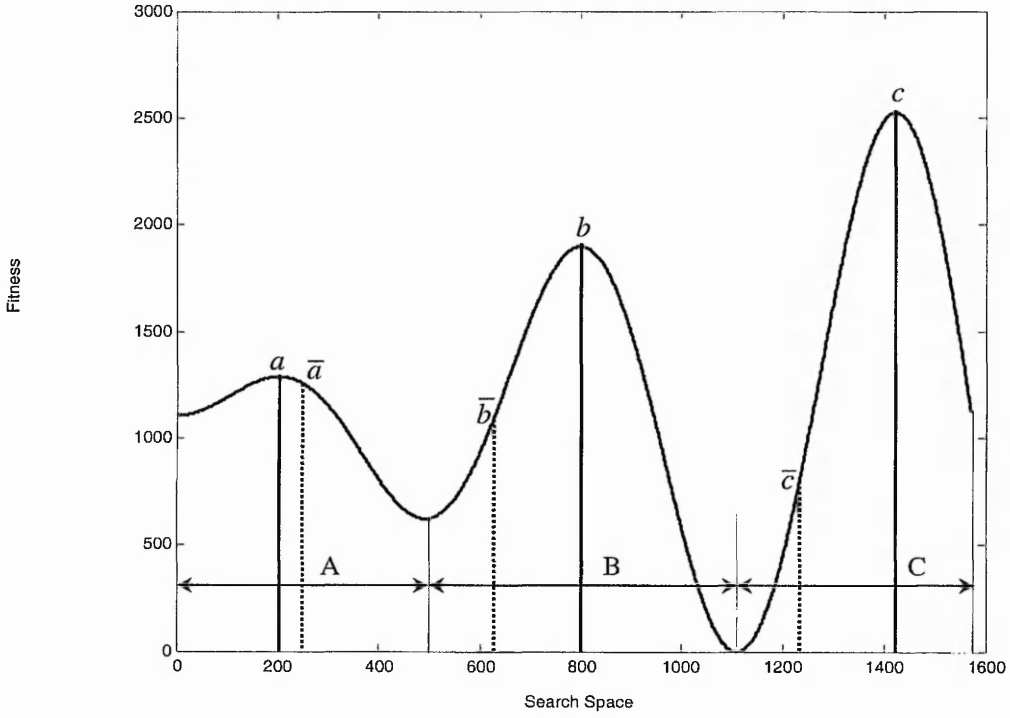

Figure 2.3: Samples Can Misguide the Genetic Search.

Limited population size can affect the sampling ability of a genetic algorithm in other ways. It can affect exploitation of information which is gathered by the algorithm to guide the search to the most promising regions. A genetic algorithm may sample bad representatives of good search regions and at the same time good representatives of bad regions. In figure 2.3, for example point $\bar{a}$ is a good representative of region A, while points $\bar{b}$ and $\bar{c}$ are bad representatives of regions B and C, respectively. Based on the samples, a genetic algorithm can direct the search to a local instead of a global optimum. In other words, the sampling error caused by a genetic algorithm with limited population size can cause premature convergence. The solution to avoid such problems is to ensure that each region is represented by a solution that reflects the region's fitness. If a genetic algorithm is forced to sample only the local optimum of each basin of attraction (points a, b and c in figure 2.3), its sampling ability can be improved which in turn reduces the possibility of premature convergence. This means mapping all the points of the search space to their local optima. In which case, the role of the genetic algorithm becomes limited in guiding the search toward the global optimum among local optima.

Mapping the current solutions of the genetic algorithm to local optima can be accomplished by applying a local search method to the solutions. Regardless of mapping details, the mapping can ensure fair representation of the different search areas and help to fight premature convergence.

A finite population size can cause a genetic algorithm to produce solutions of low quality compared with the quality of solution that can be produced using local search methods. The difficulty of finding the best solution in the best found region accounts for the genetic algorithm operator's inability to make small moves in the neighbourhood of currents solutions (Reeves 1994). The use of the blind mutation operator with a high rate in order to introduce diversity into population can lead the search to re-explore already visited regions. Michalewicz (1996 pp.108) proposed a non-uniform mutation operator that explores the search space during early stages of the genetic algorithm and refines the solutions in latter stages.

Utilising a local search method within a genetic algorithm can improve the exploiting ability of the search algorithm on the condition that it does not limit the exploring ability of the genetic algorithm (Hart 1994). If the right balance between global exploration and local exploitation capabilities can be achieved, the algorithm can easily produce solutions with high accuracy (Lobo and Goldberg 1997).

Although the rate of convergence is fast during the early stages of the genetic algorithm, a drastic reduction in convergence rate in latter generations is often encountered before the genetic algorithm provides an accurate solution. The reason for the change in convergence rate is that genetic algorithms can rapidly locate the region in which a global optimum exists, and take a relatively long time to locate the exact local optimum in a region of convergence (De Jong 2005). A combination of a genetic algorithm and a local search method can speed up the search to locate the exact global optimum. In such hybridisation, applying a local search to the solutions that are guided by a genetic algorithm to the most promising region can accelerate convergence to the global optimum. The time needed to reach the global optimum can be further reduced if the local search methods and local knowledge are used effectively (Hart 1994). A local search method can provide a genetic algorithm with good representatives of the different regions of the search space (Gruau and Whitley 1993) and accelerate locating the global optimum starting within its basin of attraction.

In a pure genetic algorithm, the appropriate balance of exploration and exploitation required for good performance depends on the amount of diversity in the population, the details of the genetic operators and the problem to be optimised. This balance is usually achieved by selecting suitable values of the genetic algorithm's control parameters such as population size and crossover, as well as mutation probabilities and selection pressure. The relationship between control parameter values and search performance is complex, not completely understood, and problem dependent (Eiben et al. 1999).

A large population size induces the search to perform more exploration which means slow convergence. On the other hand, populations of a small size can converge with a faster rate but its associated limited diversity can cause premature convergence. The optimal population size depends on the complexity of the search domain (Harik et al. 1999).

A high selection pressure can push the search toward fast exploitation of the information gathered and expose the search to premature convergence problems. A low selection pressure pushes towards the other side of the equilibrium equation. The choice of selection scheme can control the rate of genetic drift which affects the convergence rate (Rogers and Prügel-Bennett 1999). The chances of premature convergence increase when a high selection pressure is combined with a small population size.

Standard mutation and crossover operators are simply two forms of more general exploration operators that can perturb genes based on any available information (Spears 1992). In addition, both operators have an exploitation role. The mutation operator exploits the neighbourhood of current solutions to construct new longer building blocks. The crossover operator exploits the genetic structure of the current solutions to combine good building block into longer structures. The trend of both operators in performing either exploration or exploitation depends on the gathered information, the details of both operators, and the details of the genetic algorithm itself. The correct mix of these two operators is essential for the genetic search.

The limitation of genetic algorithms comes mainly from the improper choice of control parameters (Deb 1997). These methods are not expected to work on an arbitrary problem with any arbitrary control parameter setting. Depending on these parameters the algorithm can either succeed in finding a near-optimum solution in an efficient way or fail. Choosing the correct parameter values is a time-consuming task. In addition, the use of rigid, constant control parameters is in contradiction to the evolutionary spirit of genetic algorithm (Eiben et al. 1999).

Other search techniques can be utilised to set the values of these parameters while the search is progressing. The ability of fuzzy logic to represent knowledge in imprecise and non-specific ways enables it to be used to reason on knowledge that is not clearly defined or completely understood. This ability makes fuzzy logic a suitable choice for adapting the control parameters of a genetic algorithm. Fuzzy logic has allowed a group of researchers to devise ways of optimising performance and solution quality of genetic algorithms (Richter and Peak 2002). It is used to incorporate the many heuristics and techniques of experienced genetic algorithm researchers into fuzzy logic system in order to adapt the control parameters. The goal of such systems is generally to speed up the convergence of the genetic algorithm and/or obtain better quality solutions (Herrera and Lozano 2001).

Incorporating other search methods within the framework of a genetic algorithm can help to overcome most of the obstacles that arises when optimising problems as a result of finite population size. Hybridisation as a solution for some of the problems that face genetic algorithms when used to solve real world problems is the main topic of the following sections.

2.3 Hybrid genetic algorithms

Hybrid optimisation methods, as any hybrid system, are based on the complementary view of search methods (Hopgood 2001 pp.223). Different search methods can be seen as complementary tools that can be brought together to achieve an optimisation goal. The ultimate goal of any optimisation algorithm is to find the exact global optimum using minimum resources.

Hybrid genetic algorithms are genetic algorithms that incorporate one method or more to improve the performance of the genetic search. There are several ways in which a technique can complement the genetic search.

2.3.1 Capability enhancement

A technique can be utilised within a genetic algorithm to enhance search capabilities. A genetic algorithm is normally viewed as a global search method that can capture the global view of a problem domain. Different techniques can be incorporated within a genetic algorithm to improve its performance in different ways. When a genetic algorithm as a global search method is combined with a problem-specific method as a local search method, the overall search capability can be enhanced. The enhancement can be in terms of solution quality and/or efficiency. This performance can also be improved by ensuring production of feasible solutions in the case of highly constrained problems. This

thesis focuses on the global-local complementary view of genetic hybrids which have been variously referred to as memetic algorithms (Moscato 1989), genetic-local search methods (Yamada and Reeves 1998), Lamarckian genetic algorithms (Morris et al. 1998), Lamarckian search, and Baldwinian search (Julstrom 1999). How to improve the performance of this class of hybrids in optimising continuous functions is the subject of this research.

Function approximation techniques can also be incorporated in a genetic search to speed up the search. It is also possible to utilise other techniques to replace one or more of the genetic operators in order to overcome some of the problems that face genetic search.

2.3.1.1 Improving solution quality

Local search methods and genetic algorithms are usually viewed as two complementary tools. A local search algorithm's ability to locate local optima with high accuracy complements the ability of genetic algorithms to capture a global view of the search space. Holland (1975 cited in (Michalewicz 1996 pp.107)) suggested that the genetic algorithm should be used as a pre-processor for performing the initial search, before invoking a local search method to optimise the final population. Bilchev and Parmee (1995), for example, used their ant colony model for continuous search spaces as local search method to improve the quality of the solutions produced by a genetic algorithm in order to solve a heavily constrained real-world engineering design problem.

Performing local search on a genetic algorithm's population, as mentioned before, can introduce diversity and help to overcome the nemesis of drift stall. It enables fair representation of different search areas in order to fight premature convergence. Incorporating a local search algorithm also introduces an explicit refinement operator which can produce high quality solutions.

2.3.1.2 Improving efficiency

The efficiency of a local search in reaching a local optimum integrates the efficiency of a genetic algorithm, in isolating the most promising basins of the search space. Therefore, incorporating a local search into a genetic algorithm can result in an efficient algorithm. The efficiency of the search can be enhanced in terms of the time needed to reach the global solution, and/or the memory needed to process the population.

Efficiency in terms of the time needed to reach a solution of desired quality is a major concern in genetic algorithm design. In real-world problems, function evaluations are the most time consuming part of the algorithm. For example, the designer of today's complex

engineering systems usually relay on expensive computer analysis and simulation programs, where the execution time for a single function evaluation can be of the order of hours or days (Hacker et al. 2002). Finite element analysis (FEA), computational fluid dynamics (CFD), heat transfer and vehicle dynamic simulations are examples of such programs. For this reason, time is often measured as the number of fitness function evaluations. Hybridisation in addition to parallelisation (Cantu-Paz 1998), time utilisation (Goldberg 1999), and evaluation relaxation (function approximation) can be used to speed up a genetic search (Goldberg 2003).

Genetic algorithms often show significant improvement in search speed when combined with local search methods utilising domain or specific knowledge. There is an opportunity in hybrid optimisation to capture the best of both schemes (Lobo and Goldberg 1997). This is the reason why genetic hybrids are being increasingly used to solve real-world problems. Most of the local search methods reviewed in this chapter have been mixed with genetic algorithms in real-world applications (Yen et al. 1998) (Besnard et al. 1999) (Liang et al. 1999) (Preux and Talbi 1999).

Population size is crucial in a genetic algorithm. It determines the memory size and the convergence speed in serial genetic algorithms and affects the speed of search in the case of parallel genetic algorithms. Efficient population sizing is critical for getting the most out of a fixed budget of function evaluations. The gambler ruin's model (Harik et al. 1999) was used to estimate the population size of genetic algorithms. This model was used to show that population size depends on two parameters, which can be affected by incorporating local search. The two parameters represent the standard deviation of the population and signal difference between the best and second best building blocks. If a local search method is incorporated in such a way as to reduce the standard deviation of the population and to increase the signal difference between the best and the second best chromosome, the resulting hybrid can be efficient even with small population sizes. Espinoza et al. (2003a) showed the effect of incorporating a local search method on reducing the population size, compared with a pure genetic algorithm.

2.3.1.3 Guarantee feasible solutions

In highly constrained optimisation problems, the crossover and mutation operators generally produce illegal or infeasible solutions and hence waste search time. This problem can be solved by incorporating problem-specific knowledge. Problem-specific knowledge can be used either to prevent the genetic operators from producing infeasible solutions or to repair them.

The Partial Matched crossover (PMX) (Goldberg and Lingle 1985) was proposed for use in order-based problems to avoid the generation of infeasible solutions. Grefenstette et al. (1985) suggested a heuristic crossover operator that could perform a degree of local search for the Travelling Salesman Problem (TSP). Davidor (1991) designed "analogous crossover" where local information is used to decide which crossover sites can produce unfit solutions. Heuristic crossover operators were used to solve a timetabling problem in order to ensure that the most fundamental constraints are never violated (Burke et al. 1995). Freisleben and Merz (1996) proposed the Distance Preserving crossover (DPX) to produce feasible solutions to solve TSP without losing diversity. They used a non sequential 4-change as a mutation operator for the same reason. Cycle crossover (CX) (Oliver et al. 1987), Order crossover (OX) (Oliver et al. 1987), Matrix crossover (MX) (Homaifar et al. 1992), Modified Order crossover (MOX) (Wroblewski 1996), Edge Recombination crossover (ERX) (Whitley et al. 1989), 2-opt operator (Jog et al. 1991), 3-opt operator (Jog et al. 1991) and Or-opt operators (Jog et al. 1991) are examples of crossover and mutation operators which have been developed for the TSP. A special edge recombination crossover (Magyar et al. 2000) has been constructed for the three Matching Problem (3MP). The crossover operator has been replaced with the gene-pooling operator to produce feasible solutions when optimising the number and positions of fuzzy prototypes for efficient data clustering (Burdsall and Giraud-Carries 1997a).

A problem-specific knowledge search method can be used to recover the feasibility of solutions generated by the standard genetic operators. Repairing such solutions can help the genetic search to avoid the danger of premature convergence, which occurs when all or most solutions are infeasible (Orvosh and Davis 1993) (Ibaraki 1997). Konak and Smith (1999) combined a genetic algorithm with a cut-saturation algorithm for the backbone design of communication networks. They use a uniform crossover operator with a K-node-connectivity repair algorithm to repair infeasible offspring. Areibi and Yang (2004) used repair heuristics in their proposed approach to solve VLSI circuit layout. The approach combines a hierarchical design technique, genetic algorithms, constructive techniques, and advanced local search. They also used the OX operator to avoid infeasible solutions in solving VLSI design problems.

2.3.1.4 Fitness function estimation

If the fitness function is excessively slow or complex to evaluate, approximation function evaluation techniques can be utilised to accelerate the search without disrupting search effectiveness. This is because genetic algorithms are robust enough to achieve convergence in the face of noise produced by the approximation process. Fitness

approximation schemes, replace high-cost accurate fitness evaluation, with a low-cost approximate fitness assignment procedure. This can be achieved either by evolutionary approximation, where the fitness of a chromosome is estimated from its parents' fitness, or function approximation, where the fitness function is replaced by an alternate simpler model. Jin (2005) provides a comprehensive survey on fitness approximation techniques.

The selection of an appropriate approximation model to replace the real function is an important step in ensuring that the optimisation problem is solved efficiently. Neural network models have widely been used for function approximation (Lawrence et al. 1996). Willmes et al. (2003) compared neural networks and the Kriging method for constructing fitness approximation models in evolutionary algorithms. Jin and Sendhoff (2004) combined the k-nearest-neighbour clustering method and a neural network ensemble to estimate the solutions' fitness. Burdsall and Giraud-Carrier (1997b) used an approximation of the network's execution to evaluate solutions fitness instead of constructing a radial basis function network (RBF) to optmise the topology of a neural network. The approximation is based on an extension of the nearest-neighbour classification algorithm to fuzzy prototypes. Ankenbrandt et al. (1989) implemented a system of fuzzy fitness functions, to grade the quality of chromosomes, representing a semantic net. The system is used to assist in recognizing oceanic features from partially processed satellite images. Pearce and Cowley (1996) presented a study of the use of fuzzy systems to characterise engineering judgment and its use with genetic algorithms. They demonstrated an industrial design application where a system of problem-specific engineering heuristics and hard requirements are combined to form a fitness function.

2.3.1.5 Operation substitution

Genetic algorithms present a methodological framework that is easy to understand and handle. This framework is open to the incorporation of other techniques (Schwefel 1997). It is possible to utilise other techniques to perform one or more of the genetic algorithm operations. These incorporated techniques can be used to replace either the crossover operator, mutation operator or both.

In Probabilistic Model-Building Genetic Algorithms (PMBGA) or Estimation of Distribution Algorithms (EDA) (Pelikan et al. 1999b), a probabilistic model is utilised to learn the structure of a problem on the fly. This model is used instead of the standard genetic operators to ensure a proper mixing and growth of building blocks. These algorithms replace the standard crossover and mutation operators of genetic algorithms, by building a probabilistic model that estimates the true distribution of promising solutions.

New potential solutions are then generated by sampling this model. Population Based Incremental Learning (PBIL) (Baluja 1994), Univariate Marginal Distribution Algorithm (UMDA), Compact Genetic Algorithm (CGA), Bivariate Marginal Distribution Algorithms (BMDA), Factorized Distribution Algorithms (FDA) and the Bayesian Optimisation Algorithm (BOA) (Pelikan et al. 1999a) are all examples of PMBGA that are reported to have a better search ability, than that of the simple genetic algorithm, in solving a broad class of problems (Pelikan et al. 1999b). Tsutsui et al. (2005) proposed the Aggregation Pheromone System (APS), which introduced the concept of pheromone trail of the ant colony optmisation into the PMBGAs, to solve real-valued optimisation problems.

Leng (1999) proposed the Guided Genetic Algorithm (GGA), which is a hybrid genetic system that borrows the concept of feature and penalties from the Guided Local Search (GLS). The GGA modifies the fitness function by means of penalties to escape local optima. Two specialised crossover and mutation operators, which are biased by the penalties to change genes that are involved in more penalties, are used in order to explore the search space.

When a problem-specific representation is used in a genetic algorithm, the standard genetic variation operators are usually replaced with problem-specific operators. Hedar and Fukushima (2003) replaced the ordinary crossover with a simplex crossover that produces a simplex offspring from mating $n+1$ simplex parents (n is the dimension of the problem to be solved). They also used a mutation operator that was more suitable. Quantum-inspired genetic algorithms (Han and Kim 2002) (Han and Kim 2004) (Talbi et al. 2004) borrow the concepts of quantum bit and states superposition from quantum computing. In these algorithms, the individuals are represented as a string of quantum-bits. Quantum-gates are, then, used to modify these individuals instead of crossover and mutation operators. The power of these algorithms comes from the great diversity they provide by using quantum coding. Each single quantum individual in reality represents multiple classical individuals. The results reported from using this hybridisation to solve combinatorial and continuous optimisation problems are promising.

Tan et al. (1995) replaced the standard mutation operator by simulated annealing to solve system identification and linearization problems. The results showed a more accurate search and faster convergence when compared with a pure genetic algorithm. Riopka and Bock (2000) proposed a collective learning genetic algorithm, in which an intelligent recombination based on the exchange of knowledge between chromosomes, is used to effectively find high quality solutions to combinatorial optimisation problems. Magyar et

al. (2000) introduced several heuristic crossover and local hill-climbing operators to solve the 3MP. Fundamental to the technique is adaptation of operator selection. Two fuzzy connective-base (FCB) crossover operators types (dynamic and heuristic) have been proposed in (Herrera and Lozano 1996) for real-coded genetic algorithms to fight premature convergence problems.

2.3.2 Optimising the control parameters

The setting of genetic algorithm control parameters is a key factor in the determination of the exploitation versus exploitation trade-off. Other techniques can used to monitor the behaviour of a genetic algorithm in order to adapt its control parameters to improve search performance. A collection of fuzzy rules and routines can be used for dynamically adjusting the control parameters of genetic algorithms. A fuzzy logic controller uses feedback from the current state of search to improve performance and avoid undesirable behaviours such as premature convergence.

It is also possible to incorporate a genetic algorithm within another technique to optimise control parameters, since genetic algorithms are in practice very effective optimisation techniques. A genetic algorithm can be applied to optimise a neural network in a variety of ways. It can be utilised to adjust the neural network weights (Belew et al. 1991) (Montana 1995) (Liang et al. 2000) their topology (Miller et al. 1989) (Koza and Rice 1991) (Arena et al. 1993) (Chaiyaratana and Zalzala 2000) and learning rules (Chalmers 1990) (Fontanari and Meir 1991). For a comprehensive review of evolving neural networks the reader can refer to (Yao 1999). Karr (1991) described an application to the cart-pole balancing system and used a genetic algorithm to evolve the membership functions of a fuzzy controller. The resulting, optimised fuzzy logic controller performed better than the controller based on membership functions designed by a human expert. These promising results have been confirmed by an application of the method for online control of a laboratory pH system with drastically changing system characteristics (Karr and Gentry 1993). Genetic algorithms can also be used to automate the learning of fuzzy control rules (Valenzuela-Rendon 1991). They have also been used to optimise the control parameters of ant colony optimisation algorithms (White et al. 1998) (Botee and Bonabeau 1998) (Pilat and White 2002).

Some of design choices faced by a hybrid genetic algorithm designer while solving real world problems are discussed in the following sections. Due to their major impact on hybrid genetic performance, the discussion concentrates on different learning strategies and mechanisms that can be used to achieve a balance between exploration and exploitation.

First, the relation between genetic/local search and evolution/learning is presented. Then, different techniques that can be used to achieve the optimal division of labour between global genetic algorithm and local search method are reviewed.

2.4 Learning and local search

Organisms in different biological systems try to learn about themselves and their environments to acquire new skills and improve their innate characteristics in order to adapt to their environment and improve their chances of survival and reproduction. In this way, learning can increase an organism's chances at being selected to evolve. Performing a local search on a solution has a similar effect in hybrid genetic algorithms. A local search method uses local knowledge about a specific solution and its surrounding to improve its chances to be selected by the genetic algorithm to propagate its characteristics into the next generations. Since the genetic algorithm in itself is a model of the evolution process, the local search is usually viewed as learning process.

Evolution is concerned with the change in genetic structure of the population as a result of natural selection and genetic operators. Modification operators of genetic algorithms also work on the genotype or the genetic structure of the individual. However, the selection operator works on phenotype or the merit of traits that an individual shows in its environment. For this reason some kind of mapping from genotype to phenotype is embedded in genetic algorithms. In contrast to evolution and genetic algorithms, both leaning and local search methods work on phenotype. The other difference between evolution and learning is the time scale in which they occur.

Two basic biological learning models have been proposed to explain the way by which learning affect evolution. According to the Lamarckian model, learning can affect evolution directly through passing acquired traits as a result of learning from parents to their offspring. This model, which is known as Lamarckian evolution, was rejected by the Darwinian school of thought. This school believes that learning has indirect effect on evolution. Learning can guide evolution through an indirect mechanism, known as the Baldwin effect (Baldwin 1896). Learning can accelerate the genetic acquisition of learned traits without the Lamarckian mechanism. Through learning, individuals can improve their traits or their ability to adapt to their environment and this can increase their chances of survival and the passing of their genetic structures to next generations. The next generations will not be dominated only by individuals who have good genetic structures but also by individuals with the ability to learn and improve their fitness. This gives good genetic structures more chance to survive even when they are represented in the population by

individuals with under average fitness. Learning and evolution should aim to fulfil common goals in order for the Baldwin effect to occur. This condition is already satisfied in optimisation problems where the common task of local and genetic algorithm search is to optimise the same function.

The way of utilising gained information through local search within a hybrid genetic algorithm has a great impact on the performance of search process. Two basic approaches based on biological learning models have been adopted to utilise these information; the Lamarckian approach and the Baldwinian approach (Hinton and Nolan 1987). There is also a third model, which is a mixture of the basic models and its effectiveness has been proven in solving real-world problems (Orvosh and Davis 1993) (Houck et al. 1997) (Joines et al. 2000b) (Sung-Soon and Byung-Ro 2005).

2.4.1 Lamarckian learning

Lamarckian approach is based on the inheritance of acquired characteristics obtained through learning. This approach forces the genetic structure to reflect the result of the local search. The genetic structure of an individual and its fitness are changed to match the solution found by a local search method. In the Lamarckian approach, the local search method is used as a refinement genetic operator that modifies the genetic structure of an individual and places it back in the genetic population.

Lamarckian evolution, in spite of being recognised as never occurring in biological systems due to the lack of a mechanism to accomplish it, can be simulated in a computer in order to shed light on issues of general evolvability. Lamarckian evolution can accelerate the search process of genetic algorithms (Whitley et al. 1994). On the other hand, by changing the genetic structure of individuals, Lamarckian can disrupt schema processing which can badly affect the exploring abilities of genetic algorithms. This may lead to premature convergence (Whitley et al. 1994). When a Lamarckian approach is adopted, inverse mapping from phenotype to genotype is required. The inverse mapping may be computable in many simple applications. However, the computation will typically be intractable, for real-world problem solving (Tunery 1996). Most of hybrid genetic algorithms that repair chromosomes to satisfy constrains are Lamarckian and the technique has been particularly effective in solving TSP (Julstrom 1999).

2.4.2 Baldwin learning

The Baldwin learning allows an individual's fitness to be improved by applying a local search, whereas the genotype remains unchanged. In this way, it improves the

solution's chances to propagate its structure to the next generations. Like natural evolution, learning does not change the individual's genetic structure, however it increases its chances of survival. The Baldwinian approach, in contrast to the Lamarckian one, does not allow parents to pass their learned or acquired characteristics to its offspring. Instead, only the fitness after learning is retained. A local search method in the Baldwinian approach is usually used as a part of the individual's evaluation process. The local search method uses local knowledge to produce a new fitness score that can be used by the global genetic algorithm to evaluate the individual's ability to be improved.

The Baldwin effect is somewhat Lamarckian in its results although it uses different mechanisms (Turney 1996). It explains interactions between learning and evolution by paying attention to balances between benefit and cost of learning. The Baldwin effect consists of the following two steps (Turney et al. 1996). In the first step, learning gives individuals the chance to change their phenotypes to improve their fitness. Individuals, who found learning useful and help their fitness to improve, will spread in the next population. In the second step, if the environment is sufficiently stable, the cost associated with learning results in selection favouring individuals that have the traits, which are acquired by others through learning, already coded into their genotype. Through this mechanism, called genetic assimilation, learning can accelerate the genetic acquisition of learned traits indirectly. A critical precondition for genetic assimilation appears to be a strong correlation between genotype and phenotype space so that nearness in the phenotype space implies nearness in the genotype space (Mayley 1996). Otherwise, the acquired traits have little chance of eventually becoming encoded in the genome via chance through genetic operations.

Hinton and Nolan (1987) illustrated how the Baldwin effect can transform the fitness landscape of a difficult optimisation problem into a less difficult one, and how the genetic search is attracted toward the solution found by learning. Gruau and Whitley (1993) showed how local search can change the landscape of fitness function into flat landscapes around the basin of attraction. This change in fitness landscape is known as the smoothing effect. They demonstrated the impact of the smoothing effect on the search process. This learning strategy could be more effective but slower than Lamarckian, since it does not disrupt schema processing of genetic algorithms (Whitley et al. 1994). Baldwinian search can also have the effect of obscuring genetic differences and, thus, hindering the evolution process (Mayley 1996). This is known as the hindering effect. Essentially this occurs as a result of different genotypes mapping to the same or similar phenotypes (as a result of the smoothing effect) with equivalent fitness scores being produced. The genotypes cannot be

effectively discriminated according to their fitness values without considering the learning cost and the evolution of effective solutions is hindered. The Baldwinian effect can aggravate the problem of multiple genotype to phenotype mappings (Houck et al. 1997) (Julstrom 1999). This problem can also waste the resources of hybrids that use clustering techniques in the genotype domain to reduce unnecessary local search, in contrast to the Lamarckian approach which has been shown to help alleviate this problem (Joines and Kay 2002).

Hart et al. (1995) pointed to the importance of considering the cost of learning, which has been ignored by most researchers when studying the impact of the Baldwinian strategy on the hybrid search by analysing its performance based on the number of generations of the genetic algorithm only. Learning can introduce a computational cost which overweighs its benefits in search.

2.4.3 Hybrid Lamarckian-Baldwinian models

Hybrid Lamarckian-Baldwinian models are created with a view towards combing the advantages of both forms of learning models (Orvosh and Davis 1993). The combination of the Baldwinian and the Lamarckian approaches can be done at two different levels. Hybridisation can be used at the individual-level, where some individuals evolve using the Lamarckian approach while the other individuals evolve using the Baldwinian approach (Houck et al. 1997) (Joines et al. 2000b). Houck et al. (1997) found that this form of partial Lamarckian approach outperformed both the pure Lamarckian and the pure Baldwinian approaches on a selected set of test problems.

The other level is the gene-level, where a number of genes evolve using the Lamarckian strategy and the remaining genes evolve using the Baldwinian approach (Sung-Soon and Byung-Ro 2005). This approach was used to solve the sorting network problem. It can reduce the problem search space and help to produce an efficient search (Sung-Soon and Byung-Ro 2005).

The effectiveness of adopting the pure Lamarckian approach, the pure Baldwinian approach, or any mixture of them in a hybrid is affected by the fitness landscape, the representations, the percentage of population performs local search and local search method used (Michalewicz and Nazhiyath 1995) (Turney 1996) (Houck et al. 1997) (Joines et al 2000b) (Ishibuchi et al. 2003)

2.5 Balance between local and global Search

The hybrid algorithm should strike a balance between the two contrasting objectives, which are exploration and exploitation, in order to be able to solve global optimisation problems. According to the hybrid theory (Goldberg and Voessner 1999), solving an optimisation problem and reaching a solution of desired quality can be attained in one of two ways. Either the global search method alone reaches the solution or the global searcher guides the search to the basin of attraction from where the local search method can continue to lead to the desired solution. In the genetic-local hybrid, the main role of the genetic algorithm is to explore the search space in order to isolate the most promising regions of the search space or hitting the global optimum. However, the main role of the local search method is to exploit the information gathered by the global genetic algorithm. The division of the hybrid's time between the two methods influences the efficiency and the effectiveness of the search process. The optimal division of algorithm's time is an important issue that faces the designers of hybrid genetic algorithms.

Although the aim of combining a global genetic algorithm and a local search method is to reap the best out of the exploring ability of the former, and the efficiency of the latter in reaching local optima, the two methods can interact in a more complicated way than the one described above. Rosin et al. (1997) argued that the mutation operator in a hybrid plays a different role than it does in a pure genetic algorithm. The local refinement requirement of the mutation operator becomes unnecessary in the existence of an explicit local search method allowing the mutation operator to take a more exploratory role. Land (1998) suggested using larger mutations, at least large enough to move from one basin to another, in cases where each individual of the population is completely locally optimised. He went further, when he argued that local search obviates the need for crossover in solving the graph bisection problem, because local search is able to build the very same building blocks that the crossover would otherwise combine.

The exploring ability of the genetic algorithm can be further improved by utilising local search to ensure fair representation of different regions of a search. This can improve the ability of the genetic algorithm to direct the search to the most promising regions of the search space. Once the algorithm has guided the search to the basin of attraction of the global optimum, utilising local search can further improve the search to produce an effective optimisation algorithm. The first goal of the hybridisation, which is the effectiveness of search, can be satisfied if a genetic algorithm and a local search method cooperate in the manner mentioned above. However, there are other forms of interaction,

destructive forms of interactions. For example, the mutation and crossover operators can disrupt good and complete local solutions which may waste algorithm resources and produce an inefficient search. The Lamarckian local search can disrupt the schema processing of the genetic algorithm which may lead to premature convergence and produce an ineffective search.

In addition to the role of genetic operators of systemically exploring the search space, they perform some form of local search with relative low cost compared to the more accurate local search methods. The improper use of the expensive local search in a hybrid can waste algorithm resources. The algorithm should be able to decide wisely on both methods, especially when both can achieve the desired task, taking into account the benefits and costs of their utilisation. The condition of an appropriate use of both methods in addition to the condition of interacting in a cooperative way should be satisfied in order to produce an effective and efficient search algorithm.

Researchers have proposed different techniques to enable the hybrid to mix both methods wisely or at least to reduce the consequences of the improper use of the expensive local search. These techniques are based on modifying the different parameters of a local search method within a hybrid. Modifying the parameters of the local search, such as the frequency of local search, the duration of local search, and the probability of local search can help the hybrid to strike a balance between the two search methods.

2.5.1 Frequency of local search

The number of continuous uninterrupted generations that a genetic algorithm performs before applying local search is usually referred to as the frequency of local search. In the traditional hybrid genetic algorithm, the frequency of local search is one for example. The staged hybrid genetic algorithm (Mathias and Whitley 1992) (Mathias et al. 1994) was designed to separate the two search methods into two distinct stages by increasing the frequency of the local search in order to minimise the interference between the two search methods. Mathias and Whitely (1992) used a local search frequency of two to solve the TSP. However, in a hybrid algorithm to solve the static correction problem (Mathias et al. 1994), the genetic search algorithm was allowed to continue uninterrupted for ten generations before applying a single iteration of waveform steepest ascent iteration to each individual in the population. This hybrid algorithm produced solutions with improved quality of 5% and additional savings in time compared with the traditional hybrid genetic algorithm. Espinoza et al. (2001) conducted a set of experiments to find the optimal local

search frequency of two two-dimensional continuous test functions and they found that the optimal frequency of local search for these test functions was 3.

The optimal frequency of local search is function dependent and varies with time because the optimal time that should be spent on local and global search algorithm depends on the distribution of individuals in the population. Syrjakow and Szczerbicka (1995) studied the optimal switch point between the genetic algorithm and local search to fine-tune the solution found by the pre-optimiser genetic algorithm. They studied three criteria: the number of function evaluations, the convergence speed of the genetic algorithm, and the regional accumulation of search points indicating the convergence toward a specific region in the search space so as to determine the optimal switch point. The convergence speed criterion produced the highest efficiency in their experiment. Lobo and Goldberg (1997) address the problem of deciding between global search and local search in order to make the most out of either technique. They tried to answer the question; when should the local search be used and when should the global genetic algorithm be used to achieve the maximum possible efficiency? They viewed the problem as a two armed bandit problem where the payoff of each bandit is unknown and changes with time. They presented a model for efficient hybridising based on the concept of probability matching. This model can be viewed as an adaptive technique that adjusts the frequency of local search depending on the efficiency of both genetic and local techniques as the search progresses. Tuson and Ross (1996) used a similar model to adapt the operator probability in their Cost Based Operator Rate adaptation. They used their model to select the use of a mutation or crossover operation in a pure genetic algorithm. The same technique has been used to solve the 3MP (Magyar et al. 2000), where an adaptive hybrid algorithm select one operator from eight recombination and local search operators based on their current and past benefit-cost ratio.

Espinoza et al. (2001) used the change in coefficient of variation of the fitness function to determine whether the genetic algorithm is exploring new regions of the search space or exploiting the already visited regions. Based on that, the algorithm selects to perform either a genetic or a local iteration. The algorithm relies on the local search role to improve the sampling of the new regions that are being explored in the case of any increase in that coefficient. Once the search has branched to a local search, the fitness improvement-cost ratio of both the last genetic and the local iterations, and the maximum number of local iterations are used to decide on continuing the local search or going to the global search. The experiments showed that the algorithm is more efficient than a pure genetic algorithm

and is stable against a greater range of parameter settings than the standard staged hybrid genetic algorithm.

Hacker et al. (2002) proposed an approach that switches between global genetic and local search, based on the local topology of the search space. The basic idea of this approach ignores the role of local search in improving the sampling ability of the genetic algorithm. It concentrates on the efficiency of local search, i.e. at finding the optimal once the global genetic algorithm has defined its basin of attraction. The utilisation of the relative homogeneity of the population and regression analysis to determine whether the search is exploring a single basin or multiple basins was investigated. The coefficient of variance of both the fitness and phenotype was used to quantify the relative homogeneity of the population. A decrease in the values of the coefficient of variance indicates that the genetic algorithm has converged to a small area of the search space and the search process can therefore be made more efficient by switching to a local search. Whereas, an increase in its value indicates a new region of the search space is being explored indicating that there is less need to use a local search. Regression analysis has also been used to determine when to switch between global and local techniques. The value of the error of fitting the population of solutions to a second-order surface can indicate as to whether the genetic algorithm is exploring multiple basins or a single basin in the search space. Depending on the value of that error the algorithm decides to switch to a local search or continue the global search. They concluded that utilising local search could be helpful for small size search spaces in the early stages of search due to their role in helping the genetic algorithm to define the most promising regions of the search space. However, for large size and complicated search spaces, their role is limited to accelerating finding of the global optimum once the genetic algorithm isolates the most promising region and can be helpful in later stages of the search.

2.5.2 Duration of local search

Local search duration influences the balance between the global exploration genetic algorithm, and local refinement of neighbourhood search method, in hybrid genetic algorithms (Hart et al. 2000) (Ishibuchi et al. 2003). A hybrid with long local search duration will execute fewer generations of the genetic algorithm than a hybrid with shorter local duration, if both terminate after the same number of function evaluations.

On combinatorial domains, a local search can be performed until a solution converges to a local optimum. However, on continuous domains, the local search is typically truncated before reaching a local optimum when its step length becomes too small. Performing local

search until a solution converges to a local optimum, which is referred to as complete local search, may lead to the loss of population diversity (Whitley et al. 1994) depending on the learning strategy used. Hybrid genetic algorithms that adopt the pure Lamarckian approach are prone to loss of diversity more than others which utilise other learning techniques.

Applying a complete local search on costly function evaluations can also be expensive. However, there is a certain class of problems, decomposable fitness problems (Radcliffe and Surry 1994), where calculating the fitness of a solution given the fitness of its neighbour, is significantly less computationally expensive than computing its fitness from scratch. TSP is an example of this group of problems where computing the length of a tour that shares most of its edges with another tour, whose length is already known, is much cheaper than computing the length of a general tour. Radcliffe and Surry (1994) argued that hybrids are more suitable for problems exhibiting this property.

A few studies have been conducted which investigate the optimal duration of local search. Hart (1994) found that using a short duration of local search produced the best results for the Griewank functions (Griewank 1981), whereas a long duration produced better results for the Rastrigin functions (Torn and Zilinskas 1989). Rosin et al. (1997) experimented with very short and very long local search durations in a hybrid to optimise the drug-docking configuration. Both durations were found to yield similar performance. Hart et al. (2000) concluded that duration of local search is an important factor and hybrid genetic algorithms with long local searches will be most effective for nontrivial problems.

The high cost of complete local search on expensive function evaluations makes any improper use of the local search difficult to recover from. However, the recovering from any misuse of partial local search is still possible. Partial local search is more suitable for hybrids that decide on a global or local approach depending on the current state of search and the previous performance of both methods. In this case, where there is a possibility of misjudgement in some circumstances, the use of partial local search gives the hybrid more chance to recover from such errors than using complete local search.

2.5.3 Probability of local search and local search selection

In any hybrid algorithm, a local search can be applied to either every individual in the population or only few individuals. In traditional hybrid genetic algorithms, a local search is applied to every individual in the population. However, applying a local search to every individual in the population on costly function evaluations can waste resources without providing any more useful information. In this case, the local search can be applied

to individuals that fall in the same basin of attraction of the search space, whereby producing the same local optimum. Applying a local search to a large fraction of the population can limit exploration of the search space by allowing the genetic algorithm to evolve for a small number of generations. The possibility of applying local search on more than one individual from the same basin can be reduced by performing local search on only a small fraction of the population. This also lowers the chances of applying an unnecessary local search on individuals that fall in non-promising regions of the search space. Deciding upon the optimal fraction of population which should perform local search, and the basis on which these individuals are chosen, has a great impact on the performance of a hybrid.

Hart (1994) investigated the impact of the fraction of population that undergo local search on the performance of real-coded genetic algorithm. He found that a relation exists between this fraction, the population size and the performance of the hybrid. He also found that performing local search on small fraction could be more efficient when using larger populations and those large fractions can help to reflect the search space characteristics when using small populations. He concluded that a more selective use of local search could improve the efficiency of hybrids. Hart and Belew (1996) studied the impact of local search probability on the efficiency of hybrids. Their studies indicate that the probability of local search should be kept low in the initial stages and incremented in later generations. The population diversity in the initial stages of genetic algorithm enables good sampling of the search space. However, as the diversity diminishes in the later stages, the sampling ability of the genetic algorithm requires additional help from the local search.

Different techniques, such as tuning, distribution-based (Hart 1994), fitness-based (Hart 1994) techniques and local search potential (Land 1998), have been proposed to decide on the optimal fraction of population that should perform a local search. These techniques aim to reduce unnecessary local searches. However, they differ in the way they select individuals that perform the local search.

2.5.3.1 Tuning technique

In the tuning technique, a primary experiment is conducted in order to find the optimal fraction of the population that should perform local search. This fraction is usually referred to as the probability of local search. This value is then used to run the real experiment and remains fixed during the run. Typically, the individuals that undergo local search are chosen uniformly at random. Rosin et al. (1997) apply local search to 7% of the population in each generation in their hybrid to solve the docking problem. In Land et al. (1997), only 5% of randomly selected individuals of the population perform a Marquardt-

Levenberg local search in their hybrid to determine the basic parameters that describe the structure of a semiconductor wafer. Hart et al. (2000) and Morris et al. (1998) apply local search to 6% of the population. Espinoza et al. (2001) found applying local search on 10% of the population produces the best efficiency for both their adaptive hybrid algorithm and the standard staged hybrid algorithm. In their adaptive hybrid genetic algorithm, this value is used as an initial value for the probability of local search, which is reduced by a specific value after applying local search. In a hybrid to solve the TSP, Krasnogor and Smith (2000) applied their adaptive local search method with a probability of 1.0 to each individual in the population, except the one with the best fitness.

2.5.3.2 Distribution-based technique

Distribution-based techniques modify the probability of local search based on the distribution of individuals in the population. The motivation for these techniques is to ensure that only one individual from each basin of attraction in the search space can undergo local search. These techniques can improve the sampling ability of the hybrid by preventing bad representatives of good regions from misguide the global genetic algorithm.

Hart (1994) used F statistic as a measure of distance over the space of genotypes to adapt the probability of local search. Joines and Kay (2002) combined evolutionary algorithms with random linkage and borrowed the concept of short memory from tabu search to avoid performing unnecessary local search on non-promising regions of the search space. The authors defined tabu hyperspheres around the offspring of the genetic algorithm to reduce the amount of wasted function evaluations owing to the rediscovery of the same local optimum. The probability of local search of each offspring depends on the distance to the nearest tabu region. By decreasing the size of these tabu hyperpheres as the search progress, the algorithm can intensively search the most promising regions of the search space. This, in turn, can help to find the exact local optimum of the region which also represents the global optimum of the search space. The authors compared their hybrid using the Lamarckian leaning approach with a pure genetic algorithm, and the standard hybrid genetic algorithm where each offspring perform local search using two different learning strategies. They reported that their hybrid outperformed other algorithms in terms of both solution quality and computation effort. Martinez-Estudillo et al. (2004) selected individuals for local search using clustering techniques to optimise the structure and the weights of product-unit based neural networks. The results showed that the clustering approach was able to perform better than similar algorithms that do not use clustering analysis.

2.5.3.3 Fitness-based technique

A fitness-based technique adaptively calculates probability with which local search is applied. This technique uses the fitness information in the population to bias the local search toward individuals that have better fitness. The local search probability of each individual is modified based on the relationship of its fitness to the fitness of other individuals. These methods assume that individuals with better fitness are more likely to be in the basins of attraction of the most promising regions. This assumption ignores the dynamic of genetic algorithms and the cumulative effect of applying local search on successive generations which can aggravate the sampling ability of the global genetic algorithm and can misguide the search. For example, if a promising region of the search space is represented badly by an individual with under average fitness and, in the same population, a non-promising region is represented by individual with over average fitness, the representative of the non-promising region will have more chance to perform local search and improve its chances of survive.

Hart (1994) found no statistical differences between the results obtained by applying fitness-based selection and the results of fixed probability of local search. Espinoza et al. (2003b) used a clustering technique that is tailored to the three different stages the authors have defined for constrained problems to adapt the probability of local search. In the first stage, where all the solutions are infeasible, and last stage, where all the solutions are feasible, the authors experimented with clustering the individuals depending on their fitness. The selection was performed by means of Latin-Hypercube sampling from clusters which had formed. In the second stage where a few individuals are feasible, the probability of local search is proportional to the number of feasible solutions in the population. The results showed that the algorithm, which is based on a fitness clustering technique, is more reliably faster than the adaptive hybrid genetic algorithm with fixed starting local search probability. Lozano et al. (2004) proposed a simple adaptive scheme which sets the probability of local search of each individual to either 1.0 or 0.0625 depending on the individuals fitness compared to the fitness of the current worst individual in the population. The authors concluded that this adaptation mechanism allows the balance between the global genetic search and the local search to be adjusted according to the particularities of the search space, thus allowing significant improve in the performance for problems with different difficulties.

2.5.3.4 Local search potential technique

The local search potential selection (LS potential) mechanism has been proposed by Land (1998) to decide which individuals should perform a local search. Land suggested

that biasing the local search toward individuals that can be most efficiently improved by local method makes the most effective use of local search. The least easily improved solutions are likely to be those at or near to the local optimum and it is inappropriate to expend effort on fine refinement, as long as there are large differences in the population's fitness. In this way, the scheme biases the hybrid toward more exploration. As the population gets closer to the optima, this mechanism allows local search to progress to the next level of refinement. In his algorithm, he used the past local search effectiveness as a measure to estimate future effectiveness.

Different techniques have been used to control the different parameters of the local search in order for it to strike a balance with the global genetic methods. Most of the controlling techniques which are described by Eiben et al. (1999) for controlling the parameters of evolutionary algorithm have been investigated and applied to control parameters of local search methods in a hybrid. Although, self-adaptation mechanisms have been successfully used to adapt different parameters of evolutionary algorithms, for details on this subject the reader can refer to Eiben et al. (1999). To the author's best knowledge, it has not been applied in order to achieve a balance between local and global searches. The self-adaptation techniques are reported to be successfully used to decide between different local search methods in solving the OneMax problem, NK-Landscapes, and TSP (Krasnogor and Simth 2001).

This chapter aimed to shed some light on the effectiveness and efficiency of hybridising genetic algorithms with various techniques. In order to fulfil this aim different search techniques have been reviewed in addition to some of the wide variety of hybrid genetic approaches. These approaches show that hybridising is one possible way to build a competent genetic algorithm (Goldberg 1999) that solves hard problems quickly, reliably and accurately without the need for any forms of human intervention. Hybridisation has been utilised to construct competent genetic algorithms that belong to two of the three main approaches for building competent genetic algorithms, which are perturbation, linkage adaptation and probabilistic model building techniques (Chen and Goldberg 2005). The collective learning genetic algorithm is an example of a competent genetic algorithm that employs specifically designed representation and operators for adapting genetic linkage along with the evolutionary process. Other search and optimisation methods can also be used to adapt genetic linkage. Probabilistic Model-Building Genetic Algorithms (PMBGA) are examples of probabilistic model builders which learn genetic linkage via building models based on the current population.

Hybridisation is also one of the four main techniques for efficiency enhancement of genetic algorithms. Hybridisation can also be used as a tool to achieve evaluation relaxation, which in turn is another main technique for efficiency enhancement.

The ability of a genetic-local hybrid to solve hard problems quickly depends on the way of utilising local search information and the mechanism of balancing genetic and local search. By reviewing the different hybrid approaches, some of the important factors that affect the hybrid performance were presented. This review shows that there is a trend to adapt some of the hybrid design choices through adapting the control parameters associated with these choices while the search is progressing. Different adaptation techniques have been used to adapt the selection of a local search method among the available methods, the selection of individuals for a local search and other design aspects.

Chapter 3 Extent of local search

Local search algorithms usually take a considerable number of function evaluations before reaching a local optimum (see section 2.2). When combining a local search algorithm within a genetic algorithm, the duration of local search (expressed as a number of local search steps between two genetic algorithm generations) significantly affects the hybrid performance, due to its influence on the balance between exploration and exploitation (Hart et al. 2000). The optimal duration of local search depends on the problem to be solved (Hart 1994) (Radcliffe and Surry 1994) (Hart et al. 2000). For this reason, the duration of local search is a design choice faced by the hybrid practitioners while solving real-world problems. The hybrid practitioners can decide either to perform a complete local search (Rosin et al. 1997) (also referred to as exhaustive search), where local search steps are performed until a local optimum is reached, or to go for a partial local search, where a specific number of local iterations are performed before returning to the global genetic algorithm.

In this chapter, the influence of the complete and the partial local search on the performance of hybrid genetic algorithms is investigated. Their interactions with the learning strategy and their combined effect on the optimisation process are studied. However, before describing the methodology that has been followed in this investigation, the effects of the duration of local search on its role in a hybrid are analysed. The analysis helps to provide insight into the expected behaviour of a hybrid, depending on the duration of its local search. The discussion of expected behaviour is followed by results of experiments that have been conducted to support such behaviour.

3.1 Duration of local search and hybrid performance

The duration of local search has a great impact on the hybrid's performance. Through controlling the duration of the local search, the algorithm can strike a balance between the local search algorithm and the global genetic algorithm. The duration can affect the ability of a hybrid to explore the search space, to recover from sampling errors and to combat the consequences of the hindering effect.

3.1.1 The exploration ability

The duration of local search influences the exploring power of the global genetic algorithm. The global genetic algorithm, as an exploring tool, and the local search algorithm, as an exploitation tool, share a common budget of hybrid's resources. The heavy

use of these resources by any of these tools reduces the efficiency of the other. Excluding the decomposable fitness problems, the local search algorithm usually requires a heavy use of the hybrid's time when compared to the genetic algorithm which requires one function evaluation per solution. Heavy use of the algorithm's time by the local search algorithm can reduce the time budget specified for exploration by the global genetic algorithm.

Performing a complete local search, which usually requires a considerable number of function evaluations, in a hybrid can hinder its exploring abilities. It can waste algorithm resources without providing new information. It may also consume the algorithm's time by re-sampling already visited points instead of exploring new areas.

Employing a partial local search, on the other hand, usually consumes fewer function evaluations. This can reduce the possibility of wasting the algorithm's resources and gives the global genetic algorithm more chances to explore the search space effectively.

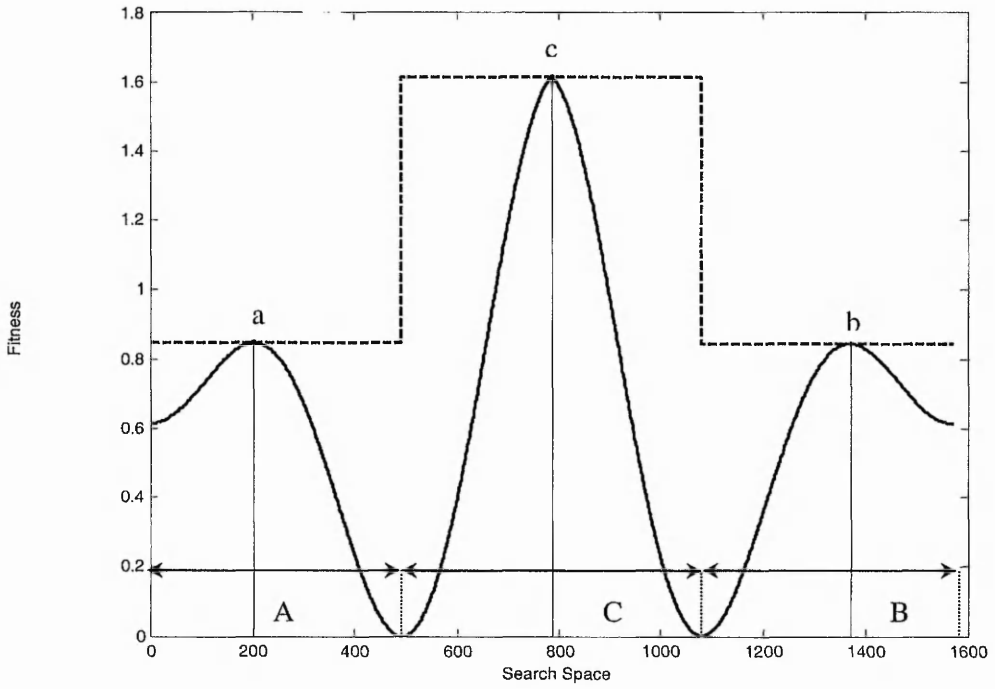

Figure 3.1: The Combined Effect of the Pure Lamarckian Learning Strategy and the Complete Local Search on Problem Search Space.

In addition to the above mentioned effect of the duration of local search on the hybrid's exploring capability, the exploring ability can be affected in another form involving the population diversity. This effect is usually associated with the pure Lamarckian learning

strategy. It is easily perceived in embedded hybrid algorithms where a local search is performed by every individual of each generation. In such algorithms, the population diversity is significantly influenced by the duration of local search.

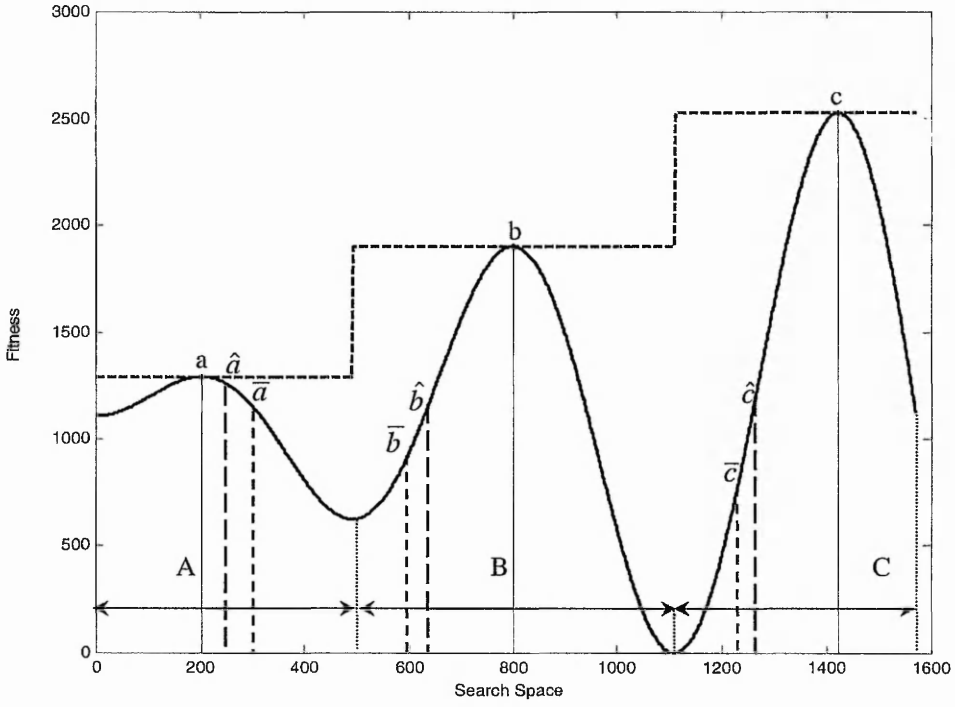

Figure 3.2: The Combined Effect of the Pure Lamarckian Approach and the Duration of Local Search.

Through the complete local search, the whole population is mapped to the local optima of the optimisation problem. For example, in the fitness landscape shown in figure 3.1, incorporating a complete local search maps the points from regions A, B and C to the local optima a, b and c, respectively. This can accelerate locating the global optimum c, once the hybrid guides the search to its basin of attraction (region C). However, if the algorithm fails to sample a point in the most promising region, C, in the initial stages of the search, the process of steering the search into the direction of the global optimum, c, can face some difficulties using a population of local optima only (points a and b in this example). Owing to a lack of population diversity, the possibilities of generating offspring in region C and as a result locating the global optimum, c, are significantly reduced. These possibilities are further decreased if the basins of attraction of local optima are clustered together in the search space away from the global optimum (figure 3.2). For example, if the population

consists of samples of regions A and B, a complete local search will transform them to points a and b. The individuals of the next generation will appear more likely around these two local optima (in regions A and B) and a complete local search will map them again to the same points. This iterative cycle can lead the search to a local instead of global optimum.

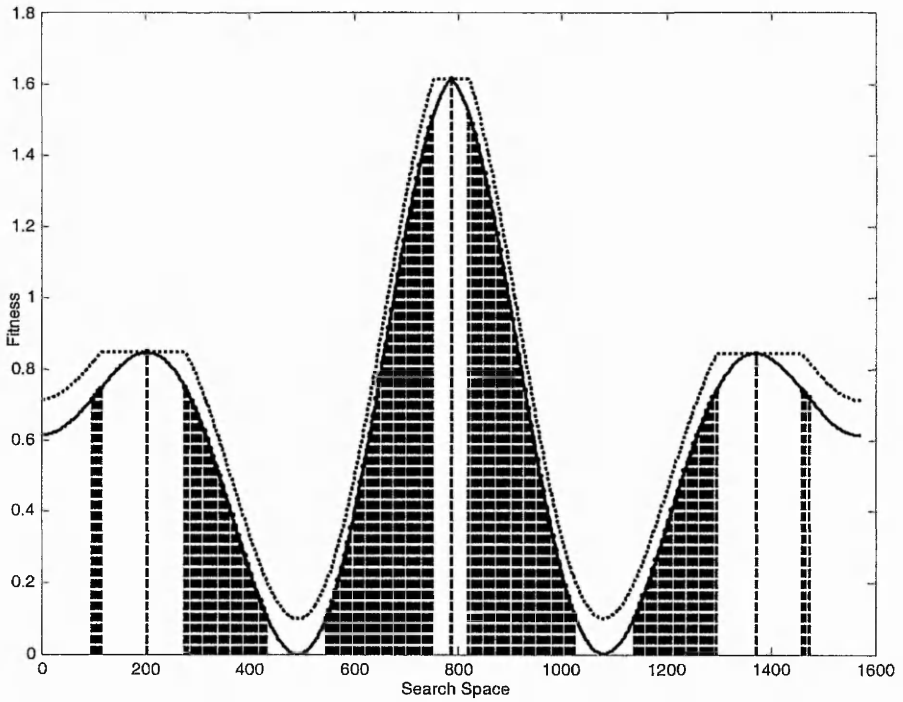

Figure 3.3: The Combined Effect of the Pure Lamarckian Strategy and the Partial Local Search on Problem Search Space.

On the other hand, the possibility of exploring limitations due to diversity loss is reduced by involving a partial local search. The partial local search maps points in the same basin of attraction to new positions in the basin. Figure 3.3 shows the effect of a partial local search on the search space of global genetic algorithms and the fitness landscape. The problem search space is mapped to the shaded parts and the fitness value is mapped to the dotted curve shown in figure 3.3. The diversity loss is limited compared with that of the complete local search (figure 3.1). By comparing the effect of using a partial local search with small durations on the search space, as depicted in figure 3.3, and that of a complete local search as shown in figure 3.1, it is clear that population diversity suffers in proportion to the local search duration. In figure 3.1, where the duration of local search is at its maximum value,

the complete search space is mapped to three local optima $\{a, b, c\}$. In contrast, where the duration is small, the search space is mapped to a subset that consists of about 50% of the original search space (figure 3.3). In the first case, there is a considerable possibility of driving the search towards a local instead of global optimum, whereas in the second case, this possibility is reduced. Consequently, the possibility of leading the search to a point near and not the exact optimum still exists, and the speed of generating such a solution is slow compared to that generated by the complete search.

3.1.2 The ability of recovering from sampling errors

The duration of local search can influence the ability of a hybrid to recover from sampling errors. The global genetic algorithm can sample bad representatives of good regions and as a result lead the search towards a local instead of global optimum. For example, in figure 3.2, the global genetic algorithm can sample points $\overline{a}$, $\overline{b}$ and $\overline{c}$ as representatives of regions A, B and C, respectively, which can misguide the search by directing it towards the non-promising region A and its local optimum. Incorporating a local search can help to recover from such sampling errors depending on its duration.

A complete local search enables fair representation of search areas in view of the fact that each area is represented by its local optimum. In figure 3.2, utilising a complete local search can help to recover from the above mentioned sampling error. Instead of using the fitness values of $\overline{a}$, $\overline{b}$ and $\overline{c}$ as representatives of the regions' fitness, the algorithm uses the fitness values of a, b and c (the regions' local optima) to direct the search towards the global optimum.

Contrary to the complete local search, the partial local search may not improve the sampling ability of the global genetic algorithm. For example, in figure 3.2, the use of a partial local search can map the points $\overline{a}, \overline{b}$ and $\overline{c}$, which are bad representatives of regions A, B and C (figure 3.2), to points $\hat{a}, \hat{b}$ and $\hat{c}$, respectively, and still guide the search in the wrong direction.

3.1.3 The ability to combat the hindering effect

In order to gain some insight into the impact of the duration of local search on the hindering effect associated with the pure Baldwinian learning strategy, the combined effect of this learning strategy and the duration of local search on the fitness landscape needs to be illustrated. Figure 3.4 demonstrates this effect. In this graph, *Ps* represents a partial local search with short durations and *Pl* stands for partial local search with long durations. Due to the smoothing effect of the pure Baldwinian strategy, the local search changes the fitness

landscape function into flat regions around the basin of attraction. The size of the flat landscape depends on the size of the basin of attraction and the duration of the local search. A complete local search, which represents the maximum duration, produces flat landscapes with sizes equal to that of the basins of attraction. On the other hand, a pure genetic algorithm, which represents a local search with minimum duration (zero), produces flat landscapes with sizes equal to that of the local optima. It is clear from the graph that the size of the introduced flat landscapes can be controlled by adjusting the duration of the local search. A local search with small durations (*Ps* in figure 3.4) generates small flat areas, while long durations (complete and *Pl*) produce larger areas.

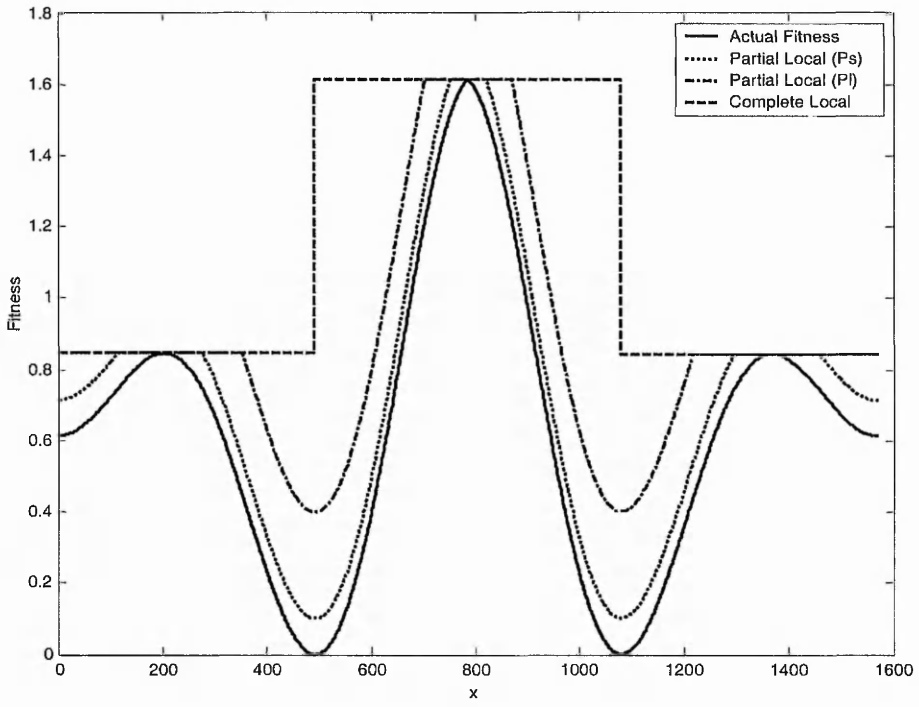

Figure 3.4: The Effect of the Partial and the Complete Local Search on Fitness Landscape.

The effect of combining the pure Baldwinian strategy and a complete local search on the fitness landscape increase the possibility of exposing the search to the drift stall. This effect is common in embedded hybrid algorithms. In early stages of the search performed by these algorithms, employing a complete local search can accelerate discrimination against solutions that are in non-promising regions of search space. In figures 3.1 and 3.2, the algorithm can easily favour samples from region C against samples from regions A and B

based on the fitness value of the regions' local optima. However, when the population converges at the most promising region of search space, region C, the use of the pure Baldwinian strategy and the individuals' acquired fitnesses (global optimum fitnesses) are not enough to guide the search to the global optimum. This demonstrates that the use of fitness alone cannot help to favour solutions with innate fitness against others with learned or acquired fitness. The absence of selection pressure, due to the hindering effect, can lead the search toward any point in the global optimum basin of attraction instead of the global optimum itself. The hindering effect increases the possibility of driving the search to the drift stall.

The utilisation of a partial local search with the pure Baldwinian learning strategy may not help in discrimination against samples from non-promising regions in the early stages of optimisation. For this reason, hybrids may take longer periods of time before converging at the most promising region of the search. The partial local search, however, can help to alleviate the hindering effect once the algorithm guides the search to the most promising area. The use of a partial local search can limit the search to only a small area around the global optimum depending on its duration and not the whole basin of attraction as in the case when using a complete search. The utilisation of a partial local search with small durations can lead the algorithm to converge at a solution very near the global optimum.

In addition to the main effects mentioned before, the duration of local search can affect the adaptation ability of the adaptive hybrids and the prediction of a suitable local step size for the local search. The low cost associated with a partial local search makes it suitable for adaptive hybrids, which decide on a global or local approach depending on the current state of search and the previous performance of both methods. In this case, the use of a partial local search gives the hybrid a higher chance to recover from misjudgement errors than using a complete local search.

In contrast to a complete local search, where a local search algorithm makes use of the information gathered during the previous local iterations to predict the next optimal local step length, the discontinuity of a partial search makes such prediction difficult.

3.2 Experiments

A set of experiments has been conducted to investigate the impact of the duration of the local search on the search process and hybrids performance. The aim of these experiments was to gain insight into how local optima were sampled by the global genetic algorithm throughout the search process. They also aimed to improve the understanding of

the effect of the duration of local search on the sampling process and its consequences on the hybrid's performance.

The first test function (Goldberg and Vosser 1999) (Espinoza et al. 2001) used in these experiments was:

$$f(x,y)=\begin{cases} h_i-\frac{h_i}{r_i^2}(\bar{r}^2)(2-\frac{\bar{r}^2}{r_i^2}) & for \quad \bar{r}^2 \le r_i^2 \\ 0 & otherwise \end{cases} \tag{3.1}$$

where $\bar{x}=x-cx_i$, $\bar{y}=y-cy_i$, $\bar{r}^2=\bar{x}^2+\bar{y}^2$,and c_i={(2.0, 8.0), (3.0, 4.0), (5.0, 7.0), (7.0, 8.5), (7.0 ,4.0)}, r_i={1.0, 1.0, 1.0, 1.0, 1.0} and h_i ={1.0, 2.0, 3.0, 4.0, 5.0}. The global maximum is 5.0 and is located at $(7.0,4.0)$. Figure 3.5 shows the fitness landscape of this function. This function, which will be referred to as F1, has conical basins of attraction. The radiuses of these basins were set to the same value to eliminate the influence of the radius on the sampling process.

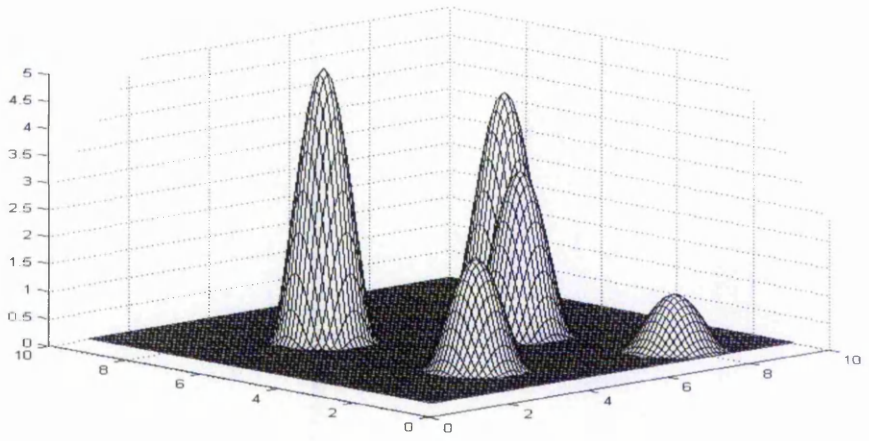

Figure 3.5: The Landscape of the Test Function Used.

The first experiment, which is referred to as experiment 3.1, was conducted by allowing an embedded hybrid algorithm, where every individual in the population performs a local search iteration (local search probability of 1.0) at each generation (local search frequency of 1), to run for a specific period of time. The number of times that each local optimum was sampled was also recorded. The frequency at which the global genetic algorithm visited each local optimum in the search space was counted. The counting was carried out during the evaluation process before selecting the mating pool. This enables the study of the

influence of the duration of local search and the learning strategy on the sampling process of the global genetic algorithm.

The interactions between duration of local search and the learning strategy were studied through conducting a set of experiments using the pure Lamarckian, 50% partial Lamarckian and the pure Baldwinian learning strategies.

The partial and the complete local search were simulated in these experiments. In the complete simulated local search, the algorithm mapped points in the basin of attraction to the local optimum in a single iteration and was assumed to consume x function evaluations. However, the partial local search transferred points to new positions in the basin of attraction depending on their location in the basin. Points that were in the upper third of the basin were transferred to the local optimum. Points that fell below that would be shifted up by $h_i/3$ in a single iteration. The cost of each transformation was assumed to be $x/3$ function evaluations.

The algorithms were intended to maximise the fitness of the test function and therefore were allowed to run 100 times. The results were compared with that of the pure genetic algorithm as a baseline to qualify the improvement in the sampling ability.

The hybrids and the pure genetic algorithm were elitist with binary tournament selection, two-point crossover. For all experiments, the crossover rate was 0.7 and the probability of mutation was 0.05. Each variable was represented by a 16-bit gray coded string with a total of 32 bits for each chromosome. The number of individuals of the population was 10. The termination criterion for all the experiments was a maximum number of function evaluations of 20,000. The algorithms sampled the local optimum as if its fitness was in the boundaries of ±0.1 of the local optimum's fitness regardless of whether it was the innate or acquired fitness. The expected sampling frequencies of each of the local optima according to this counting procedure in the case of being sampled randomly are 0.0158%, 0.0198%, 0.0264%, 0.0398% and 0.0806% respectively, starting from the global optimum.

For each algorithm, a histogram was computed to show how often the five local optima were visited during the genetic global search. Each of the bars in the histograms was normalized by dividing by 20,000 (maximum number of function evaluations) so that the plots indicate the proportion of samples occurring at a specific optimum rather than a raw count.

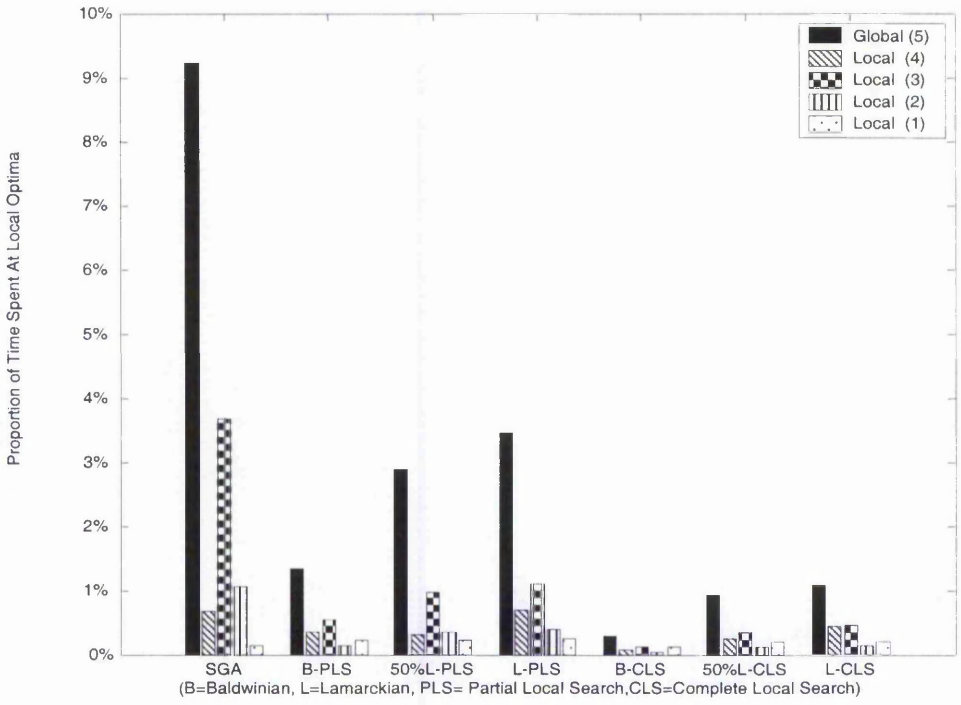

Figure 3.6: Comparing the Sampling Ability of SGA and HGA with a Partial and a Complete Local Search in Experiment 3.1.

The results of experiment 3.1 are shown in figure 3.6. The graph compares the histograms of the proportion of time the global genetic algorithm spent at each local optimum for each method. The histograms illustrate that the algorithms tend to sample some optima more heavily than others. They sampled the high fitness local optima much more frequently than optima with lower fitness. The graph also shows that the pure genetic algorithm sampled the search space in a more biased fashion, towards the fittest local optimum, than both hybrids. The proportion of the fitness landscape for which local search is useless is

$$\frac{(100-\sum_{i=1}^{5}\pi r_i^2)}{100}\times 100 = 84.29\% .$$

The nature of the fitness landscape and the high cost of the local search compared to the global genetic algorithm can explain the decline in the sampling ability of both hybrids. The fact that a considerable number of function evaluations were carried out during the local search process, to improve the sampling ability of the global genetic search, was not counted as a sampling of the search space. This fact contributed to the differences between the pure genetic algorithm and both hybrids.

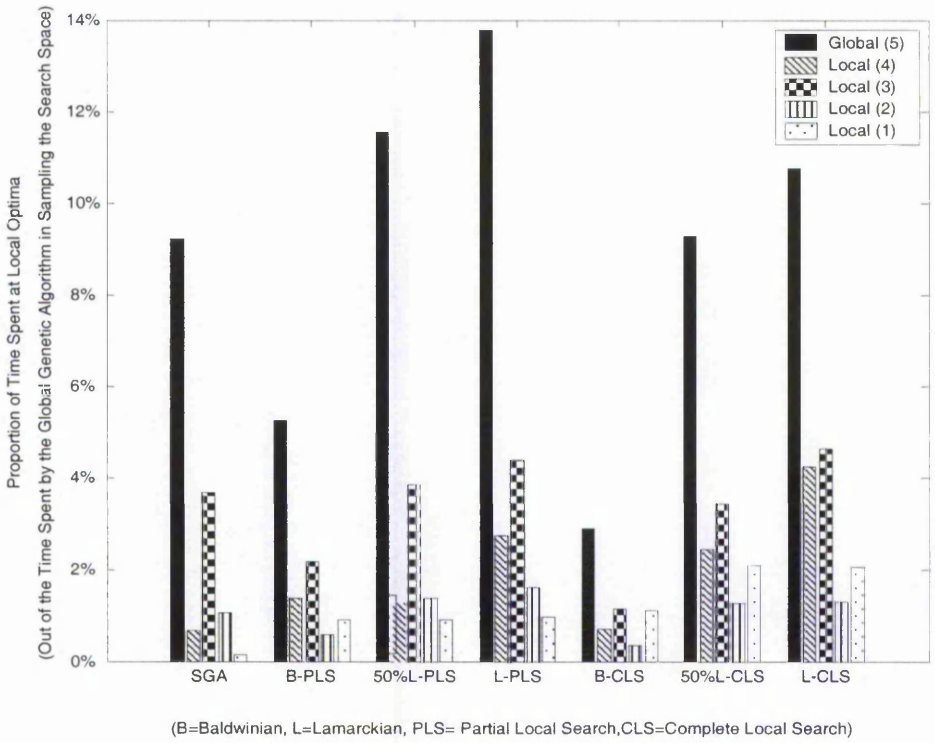

Figure 3.7: The Effect of Duration of Local Search on the Sampling Ability of the Global Genetic Algorithm (Experiment 3.1).

In figure 3.7, the previous histograms of experiment 3.1 are redrawn with the bars normalized by dividing them by the number of points that were sampled by the global genetic algorithm only. The number of points sampled by the global genetic algorithm is equal to the total number of function evaluations, minus the number of function evaluations consumed in the local search. This graph shows how the sampling process can be improved by a local search, if the cost associated with the local search can be ignored. It demonstrates that a local search can considerably improve the sampling ability of the global genetic algorithm for certain classes of functions, where the cost of local search is significantly less computationally expensive than computing its fitness from scratch.

The high cost of the complete local search compared to the partial local search explains the differences in sampling between both local search methods. The histograms in both figures also show that the three algorithms visited the third highest local optimum more frequently than the second highest optimum. This can be explained based on the nature of the landscape. The position of the third highest, in the middle of the other four basins, means it has more chances to be sampled than the second highest which is located on an edge. An experiment has been conducted that confirmed that the basin in the middle has more chances to be visited than the one on an edge. In this experiment, the heights of the third and fourth local optimum have been swapped and the frequency of visiting each optimum by a pure genetic algorithm has been counted. This experiment showed that the optimum at the middle was visited more frequently than the one on the edge regardless of the differences in their heights.

The figure demonstrates that the pure Lamarckian learning strategy, in both hybrids, sampled the local optima more frequently than the other two strategies. In the complete local search, the pure Lamarckian strategy maps both the genetic structure and the fitness value to the local optimum, while the pure Baldwinian maps only the fitness. Through the pure Baldwinian, the individuals improve their chances to survive according to their local optimum's fitness. Their structures, however, are not modified, increasing the risk that offspring of an individual located at an edge will fall outside the basin. In contrast to this, the pure Lamarckian, which maps the genetic structure as well, shifts points to the centre of the basin resulting in an increased chance of their offspring remaining in the basin. Whereas, the pure Lamarckian generates the same effect for the partial local search, where it shifts the points towards the centre of the basin, the pure Baldwinian improves their chances to survive only.

A new experiment has been conducted, which will be referred to as experiment 3.2. The radiuses of all basins except the global basin of the test function have been increased by 50% in experiment 3.2 compared to experiment 3.1. This is designed to make the local optima more reachable to local search algorithms. The proportion of time spent by the global genetic algorithm of each method at each local optimum is shown in figure 3.8. This graph demonstrates that the pure genetic algorithm sampled the local optima more often than both hybrids. It also shows that combining the complete local search with either the pure Baldwinian or 50% Lamarckian led to sampling the second highest local optimum more heavily than the global one. When considering the pure Lamarckian, differences in the portion of time spent on sampling the global optimum and the second highest local optimum are not significant. However, only the combination of the partial local search with

the 50% Lamarckian sampled the second highest local optimum more often than the global one. In general, the sampling ability of hybrids that use a partial local search is better than that of complete local search. The differences between the pure Lamarckian of both hybrids can be used as a base to explain the differences in other learning strategies. Since the use of a complete local search would map points in the boundaries of the basin of attraction to the centre of that basin and that basins of the local optima are clustered on one side relative to the global optimum, hybrids tend to shift the population away from the global optimum. This reduces the possibility of sampling a solution in the basin of attraction of the global optimum. On the other hand, the use of a partial local search, in spite of shifting the population towards the centres, does not shift it to the centre and solutions still exist not far from the basin boundaries. The possibility of producing solutions in the basin of attraction of the global optimum, in this case, is more than that of the complete local search.

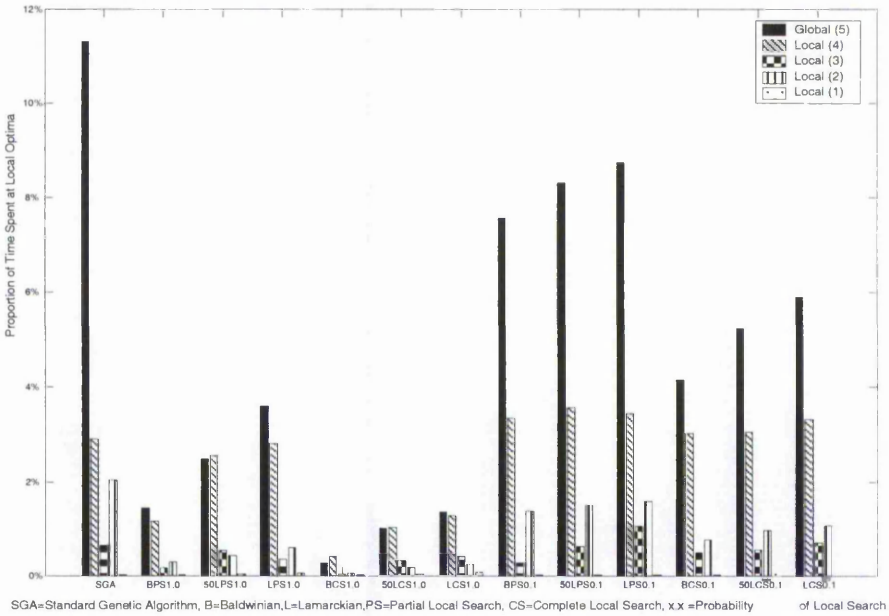

Figure 3.8: The Effect of Local Search Duration on Sampling Local Optima in Experiments 3.2 and 3.3.

Since the use of local search can improve the fitness of a sampled solution in about 31.4% of the search space of the previous test function, the same experiment has been repeated using a probability of local search of less than 0.314. The results of the previous experiment are compared with results of this new experiment, which will be referred to as experiment

3.3, where only 10% of the population was performing a local search in figure 3.8. The graph shows that reducing the probability of local search can improve the sampling ability of both hybrids using different learning strategies. Reducing the probability of local search, in this case, can lead to a reduction in algorithm resource wastage. It is clear from figure 3.8 that hybrids which use partial local search sampled the global optimum more frequently than those using a complete local search and the possibility of misguide the search according to the sampling frequency is less.

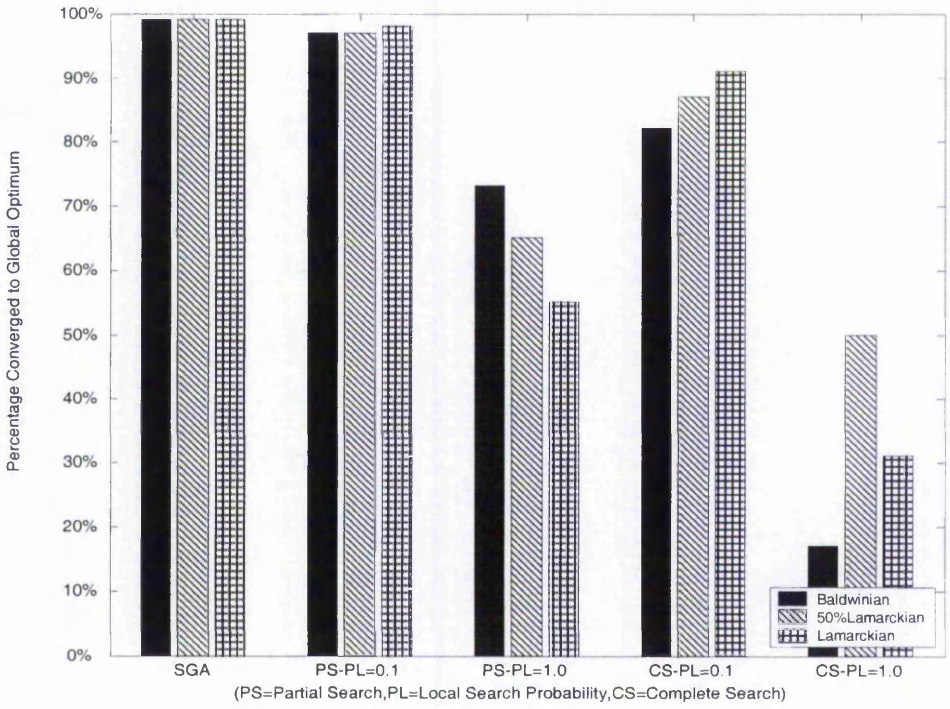

Figure 3.9: The Ability to Find the Global Optimum (Experiments 3.2 and 3.2).

Figure 3.9 compares the percentage of finding the global optimum of each algorithm in the two previous experiments (experiments 3.2 and 3.3). The graph shows that combining the complete local search with the Baldwinian learning strategy and a local search probability of 1.0 generated the worst performance. This combination can aggravate the hindering effect, which makes reliance of the selection operator on the acquired fitness alone insufficient in directing the search to the global optimum. As shown in the figure, performing the complete search on small parts of the population can alleviate this problem by improving the possibility of distinguishing between innate and acquired fitness. When the probability of local search is small, the probability of applying a local search on the

same solution in consecutive local iterations is significantly small, giving the algorithm a better chance to distinguish between innate and acquired fitness. It also provides the global genetic algorithm with more chances to use the algorithm resources to explore the search space effectively. The graph also shows that the pure Lamarckian strategy can find the global optimum more frequently than the other learning strategies when only small fractions of the population are applied to local search. Applying a local search on a small fraction of the population combined with the pure Lamarckian can solve the problem without causing diversity loss since the local search modifies the genetic structure of this fraction only. The use of partial search can alleviate the diversity loss problem further since it maps the points of the same basin to different points in contrast to the complete search, which maps them to the basin's local optimum. The figure demonstrates that the partial local search outperformed the complete local search when adopting the pure Lamarckian approach. The use of a partial local search can alleviate the hindering effect problem associated with the pure Baldwinian by limiting it to a small area of the global basin of attraction. The large difference between the acquired and actual fitness due to the Baldwinian strategy is an obstruction in directing the search towards the global optimum. The complete local search can produce an individual with a high acquired fitness compared to its innate fitness. However, the difference between them is less when utilising a partial local search. Figure 3.10 shows the differences in average innate fitness and the acquired fitness. It illustrates that these differences are proportional to the duration of the local search. These differences can be further reduced using a partial local search with small durations. In this graph the dashed line represents the average fitness after applying a local search (acquired fitness), whereas the solid represents the population average fitness after applying the genetic operations (offspring innate fitness). The population average acquired fitness and the offspring innate average fitness of the pure Lamarckian algorithm are also drawn to visualise the destructive element in the genetic operators on the local search's solutions. This should be taken into account when comparing the innate and the acquired fitness of the pure Baldwinian.

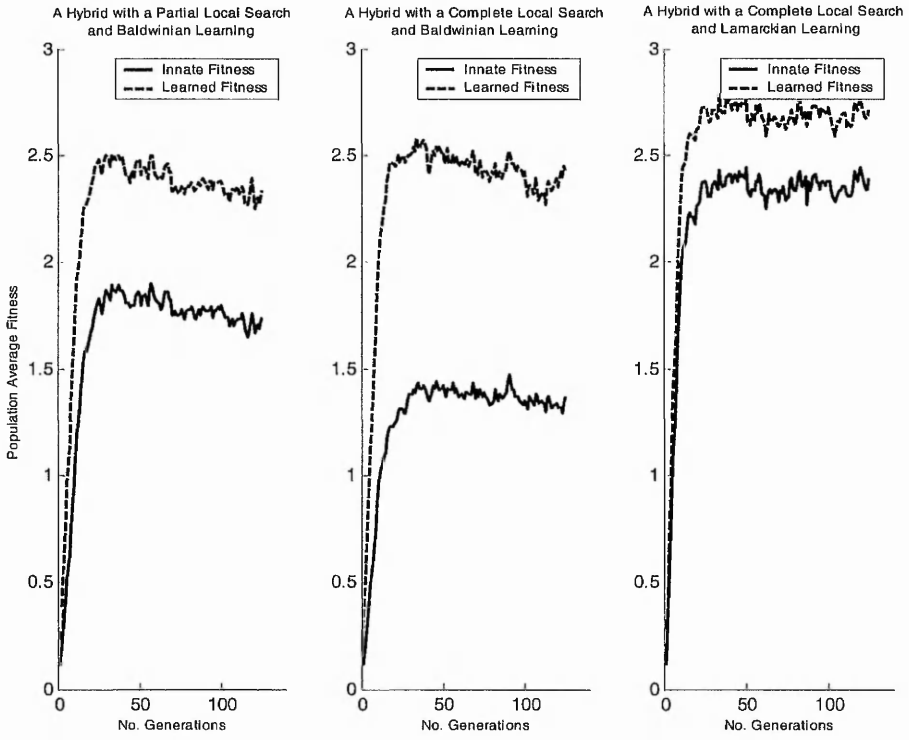

Figure 3.10: The Effect of Local Search Duration on Innate and Acquired Fitness.

Chapter 4 Local search and hybrid performance

Genetic algorithms and local search methods can be viewed as complementary search tools that can be hybridised together to find high quality solutions for an optimisation problem using minimum resources. The ability of genetic algorithms to capture a global view of the search space, when combined carefully with the fast convergence of local search methods (Turney 1996), can often produce an algorithm that outperforms either one alone (Lobo and Goldberg 1997). Hybridising a local search method provides the global genetic search algorithm with some local knowledge that can guide and may accelerate the search to the global optimum (Hart 1994).

The motivation for hybridising a genetic algorithm with a problem-specific method is to enhance the search capabilities. This enhancement can be in terms of effectiveness (Reeves 1994) (Whitely et al. 1994) and/or efficiency (Goldberg and Vosser 1999). The enhancement in the search effectiveness can be expressed as an improvement in the solution quality. Since incorporating a local search introduces an explicit refinement operator (Rosin et al. 1997), the sampling ability of the global genetic algorithm (chapter 3) and the solutions quality can be improved. On the other hand, the efficiency improvement can be expressed as an increase in the convergence speed and/or a reduction in the population size required to attain a solution of desired quality (Espinoza et al. 2003a). The reduction in the population size required reduces the size of the memory needed to process the population in the case of serial genetic algorithms. Hybridisation can be used to speed up a genetic search (Goldberg 2003) through reducing the time needed to reach a global solution. Hybridisation also influences the minimum population size required for a genetic algorithm. Through its effect on the population size, it can increase the convergence speed of parallel genetic algorithms.

The ability of achieving a balance between global and local search is an essential factor that influences the performance of any hybrid. This balance depends on many factors, such as the criteria used to decide between global and local knowledge (Lobo and Goldberg 1997) and the mechanism for striking a balance between the cost and value of local knowledge (Hart 1994).

Trade-off between the cost and the value of local knowledge can be controlled through deciding on the fraction of the population that undergoes a local search and the criteria for choosing its members. The cost of local knowledge can be measured by the number of function evaluations performed by a local search method to gain that knowledge. Its value,

however, can be measured by its effect on improving the convergence speed and/or solution quality. A local search can be applied either to every individual in the population or to only few individuals. However, a more selective use of local search can improve the effectiveness and the efficiency of hybrids (Hart 1994). The different techniques that have been used to decide on the optimal fraction of the population that should perform local search and the basis on which these individuals are chosen were reviewed in chapter 2.

Tuning is the most commonly used technique to decide on the optimal fraction of the population that should perform a local search. This fraction is usually referred to as the probability of local search. Hart (1994) referred to this fraction as the frequency of local search. This term will be used here to refer to the number of the consecutive genetic iterations before performing a local search (Espinoza et al. 2001). The members of this fraction are usually chosen randomly. The probability of local search, in addition to its effect on the solution quality produced and the convergence rate, can affect the minimum population size of the hybrid. The population size, in turn, can affect the convergence speed of the algorithm. The effect on the population size should not be ignored when evaluating the hybrid performance.

A main factor that determines the success of a hybrid is how successfully local knowledge is utilised by the global genetic algorithm (Whitley et al. 1994). The way of using the information gained during local search by the global algorithm is one of the important issues of hybrid genetic algorithms. Either the Lamarckian or the Baldwinian approach can be used. The Baldwin Effect differs from Lamarckian only in the directness of the mechanism by which phenotypic adaptations are converted into genotypic adaptation (Turney 1996). Utilising either form of learning is more effective than the standard genetic algorithm approach without a local improvement procedure (Whitley et al. 1994). The effectiveness of pure Lamarckian, pure Baldwinian or any mixture of them (Orvosh and Davis 1993) is affected by the fitness landscape, the representations, and local search method used (Whitley et al. 1994) (Houck et al. 1997) (Michalewicz and Nazhiyath 1995). For more details on this subject the reader can refer to section 2.5.

The efficiency and effectiveness of any hybrid can be measured by comparing its performance with that of the global genetic algorithm alone. The performance can be measured in terms of convergence speed, the quality of solutions produced and the minimum population size required. Espinoza et al. (2001) have proposed an adaptive hybrid algorithm that can increase the convergence speed to the global optimum. The same authors also showed the effect of a local search method on reducing the population size of the

algorithm compared with the population size of the standard genetic algorithm (Espinoza et al. 2003a).

In this chapter, the effects of the learning strategy and probability of local search on the performance of two hybrids with different mechanisms for deciding between global and local search are investigated. The way that both the learning strategy and the probability of local search interact with each other and their combined effect on the hybrid performance are analysed. The effect of both these factors on the population size requirements, convergence speed, and solution quality has been investigated.

This chapter starts by a literature review of population size requirements for a genetic algorithm. In this review, a brief description of the different facet-wise models (Goldberg 1999) used to estimate the genetic algorithm's minimum population size is given. Then, the relation of computation complexity of a genetic algorithm and the population requirement is reviewed. After the literature review, the effect of incorporating local search on the population size requirement is analysed. This includes the influence of the interactions between the local search probability and the learning strategy on the population size. This chapter ends by discussing the results of the experiments that have been conducted.

4.1 Population size and local search

Deciding on the optimal population size is an important task in the design of genetic algorithms due to its influence on their performance. Through the population size, the diversity can be controlled and an adequate supply of building-blocks can be ensured which is an essential step in designing a successful genetic algorithm (Goldberg et al. 1992). An appropriate population size is even more critical to the success of the Probabilistic Model-Building Genetic Algorithms (PMBGA) (Pelikan et al. 1999a) (Pelikan et al. 1999b) which have become an area of interest recently (Gao 2003).

Increasing the population size of a genetic algorithm improves the quality of its solutions. The greater the population size, the greater the chance that all the building-blocks of the optimal solution are represented in its initial population. However, increasing the population size can slow the speed of convergence since more time may be needed to discriminate the good and bad building-blocks. In the case of limited computational resources, the larger population may preclude convergence at all. Efficient population sizing is critical for getting the most out of a fixed budget of function evaluations.

In the practice of designing efficient genetic algorithms, there has been strong empirical evidence showing that population size is one of the most important parameters that plays a significant role in the performance of the genetic algorithms (Bäck et al. 2000).

4.1.1 Population sizing models

The issue of population sizing has been widely dealt with theoretically. The facet-wise composition approach (Goldberg 1999), which was proposed to obtain insight into genetic algorithms' behaviour, has been used to estimate the population size required. According to this approach, the behaviour of the genetic algorithm can be modelled through combining simple models and blending their effects. Different facet-wise models have been developed to address the genetic algorithm's population sizing issue. The theoretical work done can be categorised into two main groups, namely the population sizing based on initial supply of building-blocks and population sizing based on good decision making between competing building-blocks. Both issues are combined together in the gambler's ruin model (Harik et al. 1999).

The importance of building-blocks and their role in the genetic algorithm's search mechanism have long been recognised (Holland 1975) (Goldberg 1989a). One of the essential steps towards successful design of a genetic algorithm is making sure that the genetic algorithm is well supplied with a sufficient number of the building-blocks required to solve a given problem. The spatial approach (Goldberg et al. 2001), which estimates the population size required to ensure diversity and the existence of sufficient building-blocks in the initial population, can be used to address the building-blocks supply problem (Goldberg 1989b) (Reeves 1993).

Recently, Goldberg et al. (2001) have developed a facet-wise model to estimate the minimum population size that ensures the presence of all building-blocks. This model was used to derive the following formula, which has been experimentally verified.

$$N = \chi^k (k \log \chi + \log m) \tag{4.1}$$

where k is the order of the building-blocks, which represents the minimum number of digits that have physical significance to the solution of the problem, m is the maximum number of building-blocks within a single string, and χ is the alphabet cardinality.

In addition to adequate supply of building-blocks, accurate decision making of selecting the schema belonging to the global optima, when partial solutions are compared with each other, is very important for genetic algorithm success. Goldberg et al. (1992) have developed a model to estimate the population size required to make the correct decision

between the best building-block and its closest competitor in a partition, in the presence of collateral noise coming from other partitions. In their conservative model, they assume that the selection of the correct building-blocks in the first generation is essential and can guarantee the convergence to the global optimum. They used statistical decision making to derive the following population size equation for binary alphabet:

$$N = 2c(\alpha)2^k(m-1)\frac{\sigma_{bb}^2}{d^2} \tag{4.2}$$

where $c(\alpha)$ is the square of the ordinate of a unit normal distribution where the probability equals α, α is the probability of failure, k is the order of the building-blocks, σ_{bb}^2 is the fitness variance of the partition that is being considered, and d is the fitness differences between the best and the second best building-blocks. This equation ignores the external sources of noise.

Harik et al. (1999) extended the population model by integrating the gambler's ruin model and the previous two facet-models to estimate the population size of genetic algorithms. The initial supply of building-blocks and the selection of the best building-blocks over their competitors over a run are combined with the gambler's ruin model. The model views the search process as a propagation of building-blocks through the population, assuming mixing is adequate.

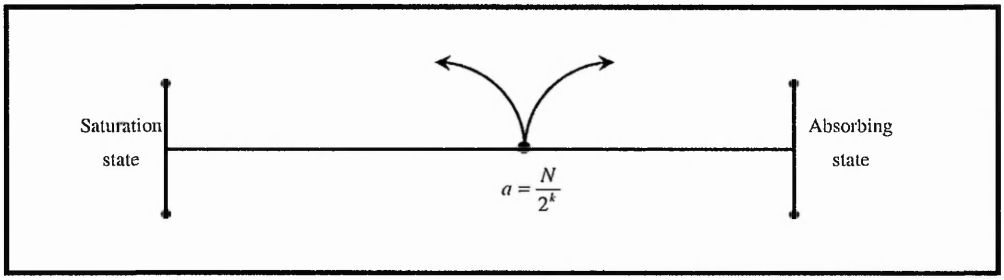

Figure 4.1: The Gambler Starts with a Capital of a Building-Blocks and Ends with either 0 or N Building-blocks.

The gambler's ruin model views the genetic algorithm search, in a single partition, as a series of competitions which progresses until either all the individuals in the population match the building-blocks, or none do. The model is one-dimensional random walk between the absorbing state, corresponding with the loss of the building-blocks, and the saturation state, corresponding with the existence of the building-blocks in all the individuals (figure 4.1). The walk starts from a, the number of building-blocks in the initial population, which is calculated by incorporating the initial supply model:

$$a = \frac{N}{2^k} \tag{4.3}$$

Whereas, the decision making model is incorporated by taking into account the probability of making the right choice between the best building-block and the second best building-block in a single trial:

$$p = \Phi\left(\frac{d}{\sqrt{2(m-1)}\sigma_{bb}}\right) \tag{4.4}$$

where Φ is the cumulative distribution function of the standard normal distribution.

Random-walk literature was utilised to determine the probability that the gambler eventually reaches the saturation state using a population of size *N*. This probability was, then, used to derive the following equation which relates the population size with the required solution quality and several domain-dependent parameters:

$$N = -2^{k-1}\ln(\alpha)\frac{\sigma_{bb}\sqrt{\pi(m-1)}}{d} \tag{4.5}$$

The term $\sqrt{\pi(m-1)}$ represents the noise interference between competing building-blocks. According to Reed et al. (2000), the term $\sigma_{bb}\sqrt{\pi(m-1)}$ can be approximated using the fitness function standard deviation, σ_{ff}. The previous relation can be simplified to the following equation (Espinoza et al. 2003a):

$$N = -2^{k-1}\ln(\alpha)\frac{\sigma_{ff}}{d'} \tag{4.6}$$

where d' represents the signal difference between the best and second best solution. The parameters σ_{ff} and d' can be estimated using a large random initial population.

4.1.2 Computation complexity and population size

The computational complexity of a genetic algorithm can be measured as the number of function evaluations that are required to attain an optimal solution. The number of function evaluations can be calculated by multiplying the population size, *N*, by the number of generations required for convergence, *t*, which is primarily determined by the selection intensity of the selection scheme (Dijk et al. 2004).

The number of generations required is strongly affected by the relative rates at which genes within the population converge. The relative rate of the convergence of building-blocks depends on the problem to be optimised. All the building-blocks of uniformly scaled problems, such as the OneMax problem, converge at a fixed rate. On the other hand, the building-blocks of exponentially scaled problems, such as the BinInt problem (Thierens et al. 1998), converge at variable rates. The convergence time of problems of uniformly scaled

fitness functions, is in the order of $O(\sqrt{l})$ for selection schemes with constant selection intensity and of order $O(l\sqrt{l})$ for proportionate selection (Thierens and Goldberg 1994). Studies (Thierens et al. 1998) (Lobo et al 2000) show that the convergence time of exponential scaled functions is linear with respect to the string length ($t = O(l)$) for constant selection intensity schemes, and exponential $O(2^l)$ for proportionate selection. These relations assume binary representation of the solutions.

The building-blocks of most engineering problems converge at variable rates within the population (Reed et al. 2000). In these problems, the convergence rate of the most important building-blocks is rather fast, while the least important ones only start to converge when the more important ones are almost fully converged. This phenomenon is known as "domino convergence". Thierens et al. (1998) studied the BinInt problem as a prototypical example of the upper boundary case of non-uniformly scaled problems. They derived the following formula for the expected number of generations, *t*, required under domino convergence for all locations to be converged:

$$t_{\text{domino}} = \frac{-\ln 2}{\ln[1-\frac{I}{\sqrt{3}}]} l \tag{4.7}$$

where *I* represents the selection intensity. The expected number of generations for domino convergence depends on the selection intensity and the number of the building-blocks within a string, which is equal to the string length in the case of the BinInt problem (Thierens et al. 1998).

The selection intensity depends on the selection scheme used. The selection intensity is constant in rank-based, truncation and tournament selection schemes and variable in proportionate selection scheme (Goldberg and Sastry 2001).

In the case of tournament selection with a tournament size of two, the selection intensity is $I = (\sqrt{\pi})^{-1}$ (Thierens et al. 1998). The expected number of generation for the entire string to converge is

$$t_{\text{domino}} = 1.76l \approx 2l \tag{4.8}$$

Another phenomenon that is closely related to domino convergence is "genetic drift" (Asoh and Mühlenbein 1994). This phenomenon is the random fluctuation of genes frequencies from generation to generation due to the stochastic sampling errors in a finite-sized population. The random accumulation of the copies of a particular allele in each gene can cause the population to converge to non-optimal value in the absence of selection pressure.

Although the less salient building-blocks of the optimal solution (i.e. the genes with reduced relevance to the solution) experience reduced selection pressure, they may converge to non-optimal values under the crossover and mutation operations.

Asoh and Mühlenbein (1994) showed that the expected time for a gene to converge, as a result of genetic drift, is proportional to the population size. They used random sampling with replacement to derive the following formula:

$$t_{drift} = cN \tag{4.9}$$

where the value of constant c depends on the initial allele proportion. They also showed that the mean drift time, in the case of uniform crossover, is proportional to the population size N and to the logarithm of the number of building-blocks within the string which was equal to the string length in their work:

$$t_{drift} = cN(a \ln m + 1.0)^b \tag{4.10}$$

where the values of constants a and b depend on the initial allele proportion. For a gene with two alleles with initial proportion equal to 0.5, the values of these constants are c=1.4, a=0.5 and b=1.1.

The drift time of a single trial can show a significantly different value from the expected drift time since the variance in the drift process is quite large and increases rapidly with population size (Asoh and Mühlenbein 1994).

In uniformly scaled fitness functions, the genetic drift can make very small populations converge prematurely. This effect is called "drift stall". However, the exponentially scaled fitness functions are more prone to the drift stall. Due to the domino convergence behaviour associated with these functions, the less salient building-blocks of the optimal solution have a high probability of being lost, as a result of the genetic drift, by the time selection and recombination can process them. The only hope to bring these building-blocks back is the mutation operator, which is a slow serial process compared with the rapid implicit parallel processing power of selection and crossover (Thierens et al. 1998). In order to prevent the drift stall effect, the convergence to the optimal solution should be faster than the genetic drift. The following relation needs to be satisfied:

$$t_{convergence} \leq t_{drift} \tag{4.11}$$

For binary represented problems with non-uniformly scaled fitness functions that use binary tournament selection, relation 4.11 can be rewritten in terms of population size and string length, ignoring the effect of number of the building-blocks on the genetic drift, as:

$$N > 1.43l \tag{4.12}$$

This condition sets the lower bound of population size in order to avoid the drift stall.

4.2 Local search and population size requirements

The details of incorporating local search and the learning strategy influence the population size requirements through affecting the population diversity and the fitness diversity. By modifying the population diversity, they affect the genetic drift and the convergence speed. They influence the signal difference between the best and second best solution and the standard deviation of the population fitness through their effect on the fitness diversity. The details of the local search algorithm can affect the population size requirements through influencing the hindering effect linked with the Baldwinian learning strategy.

Genetic drift is one of the issues that need to be considered when deciding on the optimal population size of a genetic algorithm. The population size should be chosen so that it enables the search process to converge to the global optimum before the genetic drift can guide it to the drift stall. Incorporating a local search algorithm within a genetic algorithm influences both the convergence rate and the genetic drift rate. Local search can help to fight genetic drift and protect the population from reaching the drift stall in the early stages of the search. Local search can accelerate the rate of convergence and can be an obstruction in the way of genetic drift by resisting its effect in directing the search towards a local optimum. However, a heavy use of local search can reduce the population diversity and that can lead to the fitness-convergence-state. The population, in this state, is composed of individuals with different genetic structures but with the same fitness. Once the fitness-convergence-state reached, the selection strategy faces difficulties in choosing the mating pool and that subject the search process to genetic drift.

Carefully utilised local search algorithm within a genetic algorithm usually accelerates the population convergence. Well designed hybrids have the ability of converging faster than the pure genetic algorithm in terms of number of generations. An increase in the convergence speed can ensure that the convergence to the global optimum occurs before the genetic drift leads the search to the drift stall.

The improvement in the convergence speed, due to local search, affects the population size requirement of a hybrid. Since the pure genetic algorithm convergences slower than the hybrid genetic algorithms in terms of number of generations and assuming the number of generations needed for a hybrid to converge is t_{hybrid}, then the ratio $t_{\text{hybrid}} / t_{\text{sga}}$ is less than 1.0. The exact value of this ratio depends on the local search details. An intensive use of local search can reduce this ratio. However, it can cause a premature convergence. The way to guarantee the convergence to the global optimum with an improvement in the convergence speed is through the proper use of local search. Managing to improve the convergence rate towards the global optimum reduces the convergence time of a hybrid to less than $2l$ (the convergence rate of non-uniformly fitness functions using the pure genetic algorithm).This modifies the condition imposed on the population size to avoid the stall drift to the following relation:

$$2l \frac{t_{hybrid}}{t_{sga}} < 1.4N \tag{4.13}$$

This relation shows that incorporating a local search reduces the lower bound of population size below the $1.43l$ limit depending on the improvement in the convergence speed introduced by the local search. This enables the hybrid to converge to the global optimum using population sizes of less than that of the pure genetic algorithm.

Through modifying the genetic structure, a local search can fight the genetic drift. It can change the genetic structure of an individual to reflect the genetic structure of the local optimum of the individual's region. This can drift the genetic structure of the population towards the structure of the local optimum of the basin with the biggest area of attraction in the search space. In addition to the selection process, wise use of local search can bias the drift in the direction of the global optimum. Incorporating local search can help to oppose the forces of the genetic drift to drive the population towards a local optimum. It can help to restore some of the building-blocks that may be lost as a result of the genetic drift. It also can introduce some diversity in the population and counterbalance the reduction in the diversity due to the genetic drift.

However, improper use of local search can expose the hybrid's population to the genetic drift in the late stages of the search. The local search can cause the disappearance of the some building-blocks which may not be restored. The disappearance of these building-blocks can produce individuals that have an equal fitness values and different structures which make the convergence of the whole population is governed by the genetic drift.

The state of fitness-convergence can also occur as a result of the hindering effect associated with the Baldwinian strategy. The pure Baldwinian strategy combined with the complete local search can prone the genetic search to the genetic drift. Such utilisation can accelerate reaching the fitness-convergence-state. The hindering effect does not enable the selection strategy to choose between individuals with the same fitness but with different genetic structures. As a result, the selection process degraded to a random process. The population convergence, in this case, is subject only to the genetic drift. By controlling the duration of the local search, the genetic drift can be delayed until an acceptable solution or a solution very near the global optimum is attained (chapter 3). The situation of fitness-convergence can be avoided by adjusting the probability of local search and/or its frequency in order to enable the selection process to distinguish different structures. The decrease in local search probability reduces the probability that a local operation is performed on the same individual on two consecutive iterations and that enables the algorithm to discriminate among the different structures and reduce the hindering effect. The increase in the frequency of local search helps the hybrid to choose depending on innate fitness which reflects the structure and not the acquired fitness.

The influence of the local search method on the population diversity depends on the learning strategy used. However, it impacts the population fitness diversity regardless of the learning strategy. It can reduce the standard deviation of the fitness of the population depending on the local search method used and its duration. The reduction in the standard deviation of the population fitness is proportional to the local search duration (Espinoza et al. 2003a). It can, also, be affected by the probability and the frequency of local search. According to the population size equation 4.6, the decrease in the standard deviation of population can lead to a decrease in the minimum size of population required to optimise a given problem. The local search details affect the population size requirement of a hybrid through affecting the population fitness variance.

Local search can affect the signal difference between the best solution and the second best solution. For the purpose of illustrating the effect on the signal difference, the discussion is restricted to the complete local search (chapter 3). In the early stages of the search, the signal difference between the best solution and the second best solution is equal to the difference between the fitness of the best sampled local optimum with the second best sampled local optimum. This can improve the ability of selection process to direct the search to the most promising region of the search. The samples from the best region have a high probability to win the competition against the second best region even with small population size. However, in the late stages of the search after the convergence at the most

promising region, the value of the signal difference depends on the learning strategy adopted. In the case of the pure Lamarckian approach, the signal difference is equal to the difference between the global optimum and the second best solution. This difference eventually reaches zero when the whole population converges to the global optimum. On the other hand, the hindering effect associated with the pure Baldwinian strategy can reduce this difference to zero when the population converges to the most promising region. According to equation 4.6, an infinite number of individuals is needed in order for a population to converges under these circumstances. By influencing the hindering effect, the probability and the frequency of local search can modify the signal difference in the pure Baldwinian strategy.

The local search probability can also affect the signal difference in both learning strategy depending on whether either the best two solutions or only one of them are selected for a local search. The difference can be increased if the best solution is selected and the second best solution is not and vice versa. If both are selected, the chance of affecting the signal difference is decreased. The probability of selecting either the best two solutions or one of them is affected by the probability of local search. Increasing the probability of local search increases the probability of performing a local search on the best two solutions and vice versa.

4.3 Algorithms and test functions

Two hybrids with different mechanisms for deciding between global and local search were used to gain some insight into the effect of learning strategy and probability of local search on the performance of hybrids. The standard staged hybrid genetic (SSH) algorithm (Mathias and Whitley 1994) and the adaptive staged hybrid genetic (ASH) algorithm (Espinoza et al. 2001) have been tested using two multimodal test functions.

In the standard staged hybrid genetic (SSH) algorithm, the local search step is defined by three basic parameters. These parameters are frequency of local search, probability of local search and number of local iterations. The local search frequency measures how frequently local search is performed. The probability of the local search represents the fraction of individuals in the population that undergo local search at each local search iteration. The number of local search iterations represents the number of local search iterations performed at each local search process. The interference between the local search algorithm and the global genetic algorithm can be reduced through increasing the frequency of the search. By adjusting these parameters, the performance of this type of hybrids can be optimised. The SSH does not apply any selection mechanism for performing local search.

The adaptive staged hybrid genetic (ASH) algorithm uses feedback from the current state of the search process to direct the algorithm to decide between global and local methods (Espinoza et al. 2001). The algorithm works with the same operators as SSH. It performs local search only if new regions of search space are being discovered, and local knowledge can help to guide the search. The probability of the local search is controlled by a deterministic rule that keeps this probability less than a specific value and decreases with each consecutive local iteration. When local search no longer improves the average fitness more than the most recent global search iteration or the maximum number of local iterations has been exceeded, the search returns to the global search. The algorithm focuses on the role of local search in providing the global genetic algorithm with good representatives of new discovered regions. However, the individuals are selected randomly to perform a local search. This algorithm can be criticized for the absence of any selection mechanism to guarantee the use of local search to provide good representatives of the new explored areas.

Two multimodal test functions, with multiple basins of attraction, have been used in the current work. The first function, F1, has conical basins of attraction. Its global maximum is 4 and is located at (7.0, 8.5) (Goldberg and Vosser 1999) (Espinoza et al. 2001). The second function, F2, has elliptical basins of attraction. This function has a global optimum of 4 located at (7.0, 8.5) (Espinoza et al. 2001). Figure 4.2 shows the fitness landscapes of F1 and F2.

The steepest descent method (Press et al. 1993) was used as a local search algorithm. The steepest descent algorithm uses the derivatives of the fitness function to estimate the best step size to climb to the local optimum from the current position in the basin of attraction.

4.4 Experiments and discussion

In order to evaluate the effect of learning strategy and local search probability on the hybrids' performance, a set of experiments was performed. Both hybrids use the simple elitist genetic algorithm with binary tournament selection, single-point crossover, and simple mutation. For all experiments, the probability crossover was 0.4 and the probability of mutation was $1/N$ where N is the population size (Reed et al. 2000). In the SSH algorithm, the frequency of local search was set to 3 and the number of local iterations was set to 3. For the ASH algorithm, the maximum number of local iterations was 3, e was 0.2, and the local threshold value was 0.6. Each variable was represented by 30-bit string with a total of 60 bits for each chromosome. The stopping criterion for all experiments was that 80% of the population had converged to the solution.

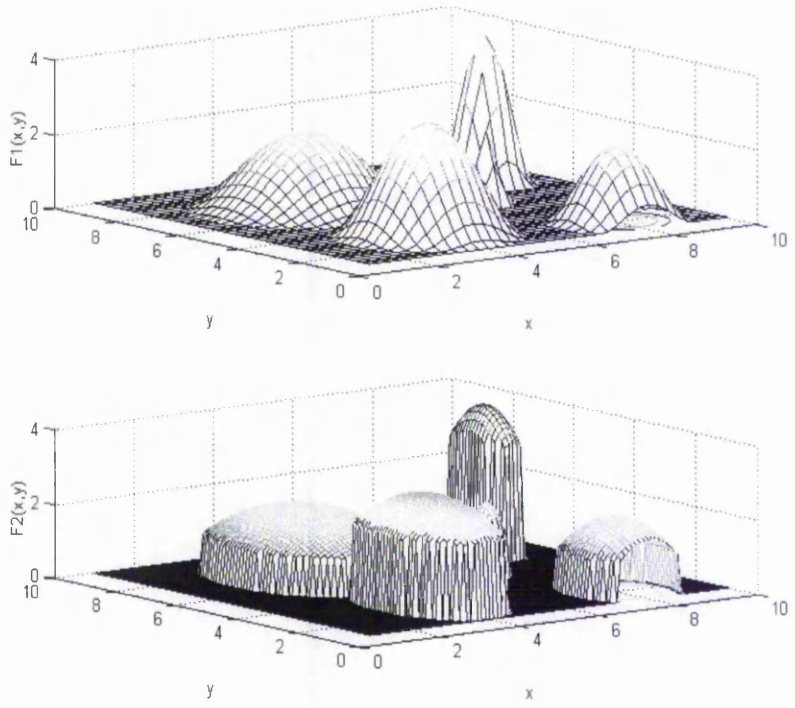

Figure 4.2: Fitness Landscapes for the Test Functions

4.4.1 Effects on convergence speed

In experiment 4.1, which aimed to evaluate the effect of learning strategy on convergence speed of hybrid algorithms, both the adaptive and standard staged algorithms used a probability of local search of 0.1, and population sizes of 800 and 1200 for F1 and F2, respectively (Espinoza 2001). The stopping criterion was that 80% of the population converged within a 0.000001 boundary of the best ever found fitness.

The results show, as expected, that increasing the fraction of the population that evolves according to the Lamarckian approach leads to an increase in the convergence speed. This increase is not linear. For example, when applying ASH algorithm to maximise F2, the speed of convergence increases sharply as the learning approach changes from pure Baldwinian (100% Baldwinian) to a mixture of 80% Baldwinian and 20% Lamarckian. In this interval the number of function evaluations decreases from 85,000 to about 37,000, while it decreases to 25,000 evaluations for the pure Lamarckian approach. Figure 4.3 shows the effect of learning strategy on the convergence speed of the adaptive staged

hybrid. The effect of learning strategy on the convergence speed of standard staged hybrid and the adaptive staged hybrid are similar for both test functions.

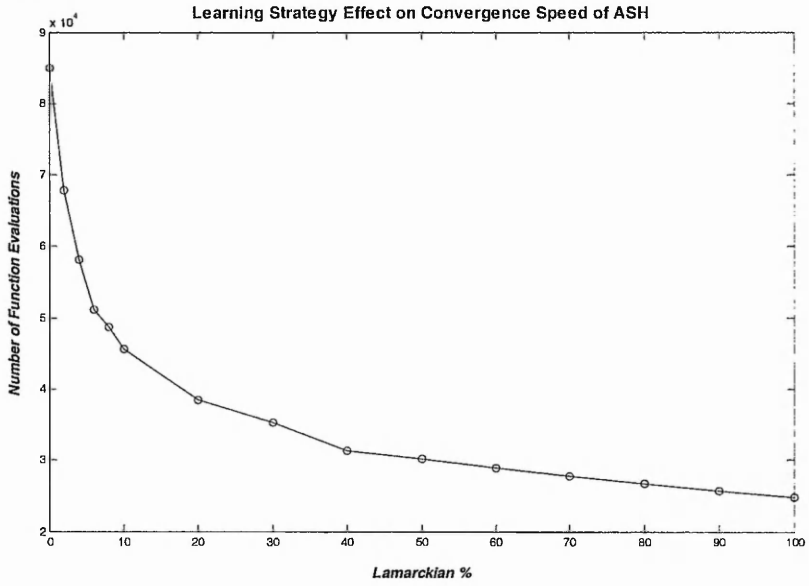

Figure 4.3: Effect of Learning Strategy on Convergence Speed (Experiment 4.1).

4.4.2 Effects on solution quality

The results of previous experiments show no clear relation between learning strategy and solution quality. This led us to consider how the local search probability interacts with the learning strategy and how this interaction affects the quality of solutions. An experiment, which will be referred to as 4.2, was carried out to consider the effect of local search probability on the solution quality for different population sizes (100, 400, 800, and 1200). The results of the experiments that optimising F2 using ASH algorithm show that as probability of local search increases, the effect of learning strategy on the solution quality becomes apparent (figure 4.4). The graphs in figure 4.5 show that, when the probability of the local search is kept small, the quality of the solution is insignificantly affected by the learning strategy. As this probability increases, the quality of the solutions degrades with an increasing Lamarckian percentage in the learning process. This means using small local search probabilities for both algorithms, even with pure Lamarckian, can produce high quality solutions because the disruption to schema processing caused by these small probabilities is neglected and has no effect on the global search process.

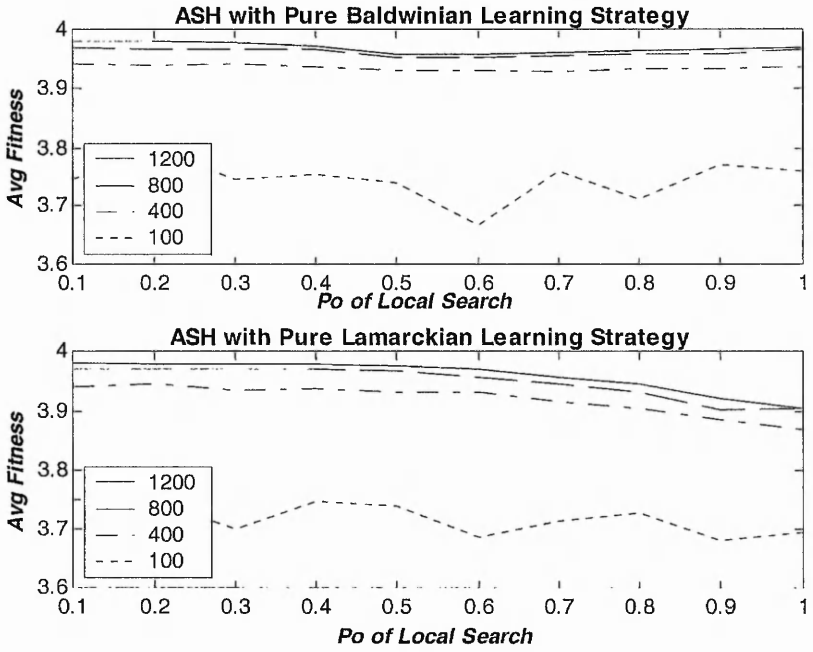

Figure 4.4: Effect of Learning Strategy and Search Probability on Solution Quality Using Different Population Sizes (Experiment 4.2).

The results in Figure 4.5 show that a mixture of 20% Lamarckian and 80% Baldwinian produces the most stable solution quality for F2, regardless of the probability of the local search. A mixture of 75% Baldwinian and 25% Lamarckian produces the most stable solution quality for F1 (Figure 4.6). The results from both hybrid algorithms show that a pure Baldwinian approach does not always produce the optimal solution quality and that the optimal learning strategy depends on the probability of local search. The use of small probabilities of local search produced the best quality of the pure Baldwinian approach due to its role in alleviating the hindering effect.

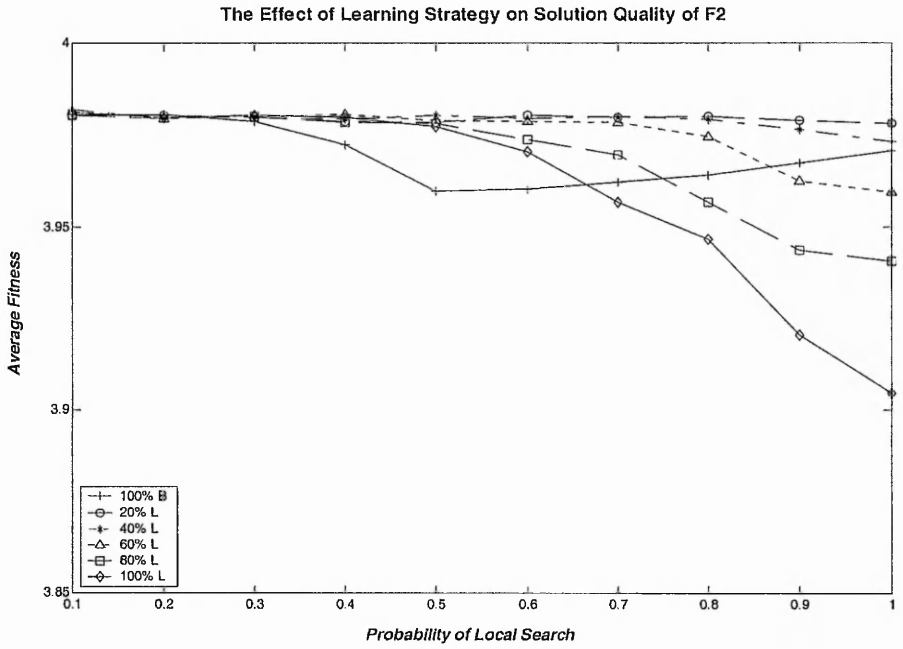

Figure 4.5: Solution Qualities for F2 (Experiment 4.2).

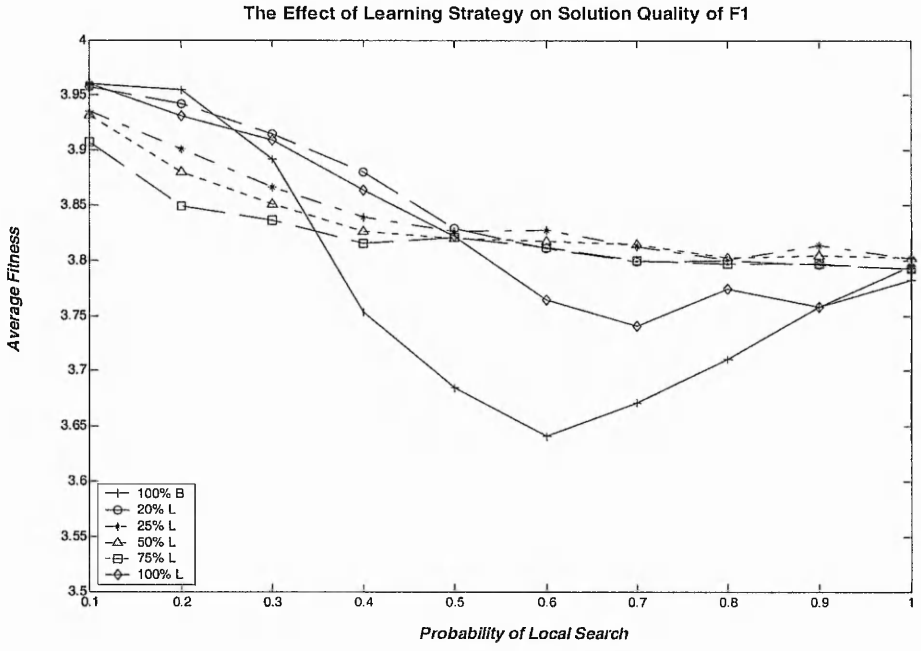

Figure 4.6: Solution Qualities for F1 (Experiment 4.2).

4.4.3 Effect on required population size

The aim of experiment 4.3 was to show how the probability of local search and learning strategy affect the minimum size requirements for both the adaptive staged hybrid and the standard staged hybrid. The results were obtained by using the bisection method. Starting with a population size of 10, the population size is doubled until the population converges to the desired solution quality. After the solution quality is attained, the population size is set midway between the current size and the last unsuccessful population size. This process is repeated until the difference between population sizes is less than or equal to 10. The stopping criterion was that 80% of the population converged within a 0.000001 boundary of the global optimum fitness. The fitness convergence was tested at the end of the global genetic search (the outer loop of the search) in order to ensure the convergence of both the population and its fitness and not the fitness only. The settings of other parameters were as in the previous experiments.

The results of SSH and ASH on the second test function are similar. Figure 4.7 shows that as the probability of local search increases, the population size decreases for a pure Lamarckian approach. On the other hand, with the pure Baldwinian strategy, the population size increases as the probability of local search increases. The hindering effect can explain the increase in the population size required for the pure Baldwinian strategy. The decrease in the probability of local search can reduce the hindering effect and that enables the algorithm to discriminate against non-optimal solutions and reduces the probability of genetic drift. For a pure Baldwinian strategy with local search probability of more than 0.4, the population size exceeds that of a pure genetic algorithm (minimum population size=640).

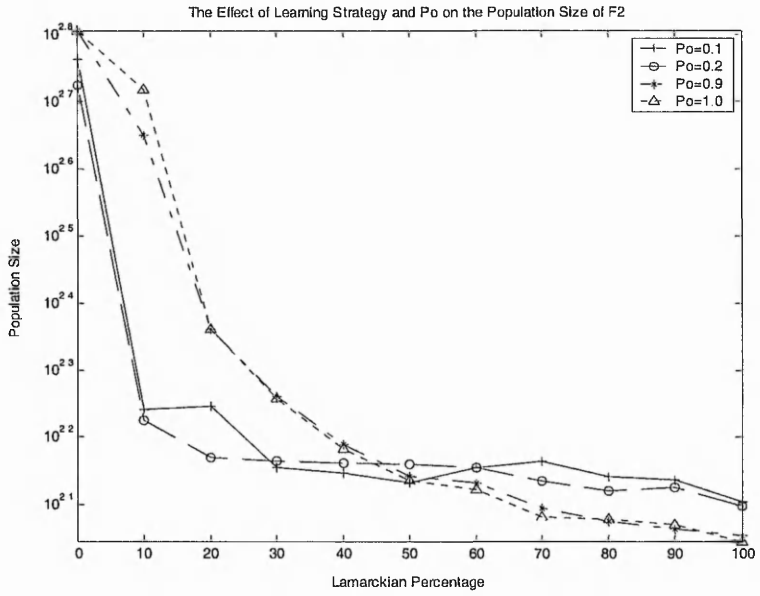

Figure 4.7: Effect of Learning Strategy and Search Probability on Population Size (Experiment 4.3).

The results also show that the relationship between the local search probability and the change in the population size depends on the learning strategy used. For example, using a partial Lamarckian approach of 50% or more, an increase in the local search probability results in a decrease in population size. With a partial Lamarckian approach of less than 50%, an increase in the local search probability leads to an increase in the population size. The increase in the percentage of partial Lamarckian approach reduces the probability of the genetic drift through reducing the percentage of the population that may experience the hindering effect. The increase in local search probability while increasing the partial Lamarckian increases the probability of mapping the second best individual to the global optimum and the signal difference becomes equal to the difference between the global optimum and the next non-converged best solution whose probability to convergence, in turn, increases by increasing the probability of local search and so on. The hindering effect is aggravated by either moving towards the pure Baldwinian or increasing the local search probability. For both hybrids, a decrease in population size leads to an increase in the convergence speed. In general, increasing the Lamarckian percentage decreases the population size and increases the convergence speed. The experiments also show that the solution quality of the pure Baldwinian approach is the optimal one and the solution quality is degraded as both the Lamarckian percentage and the probability of local search increase.

The solution quality for impure Baldwinian strategies, as shown in Figure 4.8, seems to be more dependent on the probability of local search than on the learning strategy.

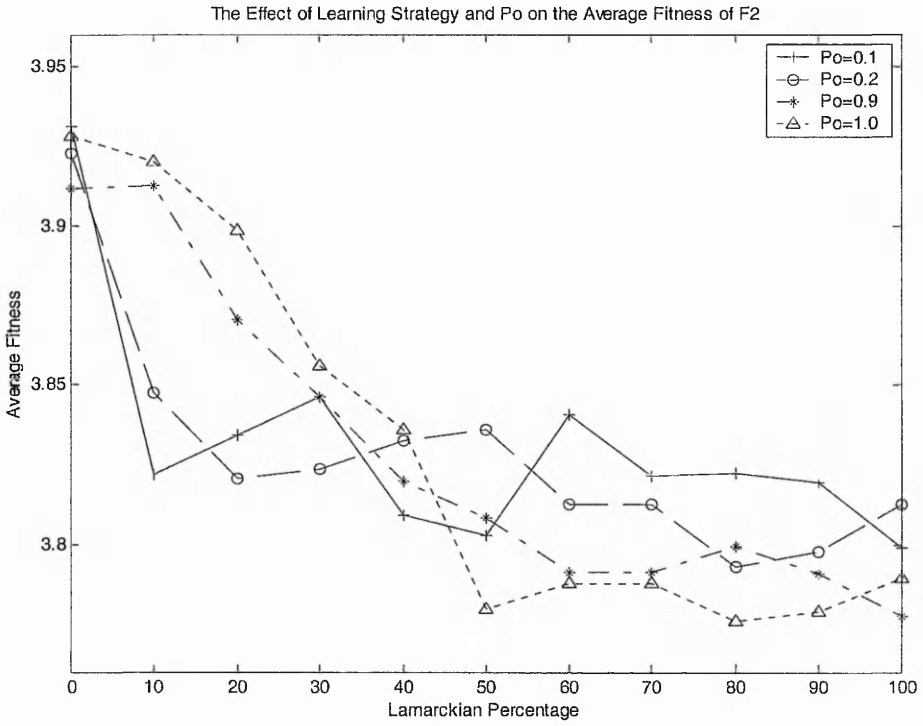

Figure 4.8: Effect of Lamarckian Proportion and Search Probability on Solution Quality (Experiment 4.3).

The local search can decrease both the standard deviation of the population and the signal difference between the best and second-best solutions, since the population size depends directly on the standard deviation of the population and the signal difference. A decrease in the former decreases the population size and a decrease in the latter increases the population size.

In addition to the hindering effect, the ability of keeping the decrease in the ratio of standard deviation to signal difference in the early stages of the search can explain the increase in the population size requirements for the pure Baldwinian approach. In a pure Baldwinian, the local search needs some help from evolution process to keep decreasing this ratio. The pure Baldwinian can reduce this ratio at the end of the local search. However, in the next global iteration, if the value of local knowledge is insufficient to keep the global genetic algorithm reducing this ratio, the algorithm will lose some of its resources (i.e. local function evaluations) without reducing that ratio. In this case, a high

probability of local search cannot lead to any reduction in the population size since it increases the probability of wasting the algorithm's resources. However, a low local search probability reduces the probability of wasting these resources while increasing the probability of maintaining the reduction in the above-mentioned ratio by the global genetic algorithm. In addition to the probability of local search, the effectiveness of pure Baldwinian in reducing the population size depends on the value of local knowledge and this depends on the method of local search and the fitness landscape.

On the other hand, the opportunity to keep the gained reduction in this ratio is improved by using a partial Lamarckian strategy. As the percentage of Lamarckian increases, the probability of keeping this reduction increases. An increase in the probability of local search increases the probability of reducing the ratio and reducing the population size.

Figure 4.9 shows the results of running the same experiment on the first test function. For a Lamarckian percentage of 65% or more, an increase in the probability of local search results in a reduction in the population size. For other percentages, an increase in this probability leads to an increase in the population size requirements. The convergence speed depends on the population size. As the population size decreases, the convergence speed increases. Comparison of Figures 4.7 and 4.9 shows that the switch point on the Lamarckian axis between increasing and reducing the population size is shifted from about 50% for F2 to about 65% for F1. This is due to the differences in the fitness landscape of both functions. While the local search can provide more significant local knowledge in F1 than in F2, an impure Lamarckian approach requires a more partial Lamarckian to accelerate the genetic assimilation process.

The local search method can provide more significant local knowledge from the landscape of F1 than F2. This is why the reduction in the population size requirements of F1, using a pure Lamarckian approach, is greater than that of F2. This also makes the genetic assimilation process more difficult for F1 using a pure Baldwin effect compared with F2. The genetic assimilation process is possible as a result of using a local search frequency of 3 in the SSH algorithm and the adaptive nature of the ASH algorithm. The use of small probabilities of local search can alleviate the hindering effect and enable the hybrid to converge to the global optimum even with small populations. The use of a local search probability of 0.1 enabled the algorithm to converge to the global optimum of F1 with a population size of about half of that needed by the pure genetic algorithm. The use of a partial Lamarckian can accelerate the genetic assimilation process and that can help to

reduce the population size required. The exact value of the switch point depends on the value of the local knowledge and the nature of the fitness landscape.

The convergence to the global optimum occurred with a population size of less than the $1.43l$ limit that is imposed on the non-uniform scaled fitness functions to protect the search from the genetic drift. The hybrid managed to converge to the global optimum using a population of 60 and 80 for the first and the second test functions respectively.

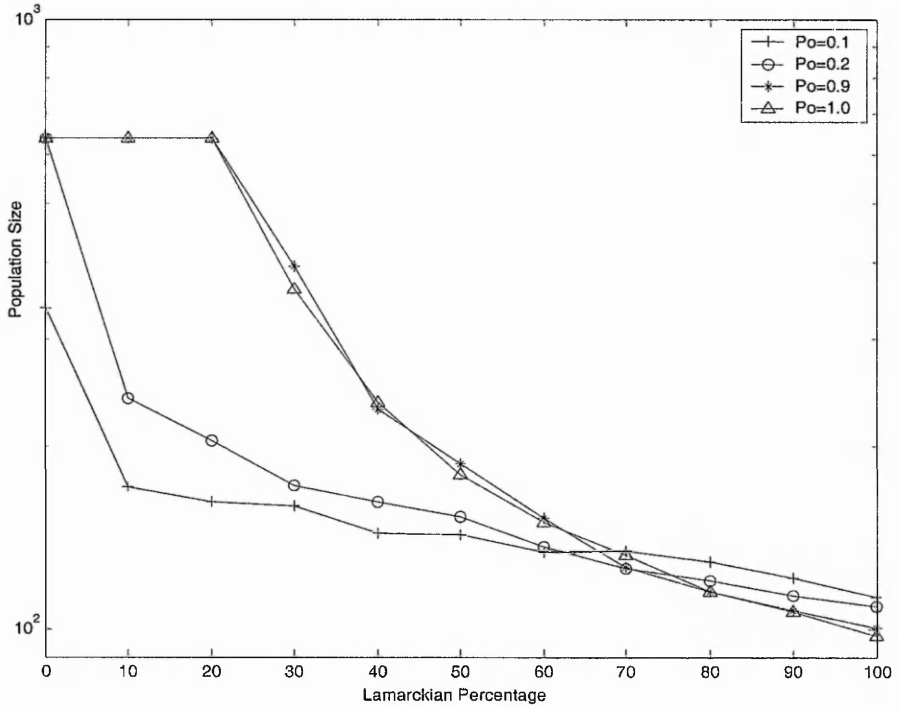

Figure 4.9: Effect of Learning Strategy and Local Search Probability on Population Size of F1 (Experiment 4.3).

Chapter 5 Improving Lamarckian Search

The power of genetic algorithms comes from their ability to combine both exploration and exploitation in an optimal way (Beasley et al. 1993a). The exploration and the exploitation abilities of a genetic algorithm can be enhanced by incorporating a local search method. A local search method can improve the ability of the genetic algorithm to explore the search space to isolate its most promising region through fair representation of the search regions (section 2.3). The refinement ability of a local search method can also enhance the quality of the solutions produced by a hybrid (Rosin et al. 1997). The combination can accelerate the search towards the most promising region and then towards the global optimum. This constructive form of cooperation between a genetic algorithm and a local search can produce an effective and efficient search algorithm.

However, there are other forms of interactions between the two search methods in addition to the one described above. The interference between the two methods can be somehow destructive. Because of its myopic nature, a local search method, when combined with the pure Lamarckian learning strategy, can disrupt the schema processing of the global genetic algorithm causing a premature convergence problem. This may force the hybrid practitioners to sacrifice the fast convergence speed associated with the pure Lamarckian strategy for high quality solutions associated with other learning strategies. The genetic operators can also destroy good local solutions that consumed a considerable amount of the algorithm's time to be constructed. Mathias et al. (1994) proposed the staged hybrid genetic algorithm to separate the global genetic algorithm and the local search method into distinct phases in order to decrease any form of destructive interactions between them. Other researchers (Rosin et al. 1997) (Land 1998) suggested choosing the control parameter values of the genetic operators consistent with their role in a hybrid in order to avoid any destructive effect of these operators on local search solutions.

A pure genetic algorithm utilises the selection operator and the standard genetic modification operators to exploit the available information in the current and previous solutions in order to direct the search towards a global optimum. The use of genetic modification operators as a technique to utilise the search information was replaced by other techniques to overcome some of the difficulties that face the pure genetic algorithms in solving real-world problems. The PMBGA algorithms (Pelikan et al. 1999b) use the search information to iteratively build a probabilistic model to learn the structure of a problem. Quantum-inspired genetic algorithms (Han and Kim 2002) utilise the search

information through using quantum bits and quantum gates. These algorithms were proposed as alternatives to the pure genetic algorithm. However, they have showed that the available search information can be utilised in different ways to achieve the same goal.

The pure genetic algorithm is making use of only a small part of the huge amount of search information that is available in genetic populations. Researchers have recognised the value of the genetic search information and tried to employ it in an optimal way. For example, it has been utilised to adapt the control parameters of genetic algorithms to improve their search performance (Eiben et al. 1999). The control parameters adaptation techniques are based on the fact that useful information can be extracted from search information. This information can be used to adapt the genetic search to the problem at hand while the genetic algorithm is seeking the global optimum. Search information has also been used to decide on performing a global genetic search or a local search in some hybrids (Lobo and Goldberg 1997) (Espinoza et al. 2001). It also has been used to decide on the optimal fraction of individuals that should perform a local search (Hart 1994) (Land 1998).

However, this valuable genetic search information is rarely used by the local search method incorporated in a genetic-local hybrid algorithm when solving real-world problems. In most cases, the global genetic algorithm provides the secondary search method with a starting solution that needs to be improved. The secondary method manipulates this solution and returns the improved solution to the genetic algorithm. Depending on the learning strategy adopted the global genetic algorithm decides either to replace the initial solution with the improved one or to assign its fitness score to the initial solution. Neither of the algorithms uses the information that is available to the other algorithm despite of its accessibility. Advanced local search methods usually need other local information in addition to the initial solution to accelerate the search (section 2.2). Due to the lack of positional information in the genetic search information, local search methods are unable to use it as additional local information. Clustering techniques can be used to provide relative positional information. However, since the advanced forms of local search methods usually work on the phenotype space and not on the genotype space, the cost of such clustering technique can be high and is dependent on the problem to be optimised.

In addition to the difficulties of using the available search information, advanced local search methods usually consume a considerable number of function evaluations (section 2.2). This can aggravate the hybrid algorithm's loss caused by any destructive interference between the local and the genetic algorithm.

To sum up, some of the good features of a search method that can enable it to be incorporated in a genetic algorithm in an effective and efficient way are:

- The ability to avoid any disruption to the genetic algorithm schema processing in order to avoid the premature convergence problem when utilised within a hybrid that adopt the pure Lamarckian strategy to improve the algorithm's efficiency.
- The ability to use the available genetic search information to improve the solution quality and/or the search efficiency.
- The cost of the search should be small to reduce the loss caused by any destructive interference.

In this chapter, a simple probabilistic search method is proposed as a secondary search method within a hybrid genetic algorithm based on the features described above. This search method was evaluated as a secondary method in a hybrid and as a stand-alone optimisation algorithm. The basic idea of this search method is reviewed in the following section with an illustrative example to explain its search mechanism. Then, the methodology used to assess the performance with the experiments that have been conducted are described. This chapter ends by a review of the experiments and a discussion of the results.

5.1 The proposed search algorithm

The proposed algorithm is a probabilistic method that works on the genotype space by making use of a group of the current population of solutions to estimate the structure of the improved solution. In this way, it aims to make use of some of the valuable genetic search information. It also aims to avoid disrupting the genetic schema processing by improving the solution in accordance with the global genetic search. The modification of the initial solution based on a group of solutions of the genetic population can provide the secondary search method with a partial global view of the problem at hand. Based on this view, the search method can produce a solution in the context of global view captured by the genetic algorithm. This form of search can minimise any conflict with the global genetic search. The partial global aspect of the search method can be controlled by the group size and the mechanism of selecting the group members.

This method is also characterised by its low costs. Its costs are equal to the costs of evolving a solution for a single iteration of the genetic search (i.e. one function evaluation per solution). This can help to minimise the loss of the hybrid's time in the case of any undesirable interference between the two search methods.

The algorithm assumes that each gene contributes uniformly to the fitness of the solution. Based on this assumption, the search method compares the genetic structure and the fitness of the solution to be improved with the structures and the fitness of a group of solutions selected from the current genetic population. Depending on the differences in both the structure and the fitness between this solution and the group members, the solution structure is modified in the direction of improving its fitness score. The new solution is evaluated and then, regardless of its new fitness, inserted back into the population.

5.1.1 The search mechanism

The algorithm starts with an initial solution and a randomly selected group of solutions from the current genetic population.

The algorithm assumes that the value of each gene in the initial solution represents the probability of that gene to have the value of one. It also assumes that the produced set of probabilities represents the initial probabilities of having the value of one in each gene of the optimal solution's structure.

This set of initial probabilities is modified according to the differences in the genetic structure and the fitness between the initial solution and the group members in order to estimate the optimal solution structure. An increase in the fitness score of a group member compared to the initial solution accompanied by a change in gene value from '0' in the initial solution to '1' in the group member means increasing the probability associated with that gene. The probability is increased by a value that is proportional to the increase in fitness score in order to bias the initial solution toward a better structure. However, if that increase in the fitness is accompanied with change from '1' to '0', the associated probability is decreased by the same value. A decrease in the fitness score in the previous cases will result in decreasing the probability in the first case and increasing it in the second by a value that is proportional to the absolute value of the difference in fitness score between the group member and the initial solution.

The algorithm compares every member of the group with the initial solution, in turn, and adjusts the genes probabilities in the way described above. The resulting set of genes' probabilities is compared against a set of randomly generated numbers over the range [0, 1]. If the gene probability is less than or equal to the random number generated, the value of that gene is set to one otherwise is set to zero. Then, the new structure is evaluated and returned as the new improved solution.

5.1.2 An illustrative example

The aim of this illustrative example is to provide insights into the basic idea and the mechanism of the proposed algorithm.

Suppose the algorithm is solving the MaxOne problem which aims to maximise the number of ones in a string of m binary digits. The fitness f of a solution to the MaxOne problem is the number of ones in its genetic code. Let the length of string to be optimised equals 6 (i.e. m=6).

Assume that an initial solution S_0 and a group of four members $\{S_1, S_2, S_3, S_4\}$ were selected to perform a search iteration. The genetic structures and fitness scores of these solutions are as given in figure 5.1.

The first step is to extract the initial probabilities of the optimal solution's structure from the initial solution. The initial solution's structure can be translated to {0.0, 0.0, 1.0, 1.0, 0.0, 1.0} of initial probabilities of the optimal solution's structure. The items represent the probability of each gene to have the value of '1'.

The second step is to calculate the fitness effect of each group member. This can be done by taking the absolute value of the difference between its fitness score and the initial solution fitness. This value is then normalised by dividing it by the sum of the absolute difference of each member from the initial solution. The calculation of the fitness effect of the group members are shown in figure 5.1. Each member of the group can affect the initial probability of the optimal solution in proportion to its fitness effect.

The third step is to calculate the effect of each group member on the initial probabilities of the optimal solution. The effect on the initial probability of each gene can be calculated by comparing the value of that gene in both the initial solution and the group member in turn. For example, by comparing the initial solution S_0 and S_1 there is an increase in the fitness score and the gene values of both solutions are identical, except the second gene. The value of that gene is '0' in S_0 and '1' in S_1. Since the increase in the fitness score is in favour of S_1, the initial probability should be modified to bias that gene towards the value of that in S_1. This can be done by increasing the probability of that gene by the fitness effect of that member. The probabilities of other genes are not modified since these genes are identical in both solutions. The effect of each group member on the initial probabilities of the optimal solution's structure is shown as the change in probabilities in figure 5.1.

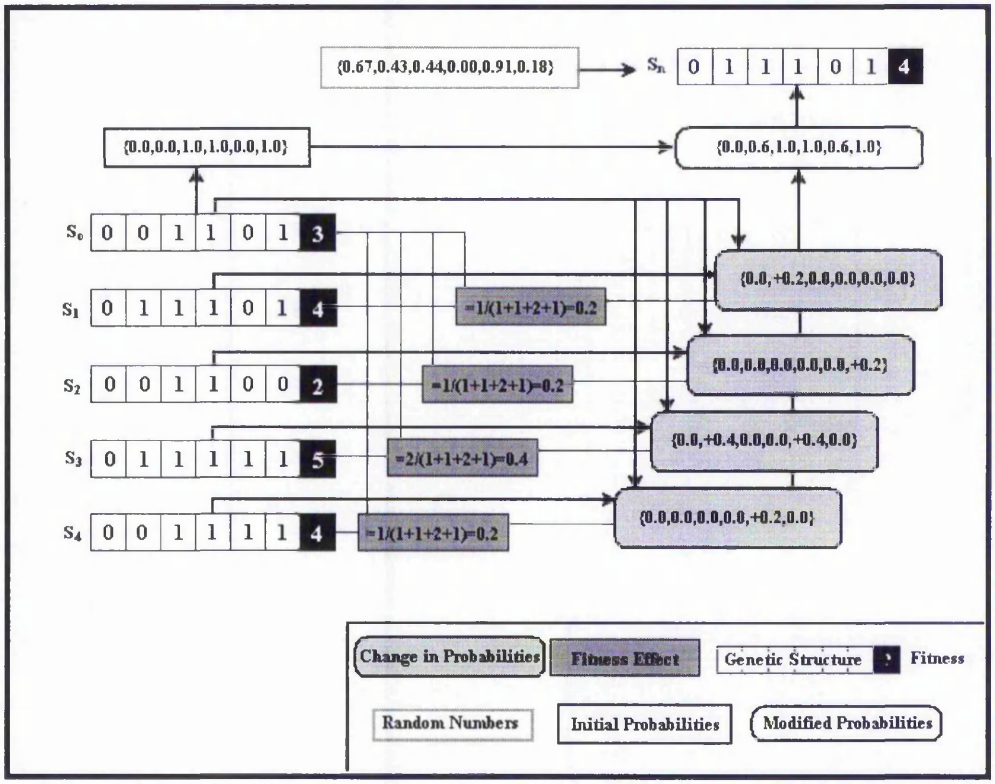

Figure 5.1: An Illustrative Example.

The change in probabilities induced by the group members are added to the initial probabilities to produce a new set of probabilities (Modified probabilities in figure 5.1) in the range of [0, 1]. These modified probabilities are compared with a set of random numbers to estimate the structure of the new solution. Assuming that the random number generator produced the following set of random numbers {0.69942, 0.433203, 0.440405, 0.000532, 0.910986, 0.18213}, the structure of the new solution will be {0, 1, 1, 1, 0, 1}. The fitness score of the new solution is four which shows an improvement in the fitness compared with the initial solution.

The change in probabilities induced by each member can be multiplied by some factor in the range (0, 1] to control the group strength effect. In this chapter, this factor will be referred to as the probability factor. In the previous example, the probability factor was set to 1.0. A value of 1.0 for this factor means that the group has the most possible effect. The algorithm can produce a completely different solution from the initial one. The new solution's structure will be more likely dependent on the structures of the group members. However, a small value of this factor can produce a solution with a structure that is likely to

be more similar to the initial solution and at the same time it is biased by the fitness and structures of the group members.

5.2 Empirical methodology used

In order to evaluate the performance of the proposed search method within a global genetic algorithm, a set of experiments has been conducted. In these experiments, the performance of a hybrid genetic algorithm that utilise the proposed search method was compared with the performance of the pure genetic algorithm. To maximise the interference between the two search methods, a local search iteration was performed after each global genetic iteration in the hybrid genetic algorithm. In order to assess the amount of disruption that this algorithm can cause on the schema processing, the algorithm was applied to every individual of the genetic population and the pure Lamarckian learning strategy was used. The decrease in the number of experiments that converge to the global optimum together with the convergence speed compared to the pure genetic algorithm was used as a measure of the disruption to the schema processing.

The optimisation problems were chosen to evaluate the basic assumption of the proposed method on the hybrid performance. Since the proposed method assumes that each gene of the solution contributes uniformly to the solution fitness, problems with different marginal fitness contribution of their genes were used. The selected problems include a lower boundary case with a uniformly scaled fitness function, an upper boundary case with an exponentially scaled fitness function and a case in between.

In these experiments, two empirical methodologies were followed. The classical methodology, which uses a set of known test functions to evaluate the performance of an algorithm, was used. The other methodology, which employs a problem generator (De Jong et al. 1997) (Kennedy and Spears 1998) to study the behaviour of evolutionary algorithms, has also been used to evaluate the proposed algorithm.

Following the classical methodology, three test functions were used to assess the performance of both the hybrid and the proposed search algorithm. Three functions with different marginal fitness contribution of their genes were used to evaluate the effect of the proposed search basic assumption. The first one is a uniformly scaled fitness function, which is the MaxOne problem, and the second is the BinInt problem (Thierens et al. 1998), which has an exponentially scaled fitness structure. The third test function is the Schwefel function (Mühlenbein et al. 1991), which is a non-linear multimodal function.

The fitness function of the MaxOne problem is defined as:

$$f(x) = \sum_{i=1}^{l} x_i \qquad x_i \in \{1,0\} \tag{5.1}$$

where l is the string length and x_i the alleles. The genes of the solution contribute uniformly to its fitness.

The fitness of the BinInt problem is defined as:

$$f(x) = \sum_{i=1}^{l} x_i 2^{l-i} \qquad x_i \in \{1,0\} \tag{5.2}$$

In contrast to the MaxOne problem, genes contribute exponentially to the fitness function. The contribution of one particular gene of the string is higher than the combined marginal fitness contribution of all the following genes.

The fitness of the Schwefel function is defined as:

$$f(x) = 418.982n + \sum_{i=1}^{n} -x_i \sin(\sqrt{|x_i|}) \qquad -500 \le x_i \le 500 \tag{5.3}$$

The Schwefel function is a multimodal function. It is characterised by a second-best minimum which is far away from the global optimum. The fitness function is non-uniformly scaled.

In addition to the classical empirical methodology, the problem generator methodology has been used. A problem generator is an abstract model capable of producing randomly generated problems on demand. The use of problem generators allows experimenting over a randomly generated set of problems rather than on a few hand-chosen examples. This can increase the predictive power of the results for a problem class as a whole.

The multimodal problem generator (De Jong et al. 1997) has been slightly modified and used. It generates O random l-bit strings, which represent the location of the O local optima (peaks in the original multimodal problem generator) in the space. The evaluation of a solution is carried out by locating the nearest local optimum in Hamming space. Then, the number of bits that the solution has in common with the nearest local optimum is divided by the string length. The result is multiplied by the amplitude of that optimum (the peaks have identical amplitudes in the original problem generator) and assigned as the fitness of the solution:

$$f(x) = A_o \max_{i=1}^{P} (l - \text{Hamming}(x, Optimum_i)) \tag{5.4}$$

where A_o represents the amplitude of the nearest local optimum which is $Optimum_i$. The global optimum is the optimum with the highest A_o.

A multimodal exponential problem generator has been proposed and used. The multimodal exponential problem generator is similar to the multimodal generator. The multimodal exponential generator also generates O local optima. The evaluation of a solution is carried out, first, by locating the nearest local optimum in Hamming space. Then, the nearest local optimum is used to generate a Hamming distance string of the current solution. The produced Hamming string is inverted and evaluated using the following fitness function:

$$f(x) = A_{fo} \sum_{i=1}^{l} x_i 2^{l-i} \qquad x_i \in \{1,0\} \tag{5.5}$$

where A_{fo} is an amplitude factor associated with each local optimum and x_i represents the value in the inverted Hamming distance string. The value of the amplitude factor is from the range (0, 1]. The value of this factor for the global optimum is 1.0.

The previous problem generators are efficient in terms of memory storage. Only the local optima and their amplitudes or amplitude factors need to be stored. The computation effort of fitness evaluations becomes very large as the number of local optima increases.

5.3 Experiments

The experiments that have been conducted aimed to evaluate the performance of the proposed algorithm within a hybrid. The performance is measured by investigating the search method's effect on the population size and the population convergence speed. The algorithm is also evaluated by studying its effect on the schema processing of the global genetic algorithm. In the last set of experiments, the search algorithm is evaluated as a stand-alone algorithm by comparing its performance to the pure genetic algorithm and a hybrid combining them.

5.3.1 Minimum population size

The first set of experiments was conducted to investigate the effect on the population size requirements by hybridising the proposed algorithm. The experiments used the bisection method, as described in chapter 4, to find the minimum population size required.

The hybrid used the simple elitist genetic algorithm with binary tournament selection, uniform crossover, and no mutation as the global search method. The crossover rate was set to 1.0. The proposed algorithm was used as an embedded search method that is performed by each individual of the population after each global genetic iteration. The experiments have been conducted using different group sizes. The group sizes tested were {0, 2, 4, 8,

16, 32}. A hybrid with a group size of 0 is identical to the pure genetic algorithm. In these experiments, two values of probability factors (0.5 and 1.0) were tested.

Figure 5.2 shows the results of experiment 5.1 which aimed to find the minimum population required for solving the MaxOne problem with a string length of 120 bit. The minimum population size is displayed as a function of the group size and the probability factor. Each point in this graph represents the average of 50 experiments. The figure also displays the convergence speed of the population to the global optimum. The graph shows that using a probability factor of 0.5 significantly reduces the minimum population size required for all the group sizes tested except the group size of 2. This reduction in the population size for the same set of group sizes is accompanied by a decrease in the convergence speed. However, using the statistical t-test shows that the decrease in the convergence speed is insignificant and the decrease in the population size is significant.

For a probability factor of 1.0 and for the same set of group sizes, the experiments show, as displayed in figure 5.2, that there is a significant increase in the convergence speed with insignificant increase in the population size.

The experiments show that the pure genetic algorithm (a hybrid with a group size of 0) outperformed the hybrid with a group size of 2. This can be explained in the terms of the partial view provided by the group size. The partial global view provided by that group size is very narrow and not enough to avoid a destructive interference between the two search methods especially when using a probability factor of 1.0, where the improved solution is more dependent on the partial view gained than the initial solution structure.

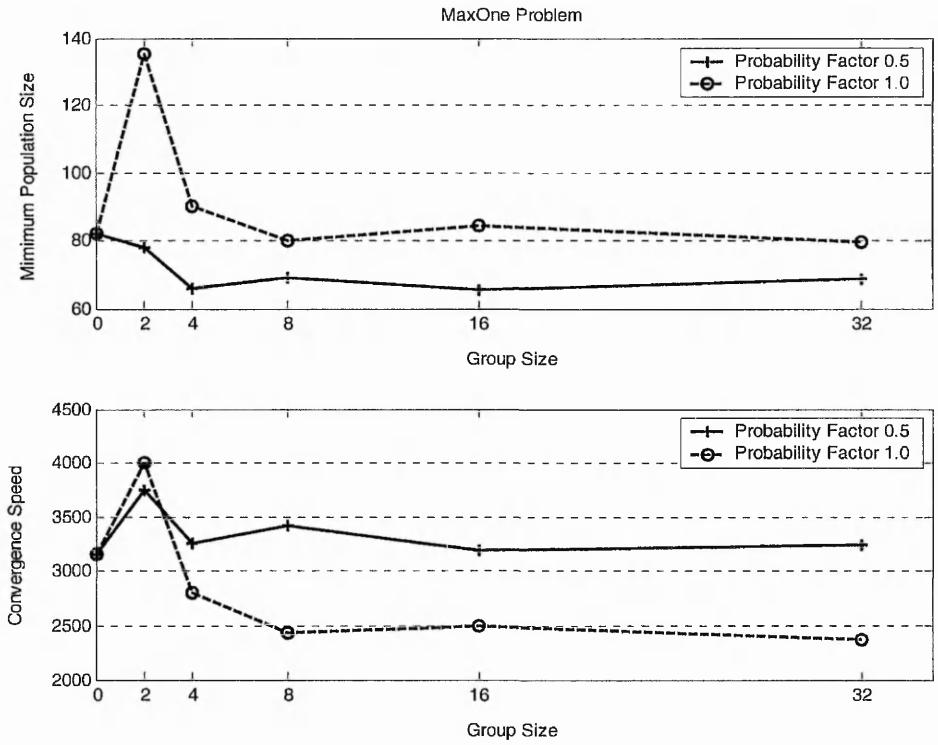

Figure 5.2: The Effect of Group Size and the Probability Factor on the Hybrid's Minimum Population Size and Convergence Speed of the MaxOne Problem (Experiment 5.1).

The results of experiment 5.2, which aimed to find the minimum population size required to solve the BinInt problem with a string length of 30, are shown in figure 5.3. Experiment 5.2 used the same control parameters as in the previous experiments. The graphs in this figure and the previous figure show similar trends for a probability factor of 0.5. The plots in figure 5.3 show a significant reduction in the population size with insignificant decrease in the convergence speed for a probability factor of 0.5. However, a probability factor of 1.0 shows a significant increase in the convergence speed with a considerable decrease in the population size. The graphs also show that a group size of two with a probability factor of 0.5 improves the performance of the hybrid in terms of population size required without a significant increase in the cost in terms of convergence speed.

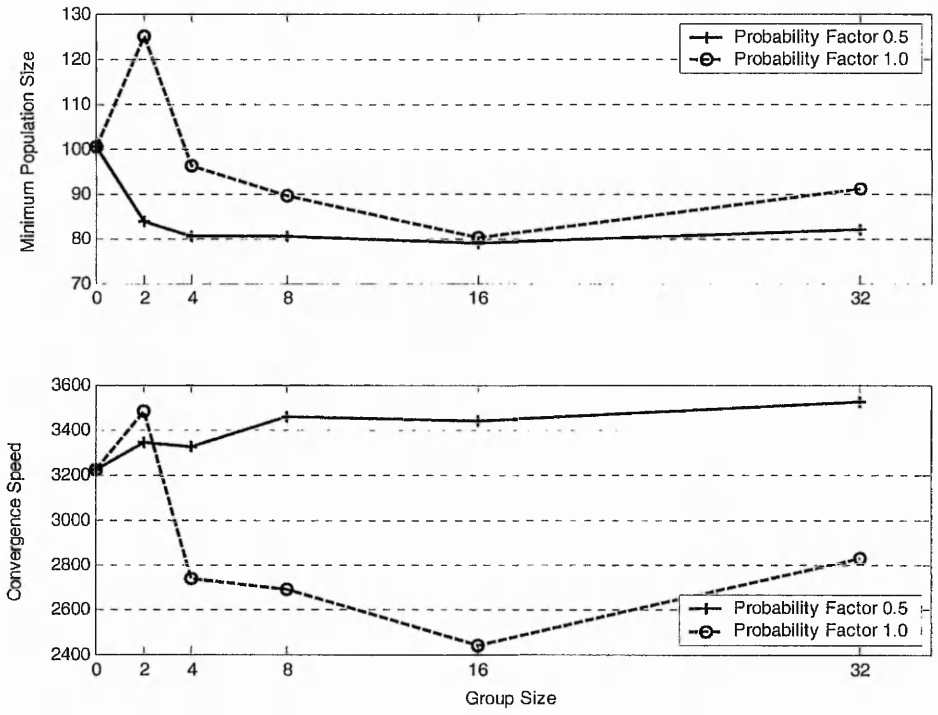

Figure 5.3: The Effect of Group Size and the Probability Factor on the Hybrid's Minimum Population Size and Convergence Speed of the BinInt Problem (Experiment 5.2).

The results show that utilising the proposed search algorithm within a genetic algorithm using a suitable group size can improve the genetic performance in terms of the population size, the convergence speed or both of them.

5.3.2 Effect on schema processing

The aim of this set of experiments was to assess the disruption to the schema processing caused by utilising the new algorithm. This can be accomplished by comparing the number of times each algorithm converges to the global optimum with that of the pure genetic algorithm as a first step. A decrease in this number indicates that a disruption was induced that misguides the overall search. However, an increase in that number indicates no disruption regardless of the convergence speed. In the case of no improvement in number of times a hybrid converged to the global optimum, a second step of evaluation is needed. In the second step, the convergence speeds are compared. In the case of a decrease in the convergence speed of a hybrid, the algorithm caused a disruption. However, any improvement in convergence speed indicates no disruption.

Experiment 5.3 was conducted on the BinInt problem with a string length of 30. The algorithm used the same control parameters that were used in the previous experiments. The algorithm used a population size of 150. The stopping criterion was satisfied if 97% of the population were identical or a maximum number of function evaluations was exceeded. The maximum number of function evaluations was set to 30,000.

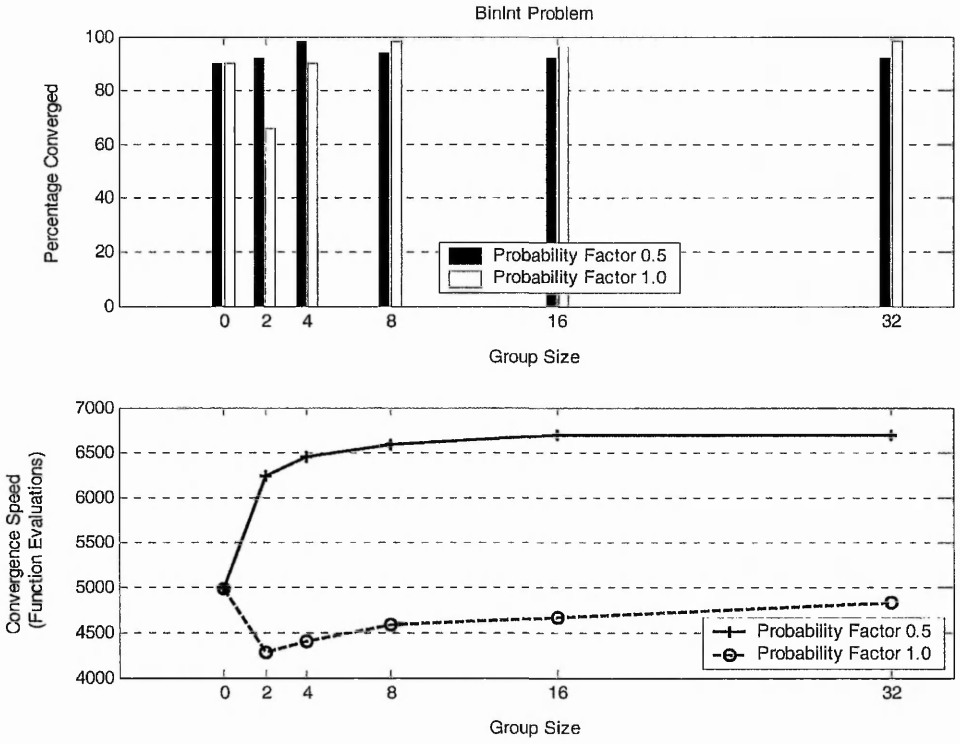

Figure 5.4: Effect on the Schema Processing when Solving the BinInt Problem (Experiment 5.3).

The results of experiment 5.3 are shown in figure 5.4. The graph shows that all the tested group sizes and probability factors, except the combination of a group size of two and a probability factor of 1.0, show no decrease in the number of experiments that converged to the global optimum. This means no disruption for the schema processing was induced by incorporating the proposed search method with the specified group sizes and probability factors. The graph also shows that using a probability factor of 1.0 increased the convergence speed of the population to the global optimum for all group sizes tested. The diagram also shows that a probability factor of 1.0 is more suitable for large group sizes, whereas a value of 0.5 seems more suitable for small sizes. This ability of a large group size to capture a wide view of the search space when combined with a strong effect on the initial solution can direct it in the right direction. However, using the same strong effect with a

small group size which provides a very narrow view of the search space can misguide the search.

In experiment 5.4, the algorithms were used to optimise the Schwefel function with ten variables. The chromosome length of each variable was 16 bit. The algorithm used a global simple elitist genetic algorithm with binary tournament selection and two-point crossover. The crossover rate was 0.6 and the mutation rate was 0.000001. The population size was 300. The stopping criterion was a maximum number of function evaluations. The value of this parameter was set to 300,000.

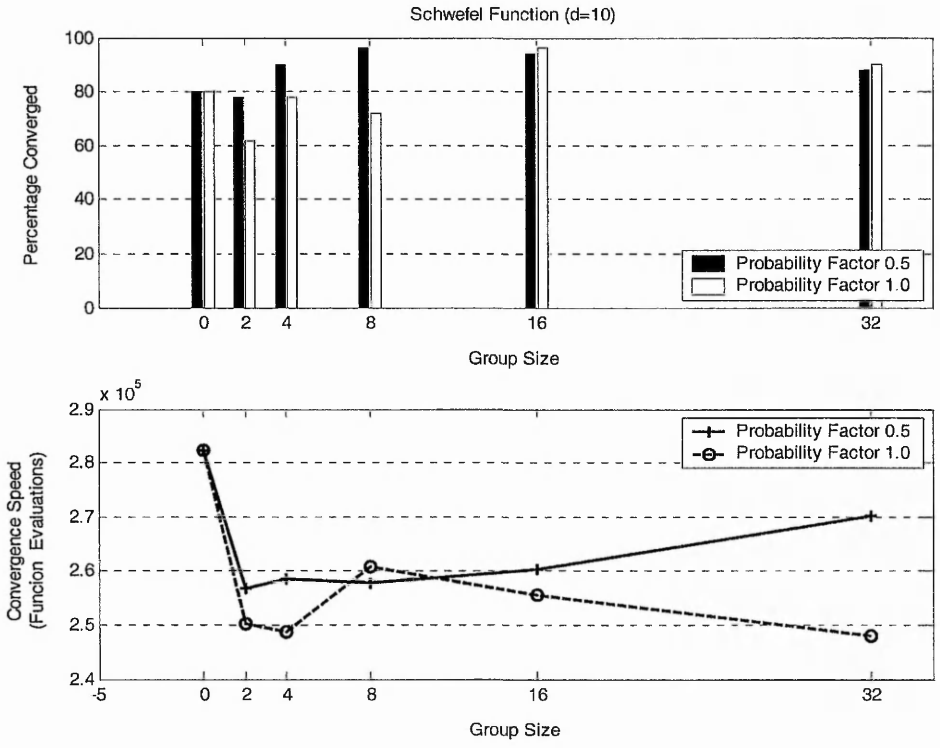

Figure 5.5: Solving the Schwefel Function (Experiment 5.4)

The results of this experiment are shown in figure 5.5. The graph shows that with a probability factor of 0.5 and for a group size greater than or equal to four, there is always an increase in the number of times of finding the global optimum. However, for a probability factor of 1.0, the increase occurred with population sizes of 16 and 32. The figure also shows that this improvement in the number of times of reaching the global optimum is always accompanied by an improvement in the convergence speed. The graphs also show that a larger group size is needed to capture a good partial view of the search space

compared to the sizes needed in the previous experiments. This is due to the nature of the fitness landscape which is more complicated than the previous problem.

The multimodal exponential problem generator was used to evaluate the performance of the proposed algorithm as a secondary search method in a hybrid. The experiment, which will be referred to as experiment 5.5, was carried out using five and ten randomly generated local optima with amplitude factors of {0.2, 0.4, 0.6, 0.8, 1.0} and {0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0}, respectively. The chromosome length was set to 30. An elitist generational genetic algorithm with binary tournament selection and uniform crossover was used as the global search algorithm. The population size was set to 100. The crossover rate was 1.0 and the mutation probability was 0.000333 ($p_m=1.0/(l\times N)$). Experiments were run for a complete convergence of the population or a maximum of 10,000 function evaluations.

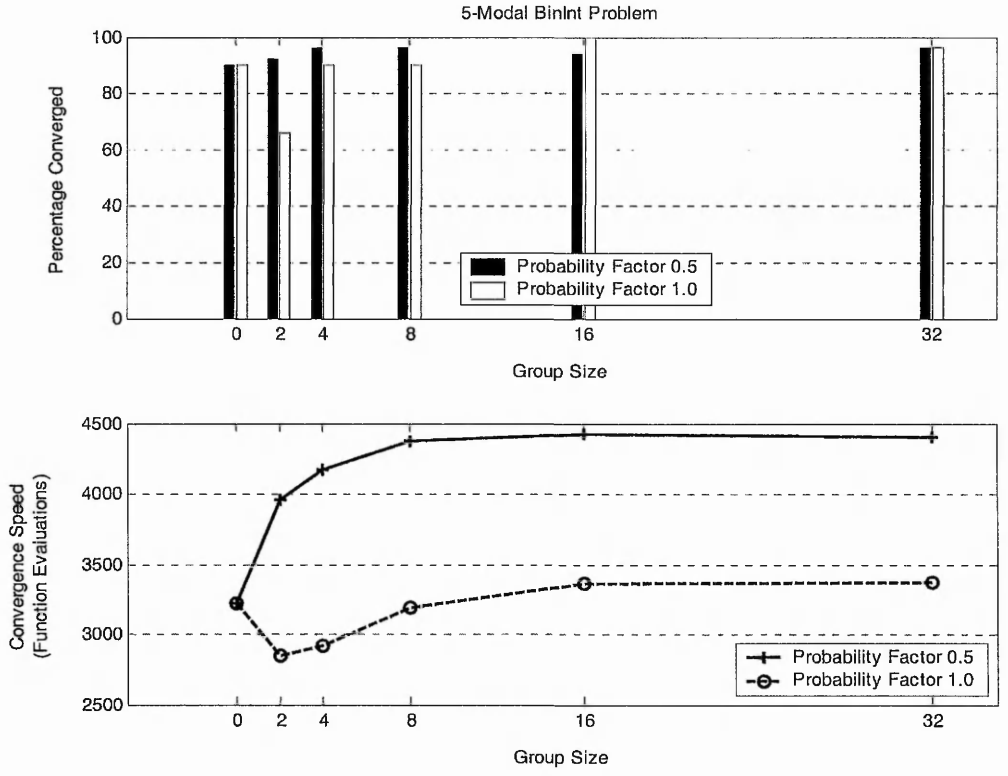

Figure 5.6: The Results of the Five-modal Exponential Problem (Experiment 5.5).

The graphs in figure 5.6 and 5.7 show the results of this experiment. These plots are similar to that shown in figure 5.4 for solving the BinInt problem. The increase in the convergence speed of the pure genetic algorithm can be a result of the use of the mutation operator in

this experiment in contrast to the BinInt problem where no mutation was used. In general, the two figures show the trends that have been described before.

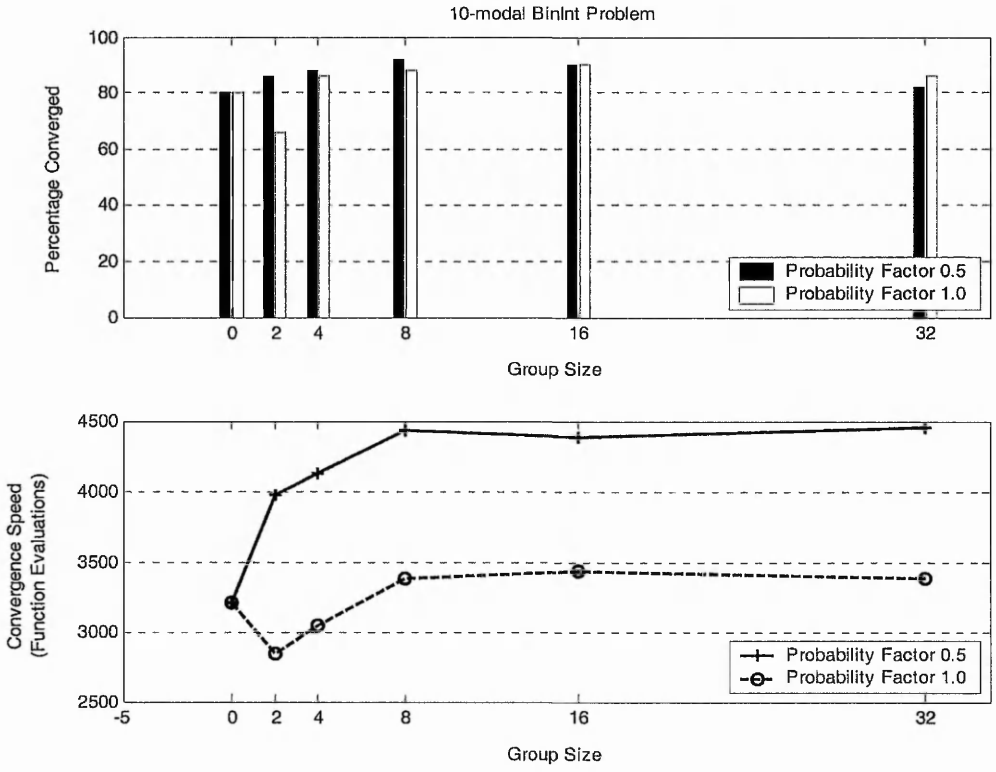

Figure 5.7: The Ten-modal Exponential Problem (Experiment 5.5).

5.3.3 The search method as a stand-alone algorithm

The proposed algorithm was tested as a stand-alone algorithm to optimise three multimodal functions. It has been used to optimise the Schwefel function with ten variables, the multimodal problem and the multimodal exponential problem. The algorithm used a population size equal to that used by the pure genetic algorithm and the hybrid. The performance of the algorithm was compared with that of the pure genetic algorithm and a hybrid combined them.

The results of experiment 5.6, which aimed to employ the proposed search algorithm to solve the Schwefel problem, demonstrated that the pure genetic algorithm outperformed the proposed algorithm using different group sizes and probability factors. Figure 5.8 shows the average fitness of the population as a function of the number of function evaluations for the algorithm as a stand-alone and a secondary algorithm compared to that of the pure genetic algorithm. The algorithm shown in this figure used a group size of 8 and a probability

factor of 0.5. The graphs demonstrate that despite the poor performance of the proposed algorithm as a stand-alone algorithm compared to the pure genetic algorithm, their combination outperformed either of them. The graphs depict the convergence of 50 experiments of each algorithm.

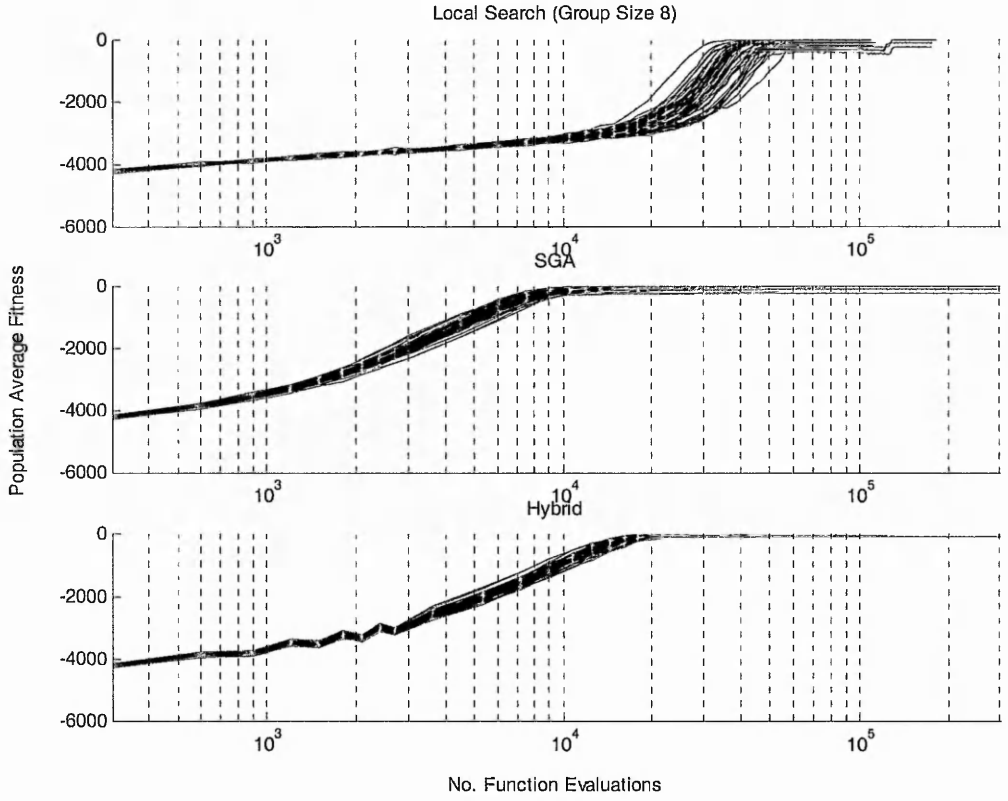

Figure 5.8: The Convergence Details of the Schwefel Function (Experiment 5.6).

A multimodal exponential problem generator with five local optima was used to evaluate the performance of the proposed algorithm as a stand-alone algorithm in experiment 5.7. The amplitude factors were set to {0.2, 0.4, 0.6, 0.8, 1.0}. The experiment demonstrated that the search algorithm, in most cases, performed worse than the pure genetic algorithm. In few cases however, the proposed algorithm outperformed the pure genetic algorithm in terms of the number of experiments that converged to the global optimum. It also outperformed the hybrid in terms of the convergence speed. The convergence of the population for the three algorithms as a function of the number of function evaluations is depicted in figure 5.9. The group size used was 16 and the probability factor was set to 1.0.

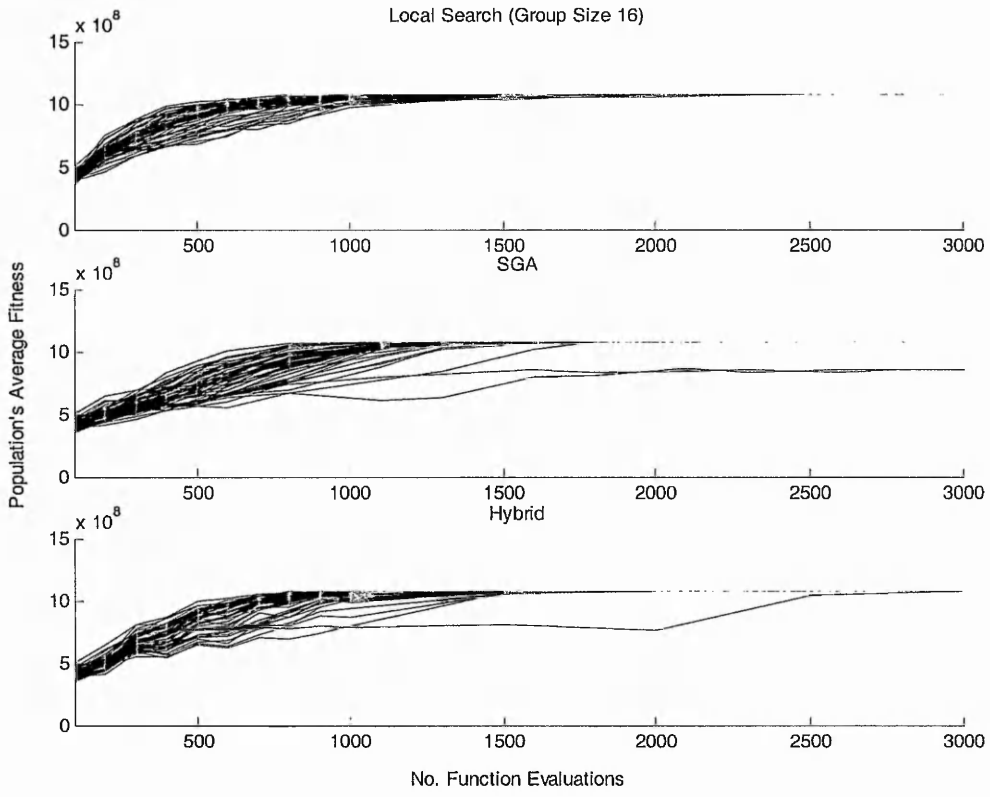

Figure 5.9: Comparing the Convergence of the Proposed Algorithm with the GA and the Hybrid on 5-modal Exponential Problem (Experiment 5.7).

In experiment 5.8, the multimodal problem generator with number of local optima of five has been used to evaluate the search method performance. The string length of solutions and the population size were set to 100. The amplitude factors were set to {5.0, 4.0, 3.0, 2.0, 1.0}.

An elitist generational genetic algorithm with binary tournament selection and two-point crossover was used as the global search algorithm. The crossover rate was 0.6 and the mutation probability was 0.0001. The experiment was run for a complete convergence of the population or a maximum of 100,000 function evaluations.

The results of experiment 5.8 were encouraging. They show that the proposed algorithm outperformed the pure genetic algorithm using different population sizes when combined with a probability factor of 0.5. The algorithm converged faster to the global optimum than the pure genetic algorithm. It also outperformed a hybrid that combined it with the genetic algorithms using a group size of 2 and 4 (figure 5.10). However, the stand-alone algorithm

showed poor performance when using a probability factor of 1.0. The algorithm with a probability factor of 1.0 can guide the search to non-optimal solutions. The probability of guiding the search towards non-optimal solution increases as the group size decreases. However, a hybrid with a probability factor of 1.0 outperformed the pure genetic algorithm, and both the stand-alone algorithm and the hybrid with a probability factor of 0.5 for group sizes of 8, 16 and 32.

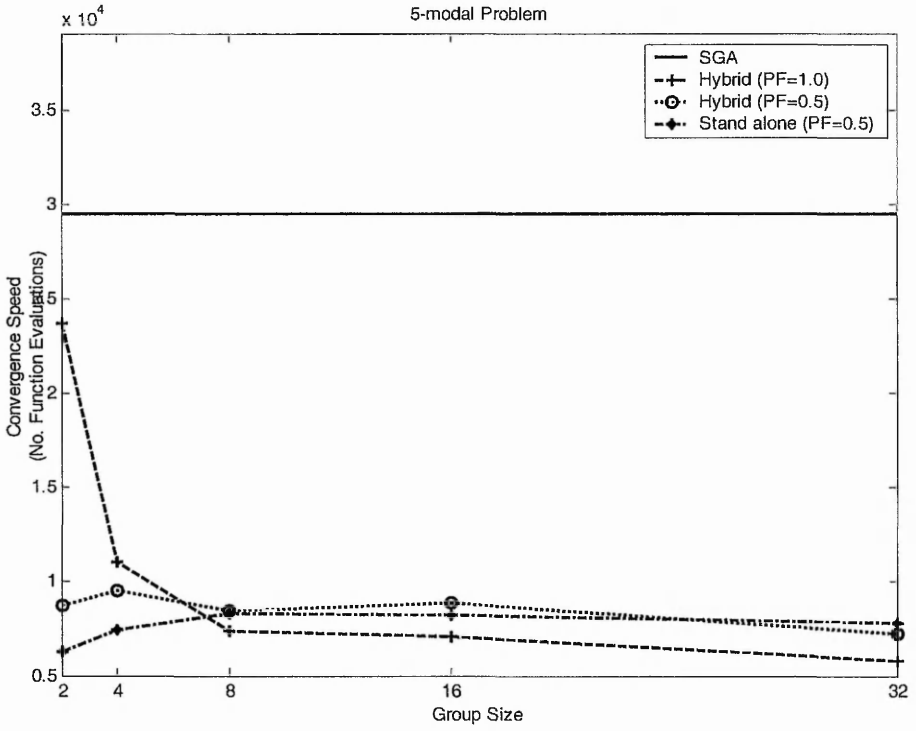

Figure 5.10: The Convergence Speed of the Proposed Algorithm as a Stand-Alone Algorithm (Experiment 5.8).

Figure 5.11 compares the performance of the algorithm using different group sizes and probability factors. The graph demonstrates the fast convergence speed associated with a probability factor of 1.0. The graph also shows that this convergence can be towards non-optimal solutions. There is a decrease in the number of experiments that converged to non-optimal solution accompanied with an improvement in the convergence speed as the group size increases. In contrast to the probability factor of 1.0, the probability factor of 0.5 shows a decrease in convergence speed as the group size changes from 2 to 32 with the ability to find the exact global optimum in all cases.

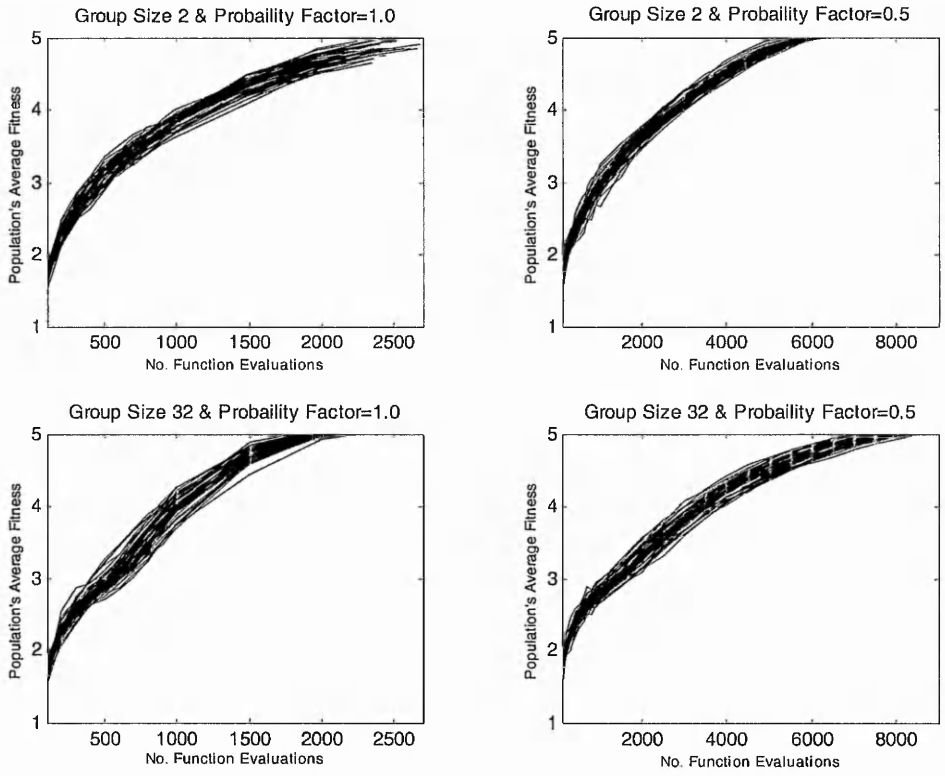

Figure 5.11: Comparing the Effect of the Probability Factor and the Group Size on Algorithm Performance (Experiment 5.8).

Figure 5.12 compares the population convergence speed of the algorithm as a stand-alone optimisation technique with that of the pure genetic algorithm and a hybridisation of them. This graph compares an algorithm with a probability factor of 1.0 and a group size of 32. The graphs show that a hybridisation can get the best out of the two search methods. It produced an algorithm that was able to find the global optimum in all the experiments, in contrast to the stand-alone algorithm which can miss that optimum some times. The hybrid was able to employ the ability of the pure genetic algorithm to reach the global optimum in all experiments and utilise the fast convergence speed of the secondary method to produce an effective and efficient algorithm.

The basic assumption of the proposed algorithm, which states that each gene contributes uniformly to the fitness of the solution, can explain the good performance of the algorithm, as a stand-alone optimisation technique, on the multimodal generator problem compared to the poor performance on the other two problems.

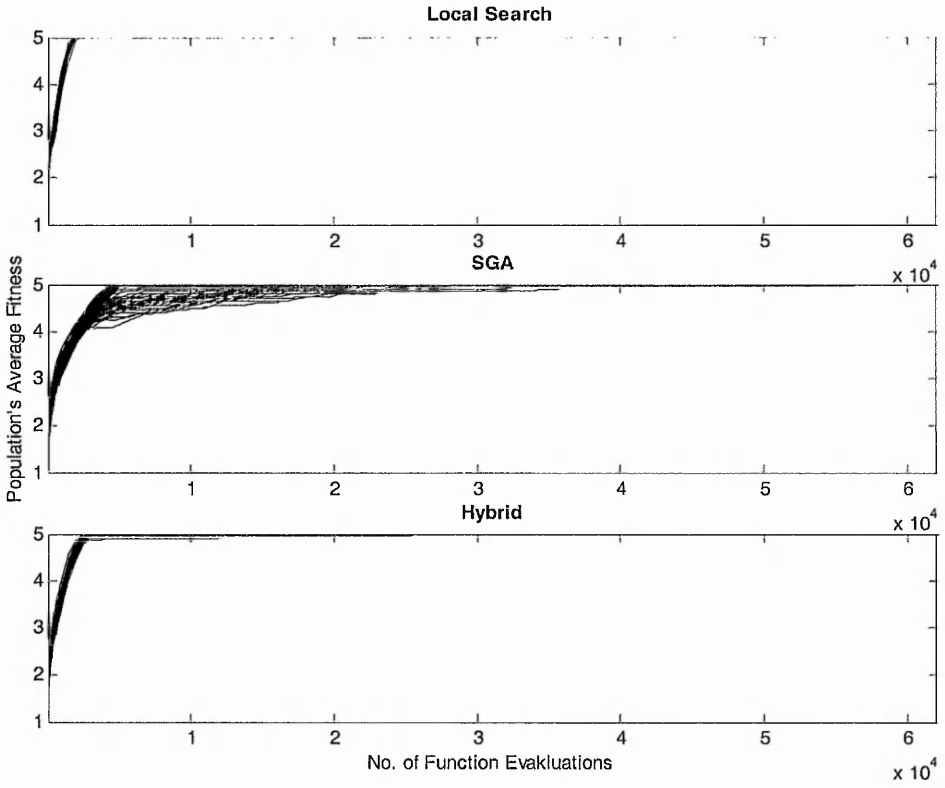

Figure 5.12: Comparing the Convergence of the Proposed Algorithm with the Pure GA and the Hybrid on 5-modal Problem (Experiment 5.8).

However, the encouraging performance of the algorithm as a secondary search method, even when applied to non-uniformly scaled fitness functions, can be explained as follows: The genes of non-uniformly scaled fitness functions converge at different rates (Thierens et al. 1998). The most important genes converge towards their optimal value before the less important genes. The proposed algorithm concentrates on the differences in the population structure and fitness to modify the non-identical genes. The algorithm does not modify the identical genes. These non-identical genes in the non-uniformly scaled problems are the genes that converge at a slower rate than the identical genes which has been converged to their optimal value as a result of the genetic search. The algorithm uses a sample of the genetic population to determine the genes that have not been converged yet. This sample involves the initial solution and a selected group of solutions. The accuracy of the algorithm in determining the converged genes increases as the sample size increases. This ability of determining the already converged genes in the population reduces the possibility of disrupting the genetic schema processing. This, in turn, can reduce the probability of facing premature convergence problems and can accelerate the search towards the global optimum. This can explain the good performance of the hybrid that uses large group sizes.

One of the possible ways of improving the performance of the proposed algorithm is to use a variable group size for each iteration. It is also possible to set the values of the probability factor depending on the group size used. This can be done in accordance with the findings of the experiments of this chapter. These experiments show that high probability factors are suitable for large groups and low factors are more suitable to small groups. The probability factor can be made adaptable to the group size in that way.

Chapter 6 Evolution to adapt the duration of local search

The success of a genetic algorithm in solving a given problem efficiently depends on its success in achieving a balance between exploration and exploitation. The correct balance depends mainly upon the fitness landscape of the problem to be solved in addition to the genetic algorithm setup. The genetic search can adapt to a variety of fitness landscapes through adapting the control parameters associated with the genetic operators (Beyer and Deb 2001).

The problem of striking a balance between the global and local search tools in a hybrid, in order to adapt the algorithm to a given problem, can be viewed as a problem of finding optimal local search control parameter settings. Evolutionary self-adaptation is one possible way to adapt the local search control parameters within a hybrid and implicitly optimise the hybrid performance to a given problem.

This chapter aims to investigate the advantages and disadvantages of applying evolution to self-adapt the control parameters associated with the utilisation of the local search within a hybrid genetic algorithm. It targets studying the effect of this form of adaptation on the hybrid's performance on different classes of test functions. The aim is also to gain some insight into the influence of the interactions between learning strategy and local search method on the self-adaptation behaviour. The possible ways to improve this form of adaptation were also studied.

This chapter starts with a brief review of adaptation in genetic algorithms. Then, it reviews adapting the local search control parameters as way of optimising the utilisation the hybrid's time. The effect of the implicit use of the productivity metric as a measure to decide on local search control parameter values on the self-adaptive hybrid's performance is analysed in that section. This chapter is drawn to an end by presenting the results of the experiments that have been conducted to support this analysis.

6.1 Adaptation in genetic algorithms

Different mechanisms for selecting the most appropriate control parameter values have been used to adapt a genetic algorithm to a specific problem. The simplest mechanism is the tuning technique (static adaptation) where a control parameter has a constant value throughout the search. A control parameter is tuned through external control by experimenting different values to choose the most appropriate one. De Jong (1975) determined experimentally recommended values for the rate of single-point crossover and

bit mutation. Grefensette (1986) optimised some of the control parameters using a genetic algorithm as a meta-algorithm.

The control parameters interact with others in a nonlinear manner (Ochoa et al. 1999). For this reason, finding the correct control parameter values is a time-consuming task (Eiben et al. 1999), and it is difficult to formulate general rules about their values (Tuson and Ross 1998). The use of constant control parameters, despite of being recommended by Ochoa et al., is in contradiction to the evolutionary spirit of genetic algorithms (Eiben et al. 1999). There is theoretical and empirical evidence that shows the most effective control parameter settings vary during the genetic search (Spear 1995) (Smith and Fogarty 1997) (Tuson and Ross 1998).

In order to eliminate the need for tuning the control parameters by external means, and to increase the search efficiency (Toussaint and Igel 2002), these control parameters can be adapted through the execution of a genetic algorithm. The adaptation process can be achieved by involving some modification mechanism that adjusts a control parameter without external control.

A deterministic update rule can be used to update the control parameter value without using any feedback from the current state of search (Hesser and Manner 1991 cited in Hinterding et al. 1997) (Bäck and Schütz 1996) (Michalewicz 1996). However, formulating a deterministic update rule can be harder than finding a good set of control parameter values (Tuson and Ross 1998). It may be advantageous, therefore, to employ adaptive learning rules, where some feedback from the search is utilised to modify the control parameters. These adaptive algorithms have been gaining popularity in the recent past due to their flexibility in adapting to different fitness landscapes (Beyer and Deb 2001).

Different learning rules have been involved to adapt the genetic algorithm control parameters. Davis (1989) proposed a learning-rule technique that required a lot of bookkeeping to adjust the probability of the reproduction operator according to its success at producing good offspring. He used this technique to drive a non-adaptive time varying schedule of reproduction operators. Arabas et al. (1994) proposed a Genetic Algorithm with a Varying Population Size (GAVaPS) to adapt the population size using a rule that assigns each individual a relative life-time, at the time of creation, in proportion to the average fitness of individuals within the population. Julstrom (1995) used a scheme similar to that proposed by Davis but which requires less bookkeeping to adapt the operator probabilities based on their recent contributions to the algorithm performance. Tuson and Ross (1996)

proposed the Cost Based Operator Rate Adaptation (CBORA) approach, where the algorithm collects information on operators' performance to adjust their probabilities.

The idea of evolution, which is modelled through genetic algorithms, can be used to optimise the performance of a genetic algorithm on a particular problem in order to solve it efficiently (Eiben et al. 1999). The idea can be implemented by encoding the control parameters into the chromosome(s) of the individual and undergo genetic operations. This approach uses the productivity metric (De Jong and Spears 1992) in an implicit fashion to select the most appropriate control parameter values. Good control parameter values will lead to good individuals and these will probably have more chances to survive and propagate the encoded control parameter values.

Algorithms that use this form of adaptation are usually referred to as evolutionary self-adaptive algorithms (Bäck et al. 1997). Evolutionary algorithms benefits from implementing this method in solving a wide range of real-world problems (Toussaint and Igel 2002). Evolutionary self-adaptation is commonly regarded as a speciality of evolution strategies (Bäck et al. 1997). However, it has been successfully extended to genetic algorithms. An early example of this was the punctuated cossover operator (Schaffer and Morishima 1987). This method was offered for adapting both the number and position of crossover points for a multipoint crossover operator in genetic algorithms. Extra bits were added to the representation of a solution to encode crossover points and were allowed to evolve over time. Other approaches incorporate the mutation rate into the representation of individuals (Bäck 1992) (Smith and Fogarty 1996). The evolutionary self-adaptation technique was also used to decide between two-point and uniform crossover operator (Spears 1995), to adapt the mutation and crossover probability in a genetic algorithm with a varying population size (GAVaPS) (Bäck et al. 2000), and to adapt the mutation and crossover rate of a Co-operative Co-evolutionary Genetic Algorithm (CCGA) (Potter and De Jong 1994) (Iorio and Li 2002).

The term self-adaptive is usually used to describe algorithms that adopt the evolutionary self-adaptation technique (Angeline 1995) (Eiben et al. 1999) (Smith and Fogarty 1997) (Toussaint and Igel 2002). However, this term has also been used to describe algorithms that use adaptation techniques that do not encode the adapted control parameter onto the chromosome (Beyer and Deb 2001) (Espinoza et al. 2001). Algorithms that utilise the evolutionary metaphor for adapting the control parameters but using a subpopulation or more for the control parameters, in addition to the subpopulation of the problem variables, are referred to as a co-evolutionary self-adaptive algorithms (Tuson and Ross 1998) (Eiben

1999). An algorithm can be described as a co-evolutionary self-adaptive algorithm even in the case where the control parameters and the problem variables are encoded together in a single chromosome, but using different operators or operators' parameters for each set (Tuson and Ross 1998). In this dissertation, the term self-adaptive will be used to describe evolutionary self-adaptive algorithms in accordance with Eiben et al. (1999) (see above), who provided a comprehensive classification of adaptation in evolutionary algorithms.

The control parameters to adapt can be parameters associated with a specific genetic operator such as the mutation rate (Bäck 1992), the number and the location of the crossover points (Schaffer and Morishima 1987), and the number of genes that group mutation will rebuild in a cutting stock problem (Hinterding 1997). They can also be probabilities of using alternative operators (Davis 1989) (Julstorm 1995) (Spears 1995) (Hinterding 1997) (Tuson and Ross 1998).

Tuson and Ross (1998) imposed three conditions for efficient adaptation of the probabilities of using alternative operators. These conditions include the use of suitable metric to evaluate operator performance. They also insisted upon the importance of a clear link between the values of the genetic settings being adapted and the search's performance. According to Bäck and Schütz (1996), the limited success of the self-adaptive crossover operator (Schaffer and Morishima 1987) is due the weak impact of a particular crossover operator on the fitness of an individual. The third imposed condition is that the benefits by the adaptation process should outweigh its cost.

6.2 Adaptation in hybrid algorithms

The distribution of the genetic-local hybrid's time between the global search method and the local search method influences the efficiency and the effectiveness of the search process. Finding an optimal division of an algorithm's time is one of the difficult tasks that the designers of hybrid genetic algorithms face. Different techniques have been proposed to use the hybrid's time efficiently, as mentioned in chapter 2. These techniques aimed to attain an optimal utilisation of hybrid's time through controlling the local search control parameters. The problem of striking a balance between a global genetic algorithm and a local search method can be viewed as a problem of finding optimal control parameters of a local search within a hybrid. The different control parameters adaptation techniques, mentioned in the previous section, can be used to adapt the parameters of a local search.

Lobo and Goldberg (1997) viewed the problem of deciding between the global genetic tool and the local search tool as a two bandit problem where the payoff of each bandit is unknown and changes with time. They suggested that a probability matching approach, which has been used to adapt the operators probabilities based on their recent performance, can be used to decide on the proper tool at any time. In their experiments, an elitist selecto-recombinative genetic algorithm was combined with a hill-climb method to solve the oneMax problem. They concluded that their adaptation mechanism led to performance slightly inferior to that obtained by carefully tuned control parameters. Magyar et al. (2000) used the operator productivity technique to select an operator from eight recombination and local search operators. Espinoza et al. (2001) proposed a hybrid algorithm uses the coefficient of variation of the fitness function as a feedback from the search process to decide whether it is appropriate to utilise a local search to improve the sampling ability or not. The algorithm uses a deterministic rule to decrease the probability of local search after performing a local iteration. The fitness improvement-cost ratio was also used to decide on continuing the local search or returning to the global search. In their adaptive algorithm, Hacker et al. (2002) used the coefficient of variance of both the fitness and phenotype, which measures the relative homogeneity of the population, as a feedback to decide on switching between global genetic and local search.

Different adaptation mechanisms have been used to adapt the probability of local search in a hybrid. By adjusting the local search probability of each individual of the population, these adaptation techniques also decide on the individuals that should perform a local search. Distribution-based adaptation techniques (Hart 1994) (Joines and Kay 2002) (Martinez-Estudillo et al. 2004) modify the probability of local search based on the distribution of individuals in the population to ensure that only one individual from each basin of attraction in the search space undergoes a local search. However, in fitness-based adaptation mechanisms (Hart 1994) (Espinoza et al. 2003b) (Lozano et al. 2004), the local search probability of each individual is modified based on the relationship of its fitness to the fitness of other individuals in order to bias the local search toward individuals with better fitness. In contrast, the local search potential (LS potential) selection mechanism (Land 1998) adapts the local search probability of each individual based on its ability to use a local search most effectively. The self-adaptation technique, despite of being reported that has been successfully used to decide between different local search methods (Krasnogor and Simth 2001), it has not been used to adapt the local search parameters in hybrids.

Based on the role of a local search method as a complementary tool for the genetic search to enhance its performance, the adaptation of the duration of a local search can achieve a

balance between exploration and exploitation. The evolutionary self-adaptation is one possible mechanism to strike that balance. This can be done by incorporating the duration of local search into the representation of individuals and allowing it to evolve. In addition to the genetic search cooperation with a local search method to find an optimal solution for a problem at hand, the genetic algorithm alone is responsible for finding an optimal duration of a local search in order to optimise the hybrid search for that problem. Implementing the metaphor of the evolution of the local search parameters to adapt the search in this simple way is partially in accordance with the third assumption laid by Tuson and Ross (1998). The implementation cost is low compared with other adaptation techniques.

In order to investigate the evolutionary self-adaptation mechanism and the possible ways of improving the hybrid adaptation ability, the capability of the productivity metric to produce an efficient and effective search is discussed in the next section. The factors that may affect this ability will also be examined. The question of whether this form provides a suitable metric to evaluate local search operator performance in the context of the hybrid performance will be addressed.

6.2.1 Duration of local search and self-adaptation

The ability of genetic search to find favourable parameter settings for pure genetic algorithms has been proven (Eiben et al. 1999). However, the ability of the global genetic algorithm to self-adapt the control parameters of a hybrid, especially those related to incorporating a local search method, may require further investigation. In this section, the influence of simultaneously exploring both the problem search space and the control space of local search duration on the hybrid's performance is analysed. This analysis can help to gain some insight into the consequences of the self-adaptation on optimising the division of the search time between global and local search and on the implicit adaptation of the hybrid to the fitness function topology.

The duration of a local search is defined as the number of the consecutive local search iterations that are performed on a solution before terminating a local search procedure and returning to the global genetic algorithm. The duration of a local search has a clear impact on the solution fitness improvement. The fitness of a solution is improved in proportion to the duration of the local search method applied. However, if that duration is more than that needed to reach a local optimum, a part of the search time is wasted. The influence of the duration of a local search on the individual's fitness depends on the solution's location in the fitness landscape which depends on the previous search iterations. Even in the case of

improving the solution fitness, the use of long durations can waste the algorithm's time. This can occur if a local search method is applied on a solution which is in the basin of attraction of a local optimum and not the global one.

There is a link between the duration of the local search and the individual's fitness. This link depends on the fitness function topology, the details of the local search method and the genetic algorithm's setup. By allowing the duration of the local search parameter to evolve by means of genetic operations just like the input variables do, the link between favourable duration of the local search and the individual's fitness can be exploited. Genetic operations can adaptively control the duration of the local search method in order to optimise the individuals' fitness. In this way, it can define the link between the control parameter and the individual's fitness.

However, the ability of defining a link between the duration of local search and the individual fitness can face some difficulties when combined with the pure Baldwinian learning strategy. The acquired fitness of an individual is the sum of the improvements introduced by applying a local search method for the encoded duration and the innate fitness. The hindering effect associated with this learning approach can direct the search towards individuals with long durations of local search and a small innate fitness. The search process, in this case, is degraded from optimising the fitness function to optimising a single control parameter. The possibility of leading the search to this direction increases as the number of variables of the fitness function increases since it may be easier for the algorithm to optimise a single control parameter than optimising a large number of function variables. In addition to the possibility of guiding the search in the wrong direction, the hindering effect can waste the algorithm's time as it can direct the individuals to perform useless local search iterations with long durations. The use of the acquired fitness as a metric to assess the quality of solutions when combined with the pure Baldwinian learning strategy can produce an algorithm with poor performance in terms of solutions quality and convergence speed.

The hindering effect can be aggravated using long durations of a local search (chapter 3). However, the use of a local search with very short durations and/or small probabilities can help to alleviate this problem. In the case of encoding the duration of the local search for self-adaptation, the use of very short durations as a unit of the duration of local search, and restricting the values of the number of local iterations parameter to very small numbers, can help to combat the hindering effect. In this way, the problem's consequences on the ability of the global genetic algorithm to define a link between the duration of local search

parameter and the individual fitness in the direction of optimising the fitness function can be alleviated.

However, the ultimate solution for the hindering effect problem is to rely on innate fitness and not acquired fitness to decide between solutions of equal acquired fitness values. One possible way to do this is to employ the number of local function evaluations, used to acquire the current fitness starting from the innate fitness, in addition to the acquired fitness. Since the number of local iterations parameter, which is a good indication of the number of function evaluations consumed in a local search, is already encoded into the individual, it can be used together with the acquired fitness to direct the search towards solutions of high quality.

Assuming that the global genetic algorithm is able to define a link between the control parameter values and an individual's fitness in the direction of optimising the solution quality, the question becomes is defining this link enough to improve the hybrid search's performance and if so in which way the performance can be enhanced? Reviewing the impact of the duration of the local search on the search's time utilisation together with a brief comparison between self-adaptation in the pure genetic algorithms and the hybrid genetic algorithm may help to answer this question.

As shown in chapter 3, the duration of local search can influence the division of the search time between a global and a local search. The interactions between local search duration, the learning strategy, the fitness topology and other genetic components have a great impact on search time utilisation (Hart et al. 2000) (Rosin et al. 1997). Allocating more time for a local search through using long local search durations can improve the performance when combined with specific learning strategies on specific classes of fitness functions. On the other hand, allocating more time for the global genetic algorithm through using short durations of local search can improve the performance if combined with other learning strategies or used to solve other classes of problems. Assigning the task of finding an optimal duration of local search to the global genetic algorithm means allowing the evolution process to decide on the optimal utilisation of the search time. In the rest of this section, the appropriateness of the individuals' fitnesses as a metric for selecting individuals and its ability to optimise the utilisation of the hybrid's time are discussed.

The evolutionary self-adaptive algorithms encode the control parameters into the chromosomes and use the individual fitness as feedback to assess the suitability of their values for solving a given problem in an effective and efficient way. Good control

parameter values will probably help individuals to produce superior offspring and these will probably have more chances to survive and to propagate.

The genetic algorithms use the fitness as a metric to assess the quality of solutions and the speed of reaching that quality. Since all individuals in the pure genetic algorithm consume an equal number of function evaluations to achieve their fitness, the fittest individual is also the fastest one in achieving that fitness. The use of this metric is fair to assess the quality and speed of the solutions at the same time. For this reason, relying on the individual fitness is enough to choose efficient and effective solutions. Control parameter values that lead to high quality solutions also produce these solutions in an efficient way. The propagation of good encoded control parameters that help to produce high quality solution can help to produce an efficient search algorithm at the same time.

A selective use of a local search method within a hybrid means that solution can consume different numbers of function evaluations. Applying a local search method with different local search durations or different local search probabilities for each solution, as in the case of self-adapting the duration or the probability of local search control parameters, are examples of this selective use of local search. Self-adapting such control parameters has a clear impact on the number of function evaluations consumed by each individual. Consequently, selecting individuals based on their fitness only in these algorithms can bias the search towards an effective algorithm and cannot guarantee its efficiency. On the other hand, selecting solutions based on the speed of convergence can lead the search towards a local optimum instead of the global one. However, since the global optimisation's priority is the effectiveness of an algorithm, hybrid genetic algorithms are discriminating on fitness basis rather than speed basis.

In non-adaptive hybrid algorithms, to reduce the possibility of producing inefficient algorithm, the hybrid's practitioners usually prefer to reduce the cost of the incorporating the local search method through using either small local search probabilities (Hart 1994) (Rosin et al. 1997) (Land et al. 1997) (Morris et al. 1998) (Hart et al. 2000) (Espinoza et al. 2001), small local search durations (Hart 1994) (Rosin et al. 1997), or local search methods with low cost compared to a genetic search iteration (Radcliffe and Surry 1994). The same direction can also be followed in the self-adaptive hybrids. Through employing a low cost local search method, or using very short durations as a unit of the duration of local search and restricting the values of the number of local iterations parameter to very small numbers, this efficiency problem can be combated.

6.3 The Algorithm

In the proposed hybrid algorithm, the number of local search iterations is incorporated into the representation of an individual. Through this parameter, the duration of a local search is controlled. It defines the number of local iterations that should be performed by the associated individual. The global genetic algorithm evolves the number of local search iterations while the hybrid is using that control parameter to optimise the fitness function variables. Through adopting the evolutionary self-adaptation metaphor, the algorithm allows the global genetic algorithm to dynamically decide on the individuals that should perform a local search. It also decides on the duration of the local search method through modifying the number of local iterations as it cooperates with the local search to solve a given problem. This can facilitate the adaptation of number of local search iteration's control parameter without exogenous control.

In general, the control parameters in the evolutionary self-adaptive algorithm can be adapted either at the individual level (i.e. local level) or at the population level (i.e. global level). In the local adaptation, the control parameter is applied to the associated solution only. In contrast, the control parameter in the global adaptation is tied to the population as a whole, and not to a particular solution. The number of local iterations of an individual is computed by taking the average of the number of local iterations of the individuals of the whole population. Local adaptation is used in the proposed algorithm because it is reasonable to assume that different individuals are following different paths through the search space. It is also proven that local adaptation outperforms global adaptation (Spears 1995).

In the proposed self-adaptive hybrid algorithm, after performing a genetic iteration, the number of local iterations associated with each solution is extracted from the chromosome's structure. Depending on the value of that parameter, a number of local search iterations are performed on that solution. If the value of that parameter is zero, no local search iteration will be performed. Otherwise the specified number of local iterations will be performed consecutively. Using the learning strategy specified by the algorithm, the resulting solution is mapped back to the mating pool. Pseudo code for the Self-Adaptive local-search-Duration Hybrid (SADH) algorithm is shown in Figure 6.1.

The maximum value of the number of local iterations in this algorithm was set to three. The first reason for selecting this value is the expected benefits of using small durations of local search as illustrated in the previous section. The second reason is to allow comparing the

adaptive ability of this algorithm with the adaptive staged hybrid (ASH) algorithm, which was proposed by Espinoza et al. (2001) and uses a maximum of three local iterations.

```
Begin
   t = 0
   initialise (Population(t))
   evaluate (Population(t ))
   while termination criteria not satisfied
   Begin
      t = t + 1
      MatePool(t) =select(Population(t-1))
      MatePool(t) =crossover (MatePool(t))
      MatePool(t) =mutate(MatePool(t))
      evaluate( MatePool(t) )
      For each Chromosome(i) of MatePool(t)
      Begin
         LocalIterations=getNumberIterations(Chromosome(i))
          if LocalIterations> 0 then
             Solution(i)=mapToPhenotype(Chromosome(i))
             for k=1 to LocalIterations
             Begin
                Solution(i)=performLocalSearchIteration(Solution(i))
              End
               Chromosome(i)=mapToChromosome(Solution(i),LearningStrategy)
         End
      End
      Population(t)=merge (MatePool(t), Population(t-1))
   End
End
```

Figure 6.1: The Self-Adaptive local-search-Duration Hybrid (SADH) Algorithm.

The algorithm also makes use of the number of local iteration's control parameter, which already exists within the chromosome, to discriminate between innate and acquired fitness. In a case of an equal fitness, the algorithm chooses the individual with the smaller value of local search iterations since its acquired fitness is closer to the innate one. This can help to alleviate the consequences of the hindering effect associated with the Baldwinian approach.

6.4 Test functions

In order to evaluate the capability of the proposed SADH algorithm for finding high quality solutions for difficult optimisation problems in a general way, a set of test functions has been selected. The test functions have also been used to evaluate some of the assumptions that have been made regarding the behaviour of the SADH algorithm and the possible ways of improving its performance.

The members of the selected set of test functions have been designed by several authors for analysing and comparing different optimisation algorithms. Most of the test functions have especially been designed to detect weak points of the different optimisation algorithms. These functions have a number of features that are known to be hard for optimisation and believed to be present in many real-world problems. Among these features is that the degree of difficulty of these functions can by scaled up by increasing the dimension of the search space due to the increasing number of local optima. According to Whitley et al. (1995), the presence of scalable functions is essential in any evolutionary test suite. Whitley et al. also insisted on that the test suite should contain non-linear and non-separable test functions. These types of functions are also represented in the test functions set. The test functions suite includes the ellipsoidal (Deb et. al 2002), the Rastrigin (Törn and Zilinskas 1989), the Schwefel (Mühlenbein et. al 1991), the Griewank (Mühlenbein et. al 1991) and the Rosenbrock (De Jong 1975) test functions.

The n-dimensional inverted ellipsoidal function (Deb et. al 2002) is defined as

$$f(x) = -\sum_{i=1}^{n} i x_i^2 \qquad \text{For } 100 \le x_i \le 100 \tag{6.1}$$

The ellipsoidal function has a unique global optimum of zero which is located at x=(0, 0, ...). The ellipsoidal is a uni-modal function with different weights for each variable. This can serve to test a badly scaled fitness functions (Ballester and Carte 2004).

The n-dimensional inverted Rastrigin function (Törn and Zilinskas 1989) (Mühlenbein et al. 1991) is defined as

$$f(x) = 10n + \sum_{i=1}^{n} x_i^2 - 10\cos(2\pi x_i) \qquad \text{For } -5.12 \le x_i \le 5.12 \tag{6.2}$$

The global optimum of the inverted Rastrigin function is zero at x=(0, 0,...). This function is characterised by the existence of many local optima whose values are proportional to their distance from the global optimum. The local optima are located at a rectangular grid with a size of one. Grid points with x_i =0 except one coordinate, where x_j=1, give the second best optimum with $f(x)$=-1. With increasing distance from the global optimum the fitness values of local optima become smaller. There is no correlation between the variables of the Rastrigin function.

The definition of the n-dimensional inverted Schwefel function (Mühlenbein et. al 1991) is

$$f(x) = 418.98288n + \sum_{i=1}^{n} -x_i \sin(\sqrt{|x_i|}) \qquad \text{for } -500 \le x_i \le 500 \tag{6.3}$$

The global optimum of zero is at the point x=(420.9687, 420.9687, ...). The interesting characteristic of this function is that the second best optimum is located far away from the global optimum. This can trap the optimisation algorithm on a local optimum. In this function, there is no correlation between the variables.

The inverted Griewank function (Mühlenbein et. al 1991) can be defined as

$$f(x) = 1 + \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} \cos(\frac{x_i}{\sqrt{i}}) \qquad \text{for } -30.0 \le x \le 30.0 \tag{6.4}$$

The global optimum of zero is located at the point x=(0, 0, ...). There are many local optima in the landscape of this function. The product term introduces a correlation between the function variables "epistasis". This can disrupt optimisation techniques that work on one function's variable at a time. The increase in the number of variables decreases the number of local optima since it makes the function surface flat.

The n-dimensional inverted Rosenbrock function (De Jong 1975), also known as banana function, is defined as

$$f(x) = -\sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2] \quad \text{for } -2.048 \le x_i \le 2.048 \tag{6.5}$$

The unique global optimum of the inverted Rosenbrock function is zero which is located at x=(1, 1, ...). There is only one peak in the landscape of the two dimension function. The global optimum of this function is inside along, narrow, parabolic shaped flat valley. This extremely flat region around the global optimum makes it difficult for search algorithms to locate the global optimum. There is evidence showing that the landscape of this function with a dimension of more than three contains several optima. Deb et al. (2002) identified three local optima with fitness values 0, 3.98662 and 65.025362 for 20-dimensional function. This shows that the difficulty of this function can by scaled up by increasing its dimension. There is also a correlation between its variables, which makes it a non-separable function.

6.5 Experiments

The SADH algorithm and the ASH algorithm were used to solve the set of test functions in order to evaluate the performance of the proposed algorithm and gain some insight into its self-adaptation behaviour. Some members of the test function set were selected to evaluate some ideas to improve the search performance.

6.5.1 Evaluating the performance of the self-adaptive hybrid algorithm

For the purpose of evaluating the SADH algorithm, its performance was compared with the pure genetic algorithm and the ASH algorithm on a selected set of test functions. The quality of the solutions produced by each algorithm was used as a main measure of the algorithm's performance. The percentage of experiments that converged to the global optimum was used as an indication of the ability of these algorithms to produce high quality solutions. The performance was compared using different population sizes and different learning strategies in order to evaluate the ability of these algorithms to adapt to different search environments.

The results of optimising the test functions using the proposed SADH algorithm were evaluated against the results obtained by the ASH with initial local search probabilities of 0.0, 0.1, 0.2, and 0.99. The ASH algorithm with an initial local search probability of 0.0 is identical to the pure genetic algorithm. An initial local search probability of 0.1 was selected because Espinoza et al. claimed that using this value produces the best hybrid performance. This can be true for simple test functions using large population sizes, as in the case of the fitness functions they reported on, but for more complicated test functions, using other values can produce better results. The same values of the other meta-parameters suggested by Espinoza et al. were used in these experiments.

The set of algorithms has been used to optimise the test functions suite for dimensions of 2, 10 and 20 and for different population sizes. For 2-dimensional test functions, the population sizes tested were 20, 40, 60, 80 and 100. The 10-dimensional functions tested with population sizes of 50, 100, 150, 200 and 250. However, population sizes of 100, 200, 300, 400 and 500 were used to optimise the test functions with 20 variables. Each of these variables was represented by a string of 16 bits.

The hybrid algorithms were tested using different learning strategies. In addition to the pure Baldwinian and the pure Lamarckian approaches, the 25% partial Lamarckian (i.e. 75% partial Baldwinian), the 50% partial Lamarckian (i.e. 50% partial Baldwinian), and 75% partial Lamarckian (i.e. 25% partial Baldwinian) approaches were used in these experiments.

The hybrids use the simple elitist genetic algorithm with binary tournament selection, two-point crossover, and simple mutation. For all experiments, the probability of crossover was set to 0.7 and the probability of mutation was $1/N$ where N is the population size. In the

adaptive ASH algorithm, the maximum number of local iterations was 3, e was 0.2, and the local threshold value was 0.6.

The stopping criterion for all experiments was a maximum number of function evaluations. The value of this parameter was set to 1000 times the population size for the functions of two and ten variables for partial or the pure Lamarckian learning approaches. However, for the test functions of 20 variables, this value was set to 5000 times the population size for partial or the pure Lamarckian approaches. For the pure Baldwinian approach, the maximum number of function evaluations was doubled because this type of search is slow and the priority in these experiments is to evaluate the effectiveness of these algorithms. This stopping rule was applied for all the test functions except the ellipsoidal function, where the maximum number of function evaluations was set to 500 times the population size, due to its simplicity. Each experiment was repeated 50 times.

A local search method, which combines the steepest descent method and Brent's method (Press et al. 1993) to estimate the best step size to climb to the local optimum from the current position in the basin of attraction, was used. The steepest descent algorithm uses the derivatives of the fitness function to estimate the best step size to climb to the local optimum from the current position in the basin of attraction. Brent's method fits a parabola to three initial solutions and uses the maximum of the parabola as the next potential solution of the overall function (chapter 2).

The percentage of times each hybrid algorithm found the global optimum using different learning strategies was used as a metric to compare the performance of the hybrids on the different test function used. The number of times each hybrid found the global optimum using the different learning strategies were added together and then divided by the total number of experiments conducted. The resulting percentage of each hybrid was compared with other hybrids and with the percentage of times the pure genetic algorithm found the global optimum.

In experiment 6.1, the pure genetic, the SADH and the ASH algorithms were used to optimise the ellipsoidal function and the percentages times of finding the global optimum were compared. The results of comparing these percentages when solving the 2-dimensional ellipsoidal function show that the pure genetic algorithm performed better than the hybrid algorithms. Actually, the difference in the percentages between the pure genetic algorithm and the two hybrids is due the poor performance of both hybrids when utilising the pure Baldwinian approach. This bad performance is related to the hindering effect,

which leads the search to converge to a point near the global optimum. The closeness of this point to the global optimum depends on the local search method, its duration, and the size of the basin of attraction of the global optimum (chapter 3). The use of small probability of local search helps to alleviate this problem to some extent. The results show that using a small probability of local search with the ASH gives the best performance of the hybrids for all the population sizes used. The nature of fitness landscape and the nature of the local search method, where a small number of local iteration may be enough to map any point in the basin of attraction to the global optimum, explain of the bad performance of the hybrids combined with the pure Baldwinian approach.

The hybrid algorithms that adopted the pure Baldwinian learning strategy were not able to find the global optimum of the ellipsoidal function with ten variables, as shown in figure 6.2. However, in general, the ASH algorithm that used small probabilities of local search outperformed the pure genetic algorithm using a small population sizes in terms of percentage of experiments that converged to the global optimum, and vice versa for large population sizes. The diversity introduced as a result of using local search methods enables small population sizes to evolve to reach the global optimum. The performance of the SADH algorithm on this function is still better than the performance of the ASH algorithm with a local search probability of 0.99. The behaviour of the SADH algorithm when combined with the pure Baldwinian approach was expected since the hindering effect may lead the search towards long durations of local search and bad quality solutions.

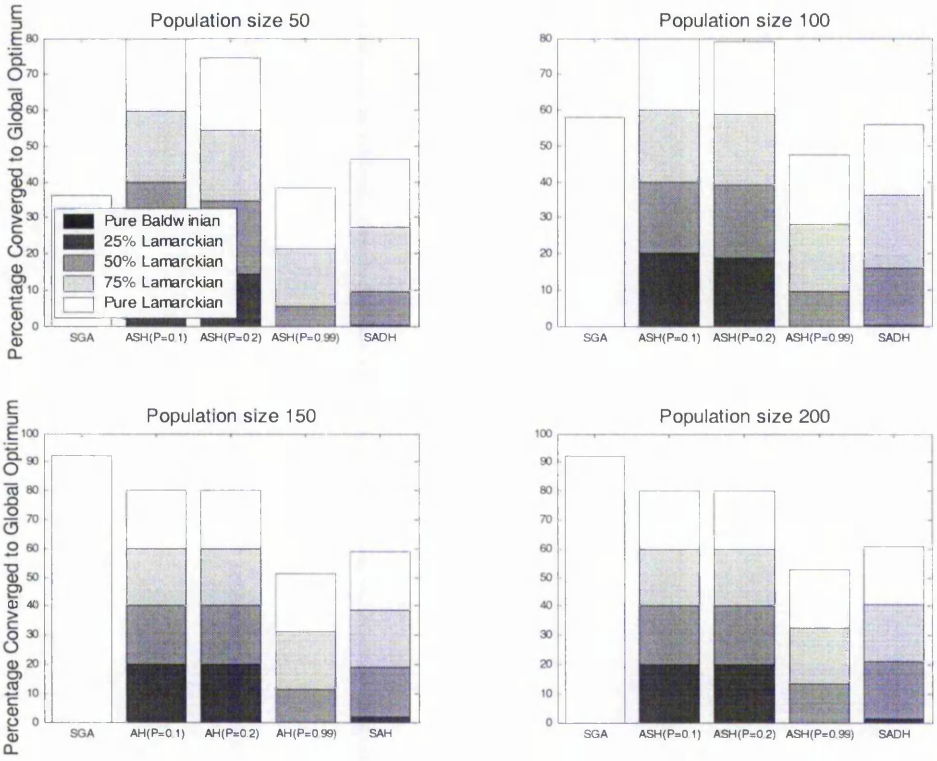

Figure 6.2: Results of Optimising the Ellipsoidal Function with 10 Variables (Experiment 6.1).

Figure 6.3 shows the changes in the number of local iterations of the population over generations for the 10-dimensional ellipsoidal function, when combined with the pure Baldwinian approach, for population sizes of 50 and 150. The graph shows that for a population size of 50, the values of local iterations of 1 and 2 were dominating at the end of search. At the start, the population moves towards a longer local durations. Then, when long durations are not improving the fitness any more, the search moves towards shorter durations. At the end of the search, short durations are dominated in order to distinguish innate from acquired fitness. However, even with these short durations of local search and because of the disappearance of the value of zero of this parameter, the algorithm was unable to guide the search to the exact global optimum. The graphs show a similar trend for a population size of 150 where the search moved toward longer durations.

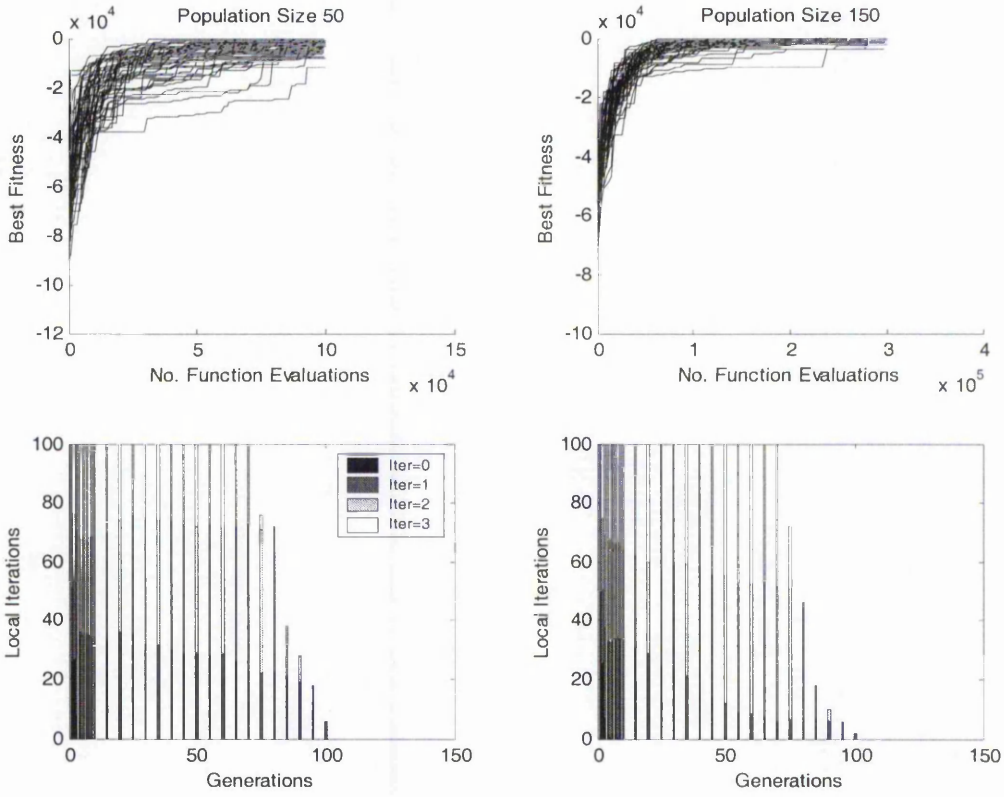

Figure 6.3: Optimising the Ellipsoidal Function Using the Baldwinian Approach (Experiment 6.1).

In figure 6.4, the changes in the number of local iterations and the best fitness over time are shown for the 50% partial Lamarckian approach. The graphs show that, for a population size of 50, a greater part of the population was using two local iterations and that part was fixed until the end of the search. However, with a population size of 150, a greater part of the population was using three iterations. This part increased while the individuals that were using a single local iteration decreased as the search progressed.

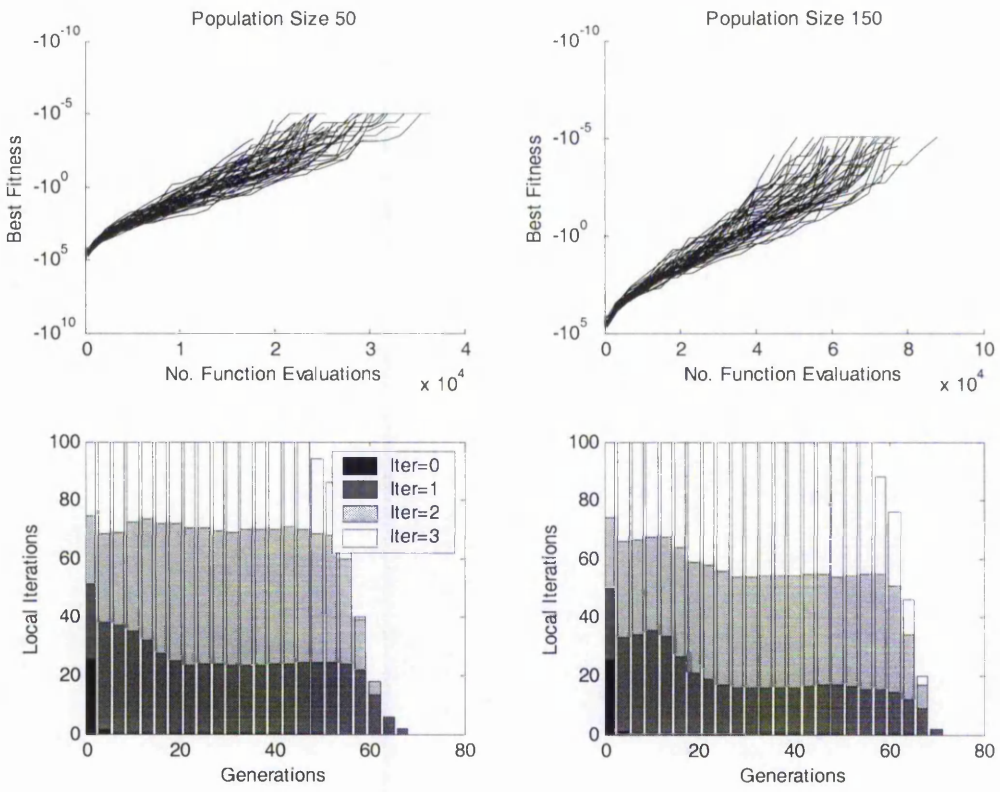

Figure 6.4: Optimising the Ellipsoidal Function Using the 50% Lamarckian Approach (Experiment 6.1).

By comparing the two previous graphs with figure 6.5, which shows the percentage of population that using the available local search iteration for the pure Lamarckian, it is clear that as the search moves towards more partial Lamarckian and larger population sizes, the population moves toward adopting longer durations of local search.

An experiment was conducted to illustrate the impact of the local search method on the performance of hybrids when used to solve the ellipsoidal function. The simple binary hill-climbing algorithm, where a randomly selected bit is flipped, was used as a local search method in the hybrids to solve the 2-dimensional and 10-dimensional ellipsoidal function. The results of experiments demonstrated, not shown here, that the performance of all the hybrids that adopt the pure Baldwinian approach is significantly improved using this local search method. For example, 58%, 86%, 100%, 96% and 100% were the percentages of the experiments that found the exact global optimum incorporating the simple binary hill-climbing method within the proposed algorithm to solve the 2-dimensional ellipsoidal

function compared with zero percentages when utilising the steepest descent method for population sizes of 20, 40, 60, 80 and 100, respectively.

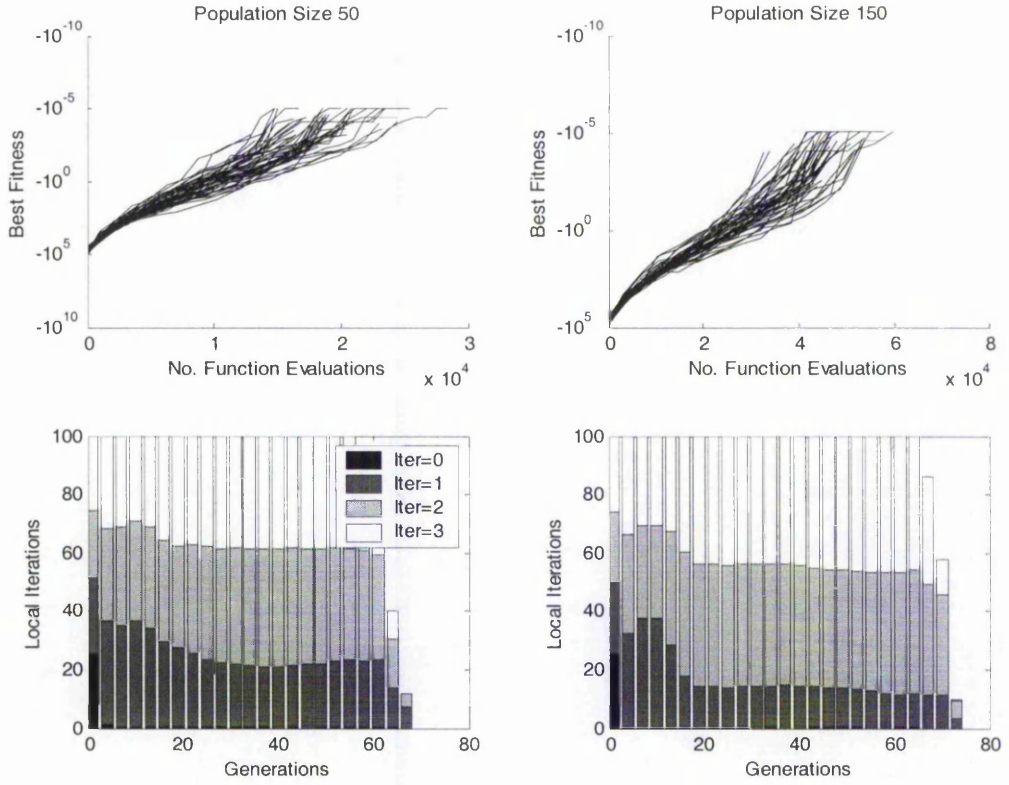

Figure 6.5: Optimising the Ellipsoidal Function Using the Lamarckian Approach (Experiment 6.1).

The results obtained in experiment 6.2, which evaluated the ability of the different algorithms to solve the Rastrigin function with 2, 10 and 20 variables, illustrated that the self-adaptive algorithm produced the third best performance for 2 variables function, the first or the second best for 10 variables function, and the best performance for 20 variables function (figure 6.6). The good performance of the proposed algorithm for 20-dimensional function is expected since the cost of optimising a single control parameter can be neglected compared to the cost of optimising 20 variables, whereas this cost cannot be neglected when optimising a function of 2 variables. The experiments also show that the proposed algorithm faced some difficulties when combined with the pure Baldwinian approach.

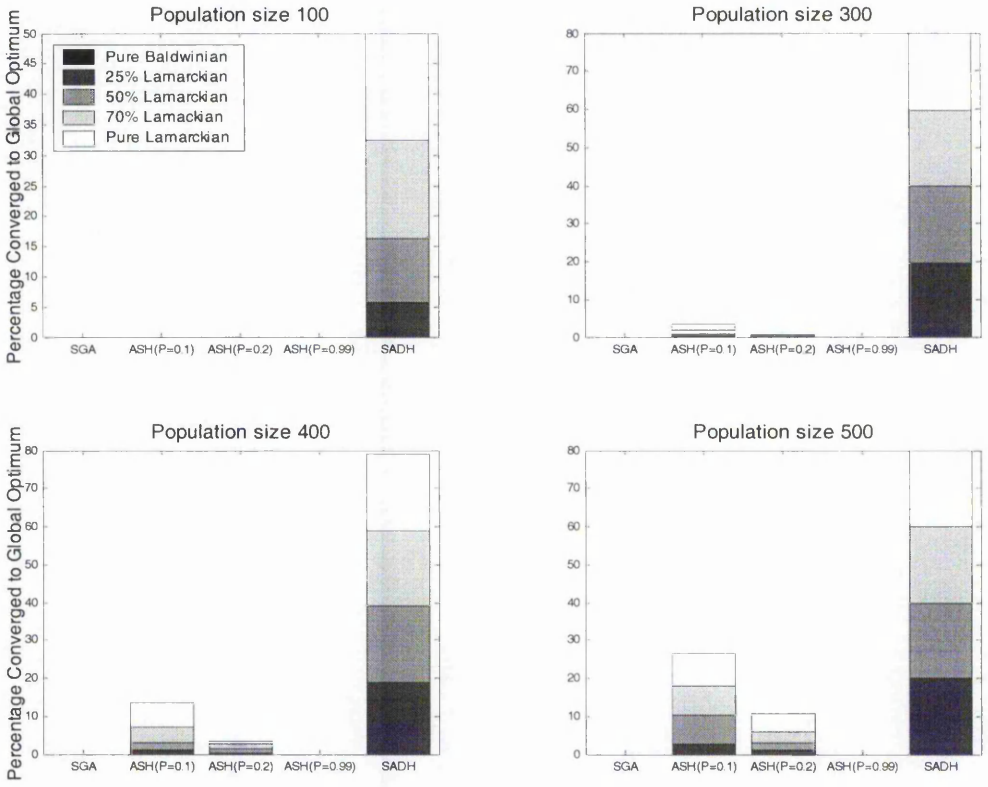

Figure 6.6: The Results of Solving the 20-dimensional Rastrigin Problem (Experiment 6.2).

The change in the percentage of the number of the local search iterations over generations, and the change in the best fitness as a function of the number of function evaluations, for the 10-dimensional Rastrigin function for different learning strategies are shown in figures 6.7, 6.8 and 6.9. These graphs, as the previous graphs, show the disappearance of the 0 value of the control parameter. The effect of this disappearance cannot help algorithms that are biased toward the Baldwinian approach to produce high quality solutions. They also show that increasing the population size can help to combat this problem. These plots show that, as the part of the population that use the Lamarckian approach increases, the trend to use long durations of local search increases.

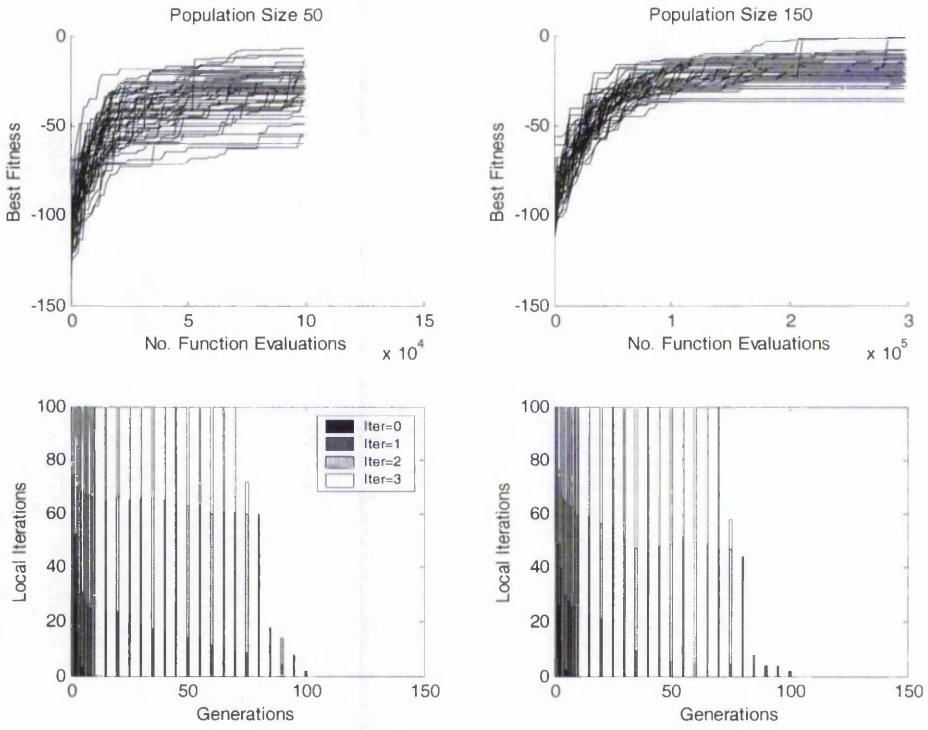

Figure 6.7: Optimising the Rastrigin Function using the Pure Baldwinian Approach (Experiment 6.2).

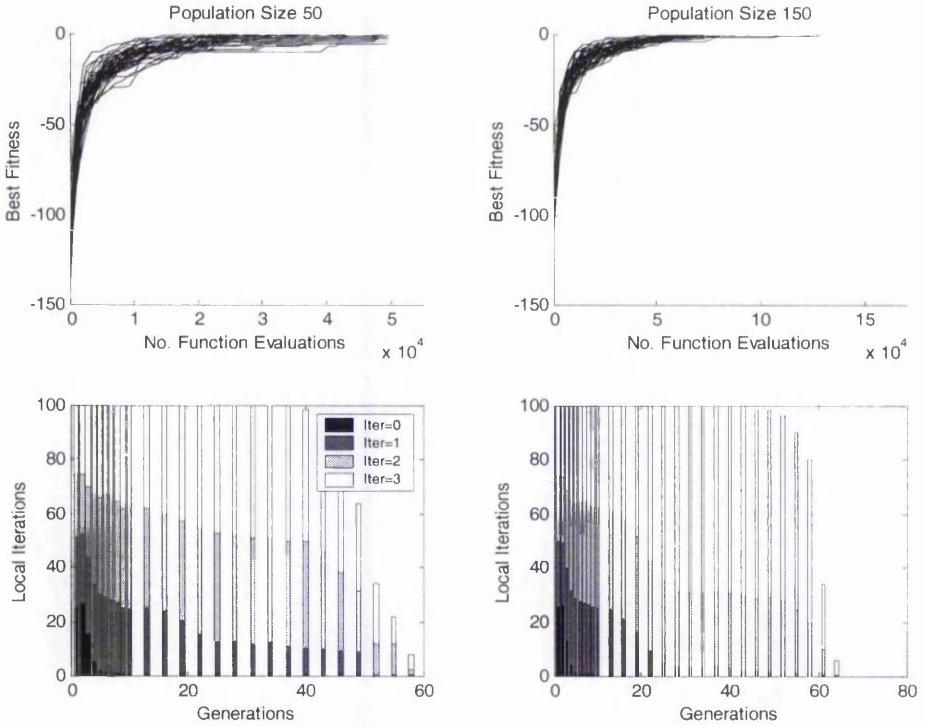

Figure 6.8: Optimising the Rastrigin Function Using 50% Lamarckian approach (Experiment 6.2).

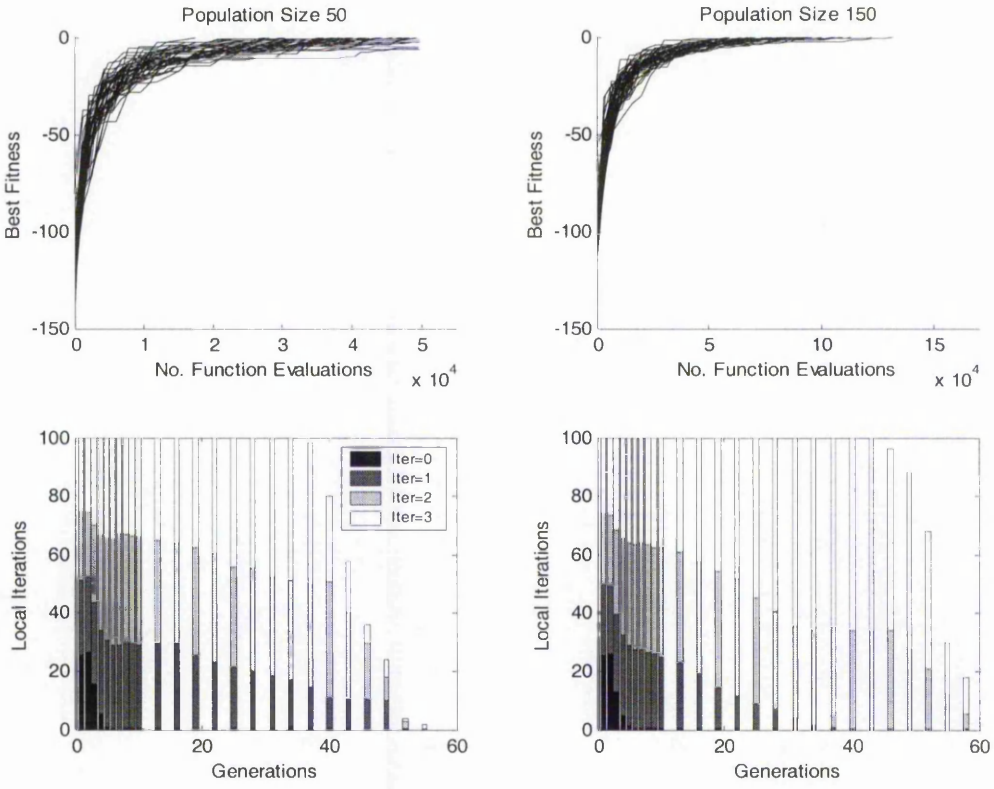

Figure 6.9: Optimising the Rastrigin Function Using the Lamarckian Approach (Experiment 6.2).

The results for solving the 2-dimensional Schwefel function using the ASH algorithm in experiment 6.3 show that the optimal probability of local search depends on the population size. The results also showed that the SADH algorithm produced the second or the third best performance. They also showed that the SADH algorithm performs poorly with the pure Baldwinian approach.

When the algorithms were used to solve the 10-dimensional Schwefel function, the percentage of times that each algorithm managed to find the global optimum is depicted in figure 6.10. The graphs show that the SADH algorithm and the ASH algorithm with a local probability of 0.99 produced the best performance. However, both algorithms produced the worst performance when utilising the pure Baldwinian learning strategy.

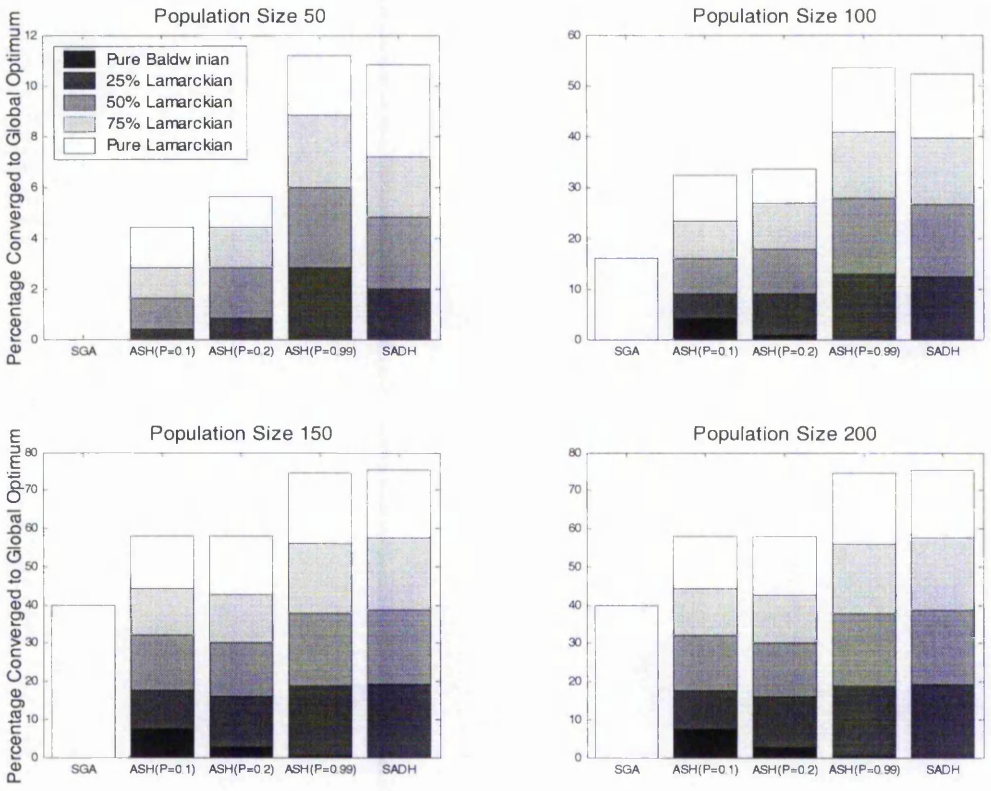

Figure 6.10: The Results of Optimising the Schwefel Function with 10 variables (Experiment 6.3).

When used to solve the 20-dimensional Schwefel function, the SADH algorithm produced the best or the second best performance when combined with a partial or the pure Lamarckian approach. However, as in the previous experiments, it produced the worst performance when combined with the pure Baldwinian learning strategy.

Figures 6.11, 6.12 and 6.13 show how the number of local iterations parameter evolves during optimising the 10-dimensional Schwefel function for population sizes of 50 and 150. The hindering effect problem makes the algorithm unable to decide on an optimal duration of local search. At the start of the search, using a local search of any duration helps individuals to improve their fitness. This accelerated the disappearance of the zero value of the number of local search iterations control parameter. Since the probability of mutating the control parameter is very small, the chance of restoring that value is very small too. The effect of the disappearance of that value is less significant on the algorithm performance when adopting other learning strategies.

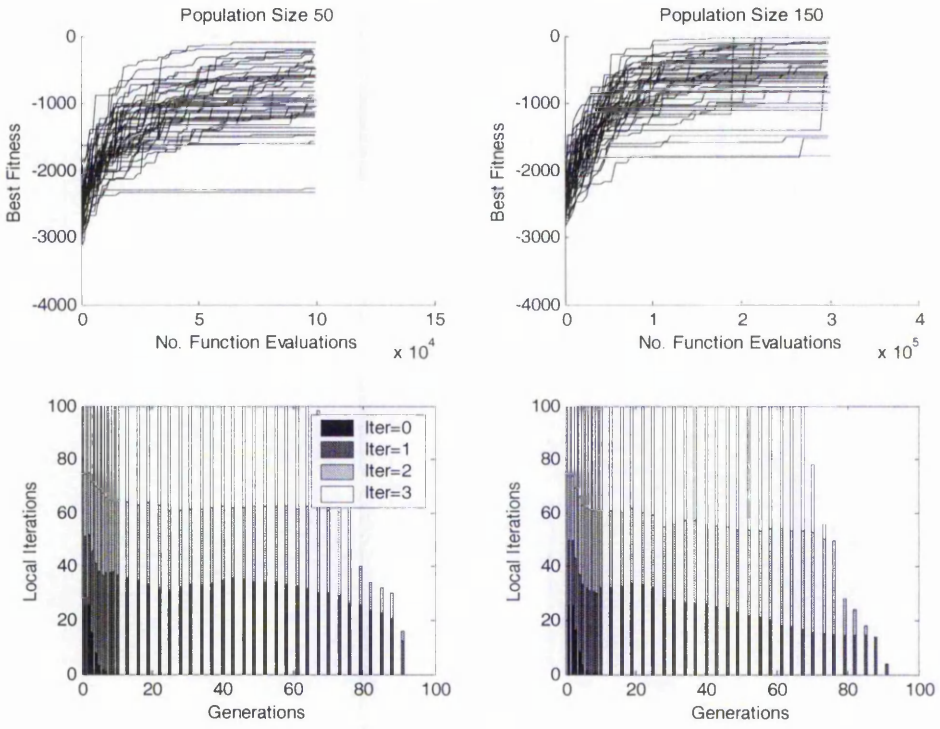

Figure 6.11: The Baldwinian Search and the Schwefel Function (Experiment 6.3).

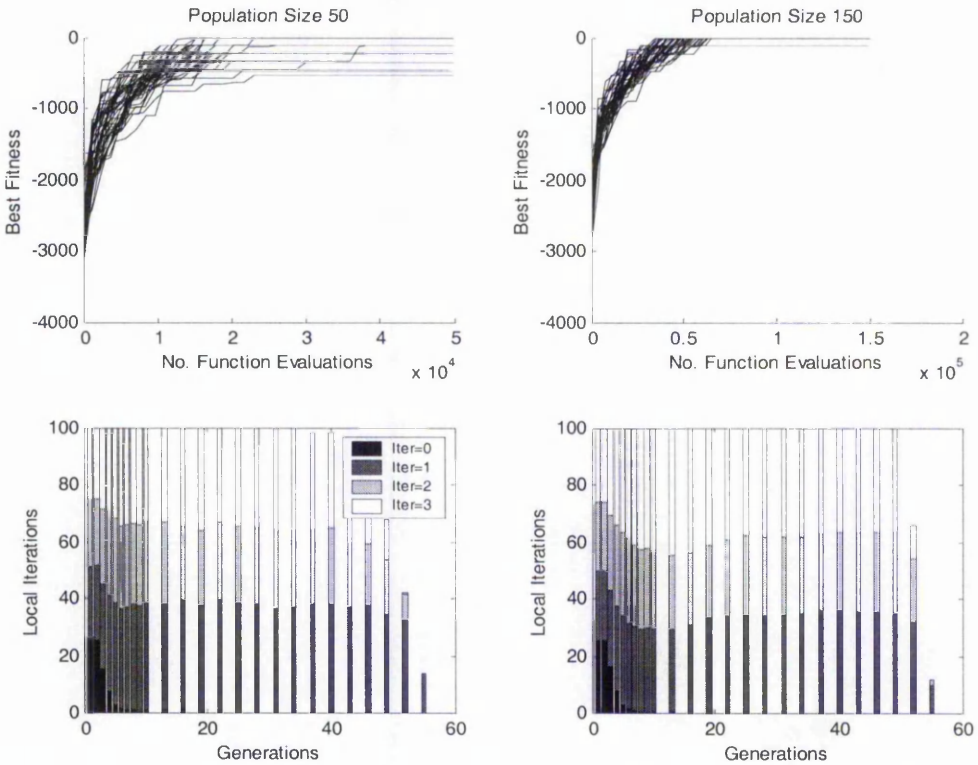

Figure 6.12: Optimising the Schwefel Function Using the 50% Lamarckian approach (Experiment 6.3).

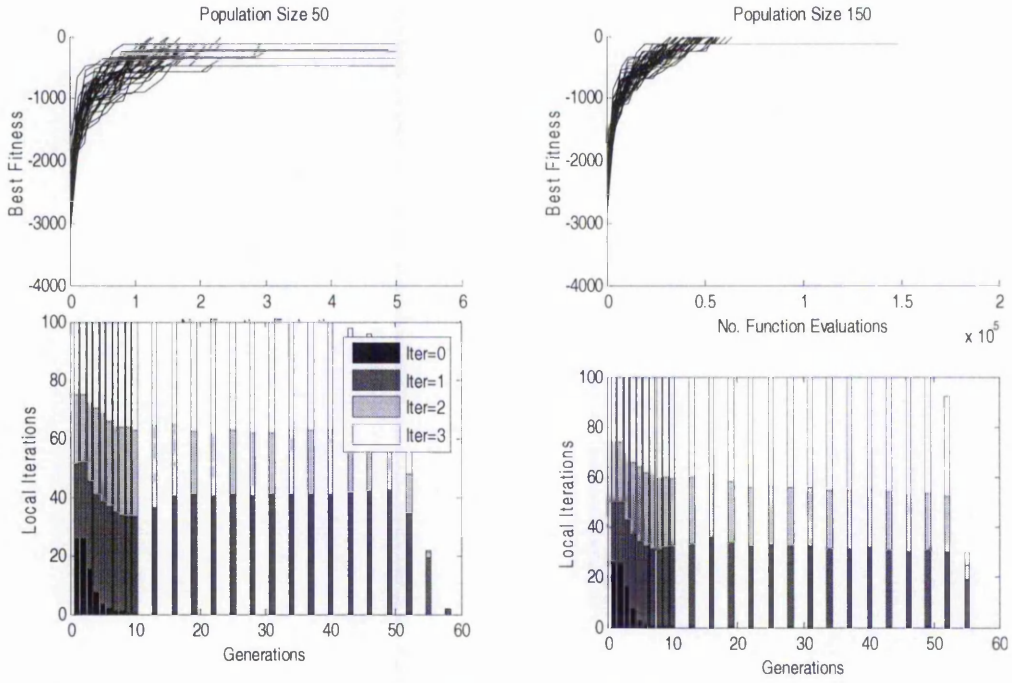

Figure 6.13: Optimising the Schwefel Function Using the Lamarckian Approach (Experiment 6.3).

The results of applying the different algorithms to optimise the 2-dimesnional Griewank function in experiment 6.4 showed that the SADH algorithm produced the best performance for a population size of 100. However, it comes second using a population size of 80 and third using population sizes of 60 and 20. The results also confirmed the bad performance of the proposed algorithm using the pure Baldwinian approach.

All the hybrids showed a similar performance when used to solve the Griewank problem with 10 and 20 variables. Figure 6.14 shows the results of applying the algorithms to the 10-dimensional Griewank function. The hybrids significantly outperform the pure genetic algorithm. However, they showed poor performance when combined with the pure Baldwinian learning strategy.

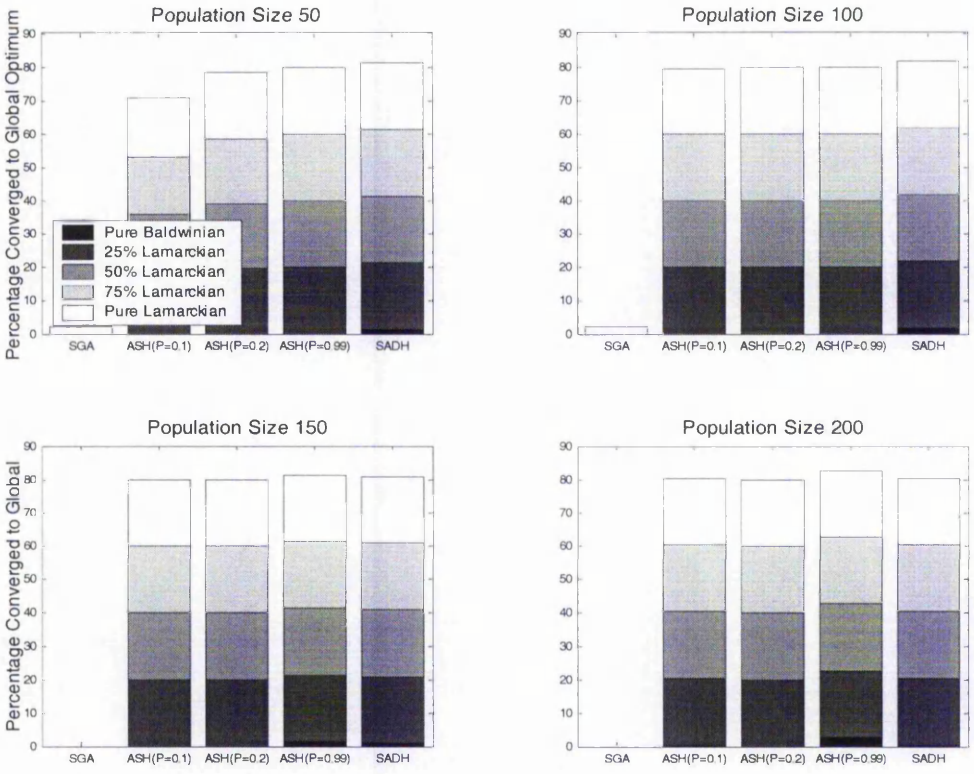

Figure 6.14: The Results of Solving the Griewank Problem with 10 Variables (Experiment 6.4).

Figures 6.15, 6.16, and 6.17 depict how the number of local iterations control parameter evolves with time while solving the 10-dimensional Griewank problem. In figure 6.15, the algorithm discovered that it is beneficial to use long durations of local search at the first generations. This leads the algorithm to favour long durations and after a number of generations, long duration values dominate other values. The values of one and zero disappeared very quickly making the chances of restoring these values, when needed, very small. These graphs show that as the search progressed, the largest part of the population was using the largest available number of local iterations.

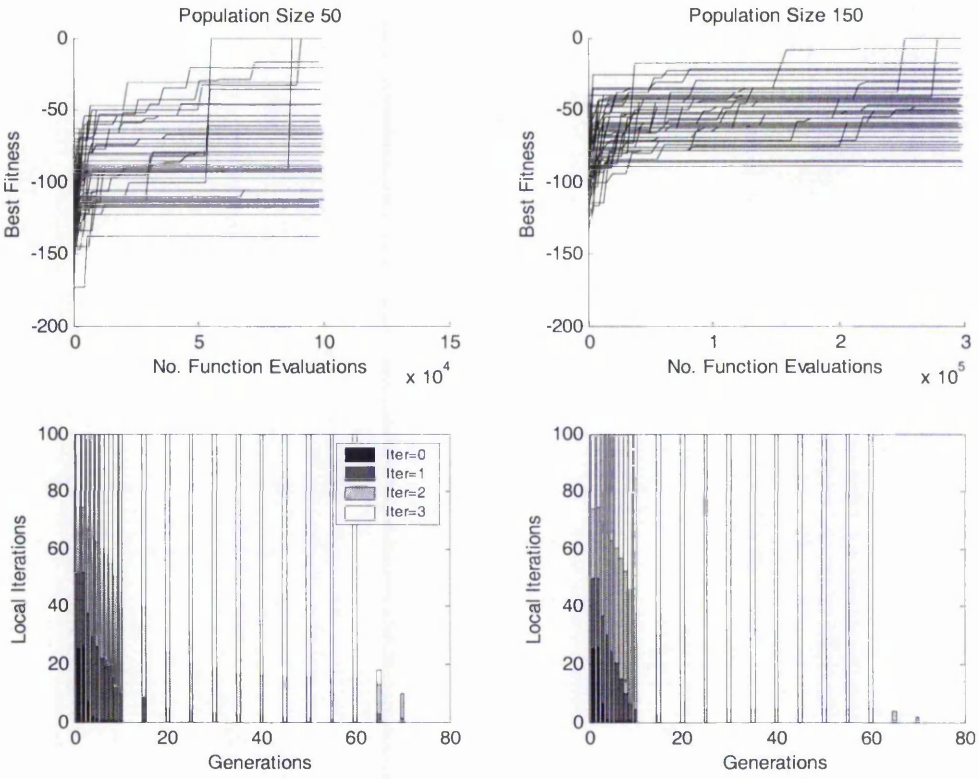

Figure 6.15: Optimising the Griewank Function Using the Baldwinian Approach (Experiment 6.4).

The other two graphs show that the value of 1 for the control parameter propagated until the end of the search. This can be explained by the fact that partial or the pure Lamarckian approaches enable the algorithm to distinguish between individuals on the basis of their genetic structure which is reflected through their fitness. The use of the number of local iterations to discriminate between individuals with an equal fitness helped to establish this in the case of a partial Lamarckian approach. Keeping diversity in the number of local search iteration is useful as it improves the probability of restoring good values that were not useful at previous stages.

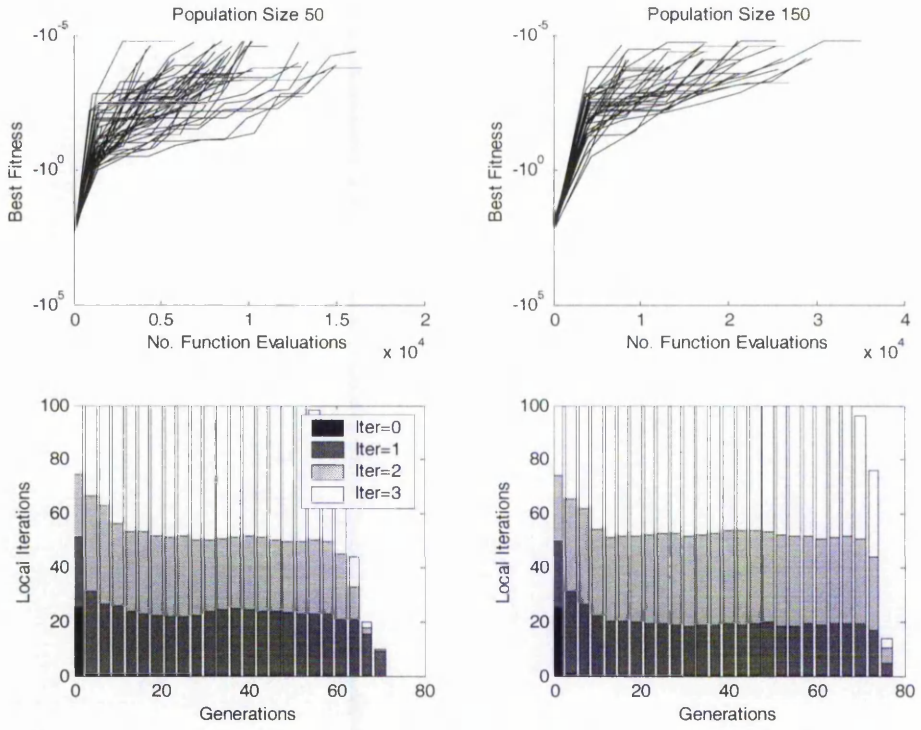

Figure 6.16: Optimising the Griewank Function Using the 50% Lamarckian Approach (Experiment 6.4).

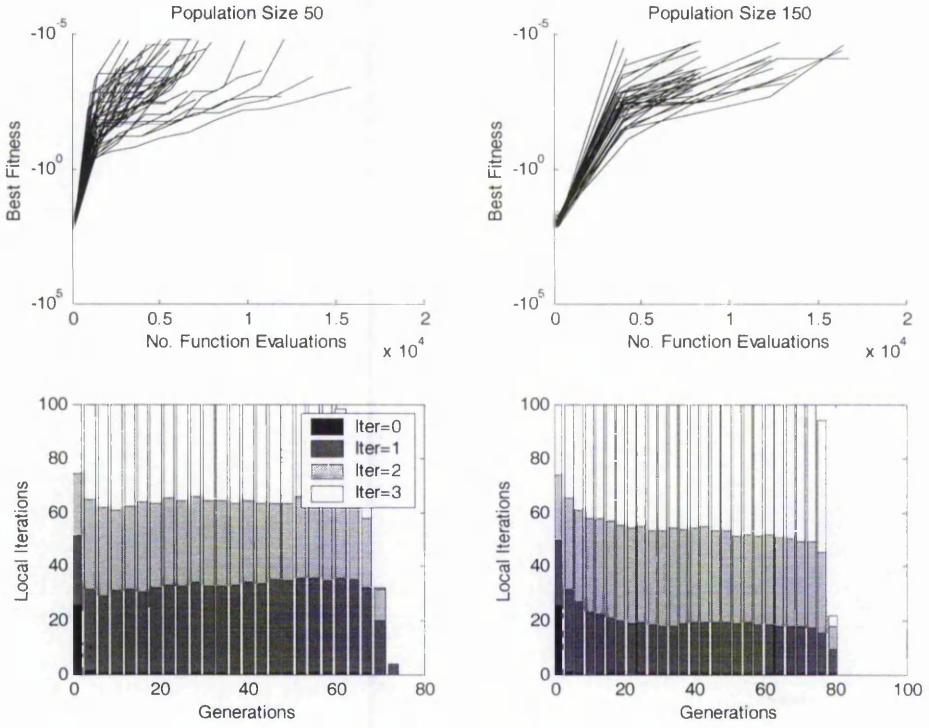

Figure 6.17: Optimising the Griewank Function Using the Lamarckian Approach (Experiment 6.4).

The results of optimising the 2-dimesional Rosenbrock function using the two hybrids in experiment 6.5 demonstrate that the proposed SADH algorithm outperformed the other algorithms for the different population sizes used. However, the performance of the SADH hybrid utilising the pure Baldwinian approach is poor compared with the pure genetic algorithm and the ASH algorithm with small probabilities of local search for most of the population sizes used.

The results for optimising the Rosenbrock function of 10 variables show the supremacy of the SADH algorithm over other algorithms, as shown in figure 6.18. The experiments illustrated that the SADH algorithm with most of its population adopted the Lamarckian approach can find the global optimum more frequently than others with most their population adopted the Baldwinian approach.

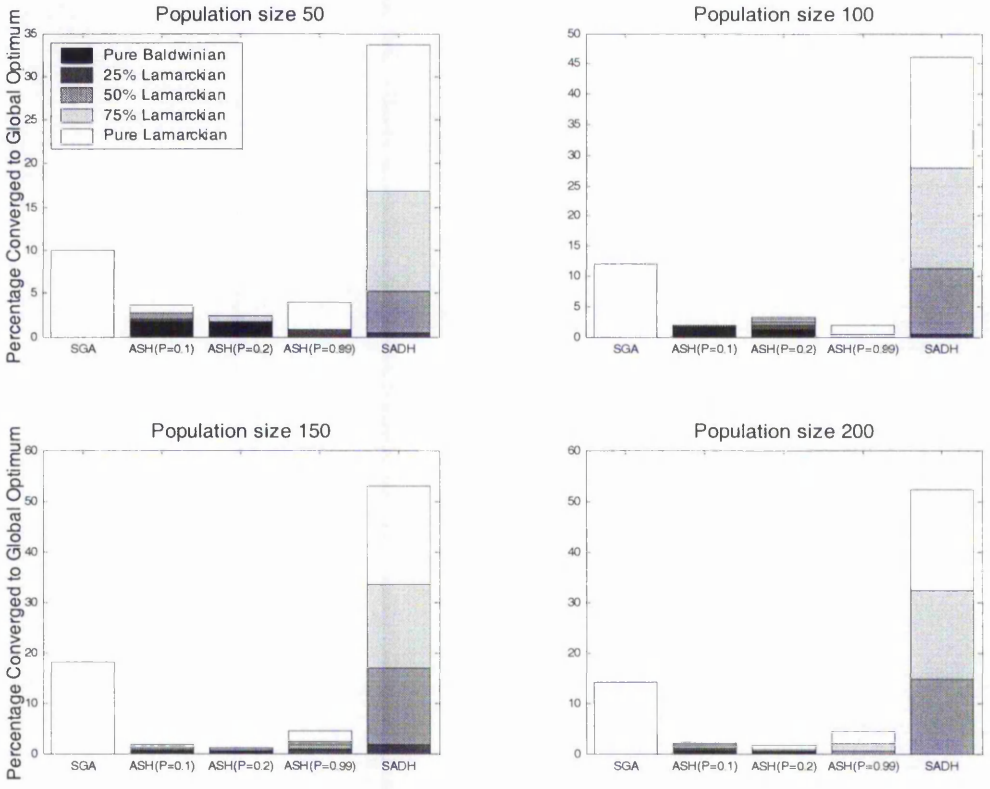

Figure 6.18: The Results of Optimising the Rosenbrock Function with 10 Variables (Experiment 6.5).

The graphs of figure 6.19 show the changes in the percentage of the number of local iterations over generations for the 10-dimensional Rosenbrock function using the pure Baldwinian approach. This figure shows that the values of 0 and 1 of the local search

iterations disappeared quickly. The whole population used either two or three local iterations. The use of the same mutation rate for the fitness function variables and the number of local iterations control parameter reduces the chances of restoring good gene values of the control parameter. The quick disappearance of short durations of local search and the use of the same mutation rate can explain the bad performance of the proposed algorithm when most of its population is using the Baldwinian approach.

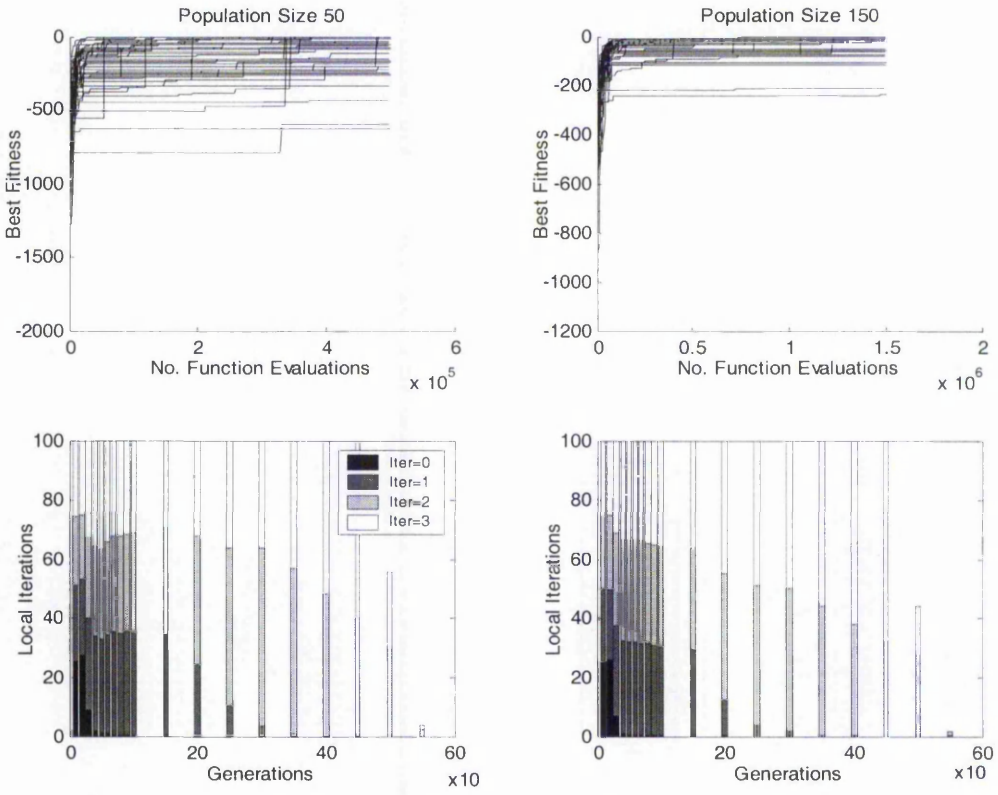

Figure 6.19: Optimising the Rosenbrock Function Using the Baldwinian Approach (Experiment 6.5).

Figure 6.20 shows the results for the optimisation of the same function using the 50% partial Lamarckian. The plots show that small durations of local search disappeared quicker than in the case of the pure Baldwinian as the algorithm discovered that the use of long duration can be beneficial with this learning strategy. Local iterations of a value of two dominate other values.

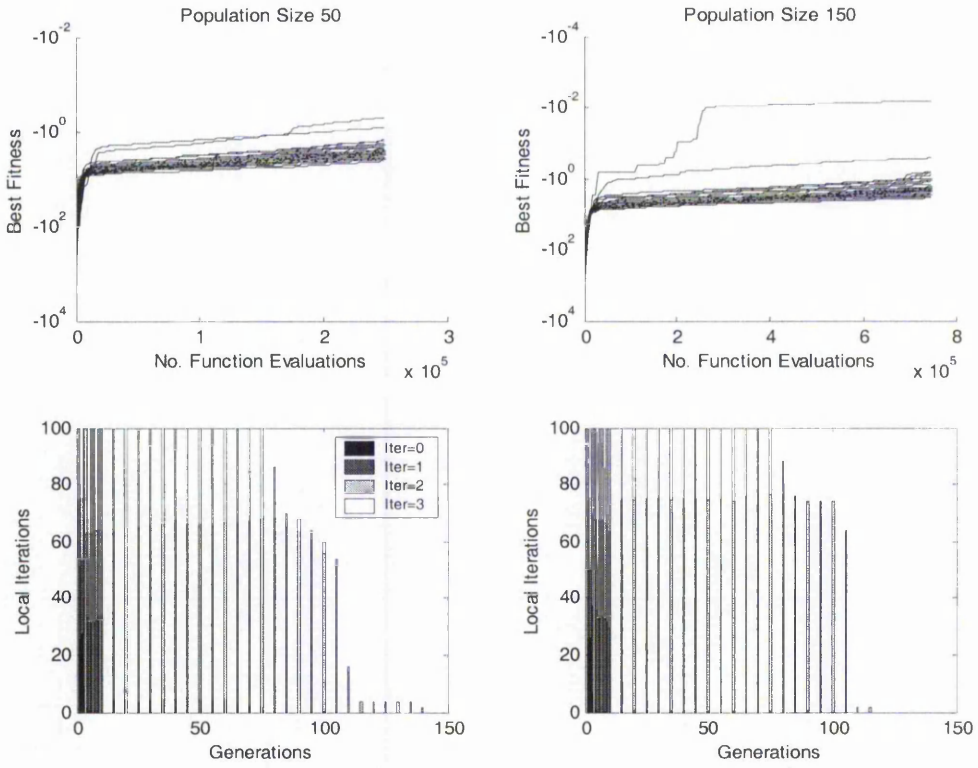

Figure 6.20: Optimising the Rosenbrock Function using the 50% Lamarckian Approach (Experiment 6.5).

In figure 6.21, which shows the change in the number of local iterations control parameter for the pure Lamarckian approach, the value of two dominates the local search values. This trend becomes apparent in a population size of 150.

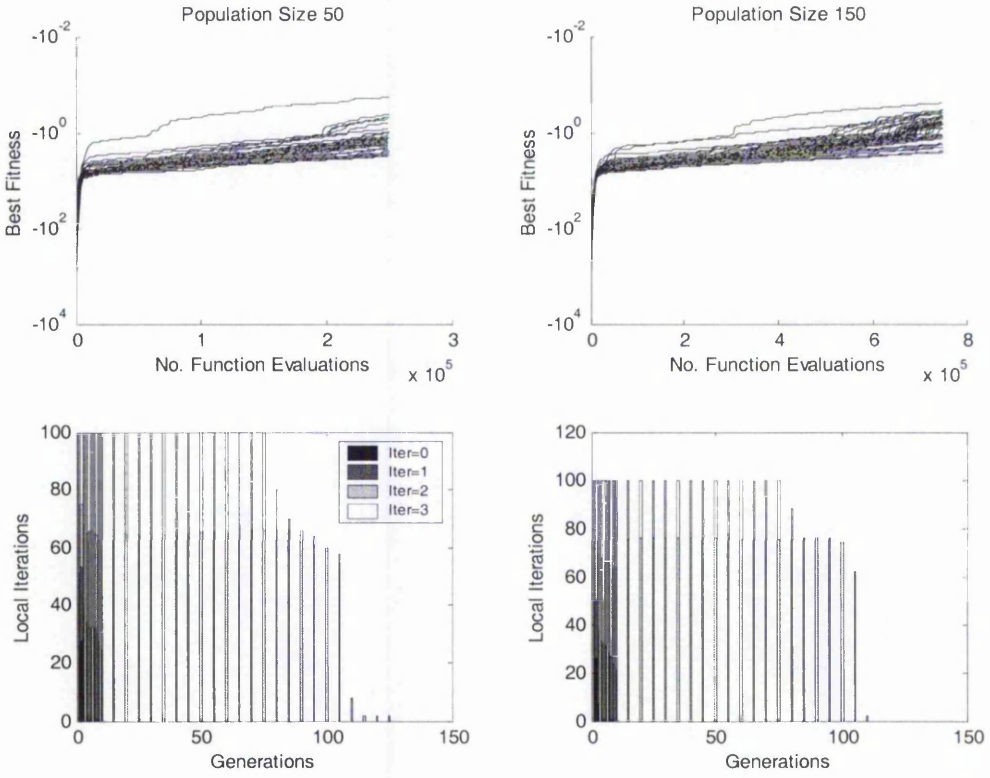

Figure 6.21: Optimising the Rosenbrock Function Using the Lamarckian Approach (Experiment 6.5).

In the previous experiments, the convergence rate of the SADH algorithm was as slow as the ASH hybrid that uses a local search probability of 0.99. This was expected since the fitness is used as a metric for selecting solutions with different costs. The idea of selecting a part of the individuals based on their fitness and the other part based on their speed has been tested. The results of the conducted experiments show that selecting a part of the solutions based on their speed can help to improve the speed of the convergence of the SADH algorithm. However, a fine tuning of this percentage is needed in order to get an efficient and effective search.

6.5.2 Evolutionary self-adaptation versus co-evolutionary self-adaptation

The use of two different mutation rates for the fitness function's variables and the control parameter as a way to help restoring useful genes in the number of local search iterations parameter has also been tested. In other words, the idea of self-adaptation through co-evolutionary was implemented and the performance of the produced algorithm was compared with the evolutionary self-adaptive algorithm.

A set of experiments was conducted on the 10-dimensional Schwefel and the 10-dimensional Rastrigin function for population sizes of 50 and 150 and using the pure Baldwinian, the pure Lamarckian and the 50% partial Lamarckian.

The mutation rate for the number of local iterations control parameter was set so that the chance of mutating the control parameter was equal to the chance of mutating any variable of the fitness function.

The use of the modified mutation rate improved the performance of the self-adaptive algorithm when used to optimise the two test functions, especially when combined with the pure Baldwinian approach. The use of a different mutation rate for the control parameters can help to restore useful control parameter genes and through that it helps to keep diversity in the values of that parameter. The diversity improves the performance of the algorithm using the pure Baldwinian approach.

For example, the co-evolutionary self-adaptive algorithm outperform the evolutionary self-adaptive algorithm when used to optimise the 10-dimensional Rastrigin function in experiment 6.6 using the pure Baldwinian learning strategy as shown in figure 6.22. It maintained the values of 0 and 1, which are not of great impact on the fitness in the first generations. However, these values have a considerable impact on the discriminating between acquired and innate as the search approaches fitness-convergence state. This can explain the improvement in the search performance compared to the one that used a single value for a mutation rate.

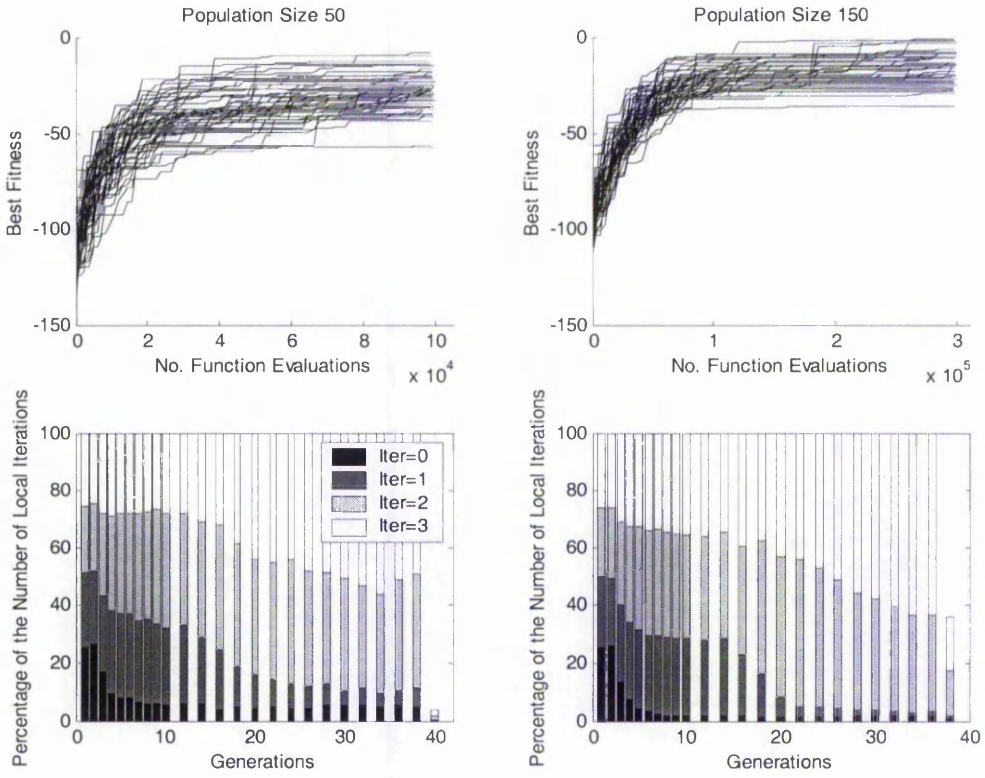

Figure 6.22: The Effect of Modifying the Mutation Rate on the Baldwinian Search (Experiment 6.6).

The results for using the 50% partial Lamarckian and the pure Lamarckian approaches are shown in figure 6.23 and figure 6.24. The graphs show that the algorithm biased the population to use long durations of local search in the first generations. Then, it directed the search to lower the duration value as it approached an optimum. In the case of 50% partial Lamarckian approach, this can help to distinguish between acquired and innate fitness. The use of a co-evolutionary self-adaptive algorithm, as shown in figure 6.23 and 6.24, enables the control parameter to adapt according to the current state of the search, whereas the use of a single mutation rate hinders the ability to adapt since it does not enable the algorithm to restore useful genes. This is reflected in the hybrid performance, which improved using these two learning approaches.

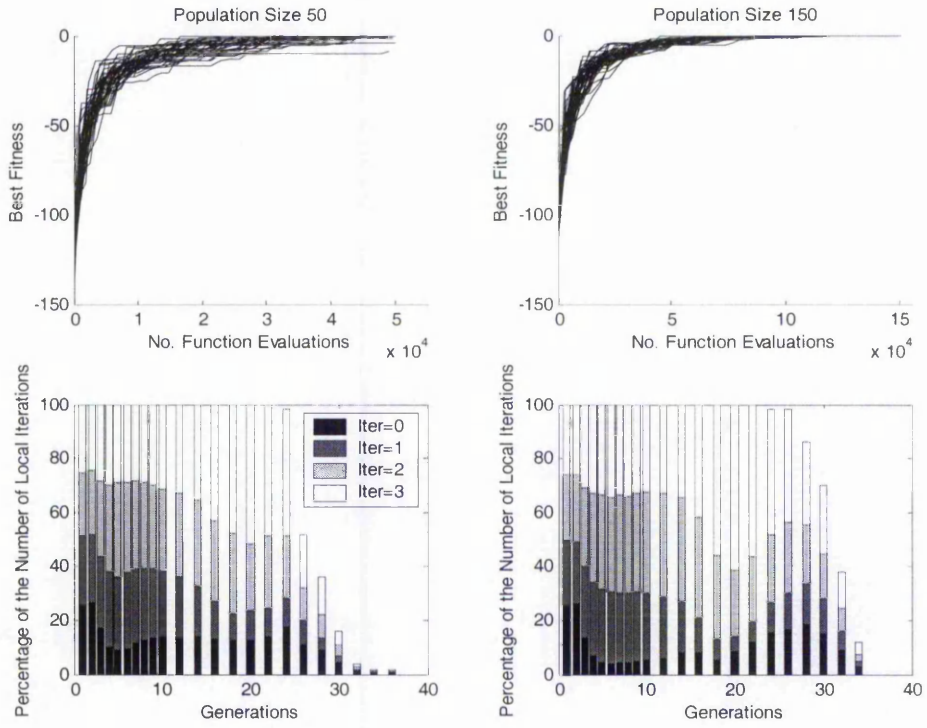

Figure 6.23: The Combined Effect of Modifying the Mutation Rate and the 50% Baldwinian approach (Experiment 6.6).

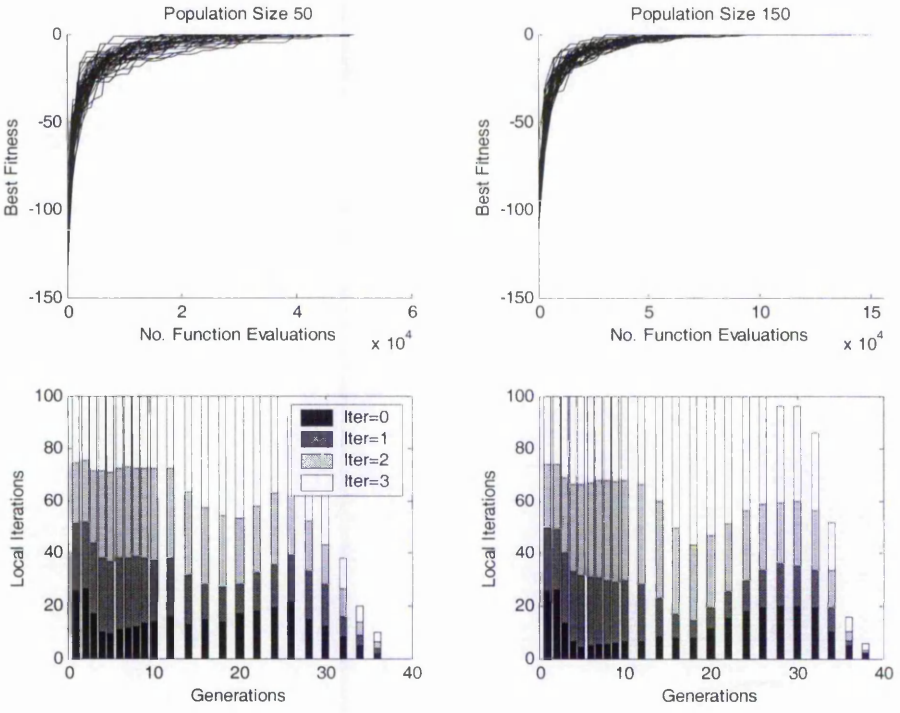

Figure 6.24: The Effect of Modifying the Mutation Rate on the Lamarckian Search (Experiment 6.6).

Optimising the 10-dimensional Schwefel functions in experiment 6.7 showed similar trends where the need for using large number of local iterations reduces as the algorithm approaches an optimum. For this reason, the number of individuals that use no local search increases as the search approaches an optimum. The consequence of this is an improvement in the search performance. The use of a mutation rate for the control parameter that ensures mutating the control parameter at the same rate of the function variables can help to restore useful genes when needed as in this case.

Another set of experiments was conducted in order to compare the performance of the evolutionary self-adaptive and the co-evolutionary self-adaptive hybrids. The algorithm was modified to ensure that a proper mixing of genes representing the number of local iterations control parameter. This can be done by ensuring that a single-point crossover with a specific rate is applied so that the crossover point is within the control parameter representation.

The modified algorithm was able to improve the performance of the self-adaptive algorithm when used to solve the two test function used in the previous experiment.

Figure 6.25 shows the results of experiment 6.8, which used the modified algorithm to solve the 10-dimensional Schwefel function, when combined with the pure Baldwinian approach. The graphs show that ensuring mixing the genes of the control parameter enhanced the performance of the algorithm further. For example, when using a population size of 50, the algorithm can direct the search towards better solutions even in the case of reaching a local optimum. The lower lines of the 50 population size in the graph show that the algorithm was able to improve the quality of these best solutions after reaching a local optimum. The adaptation behaviour of the number of local iterations control parameter becomes clear in these graphs.

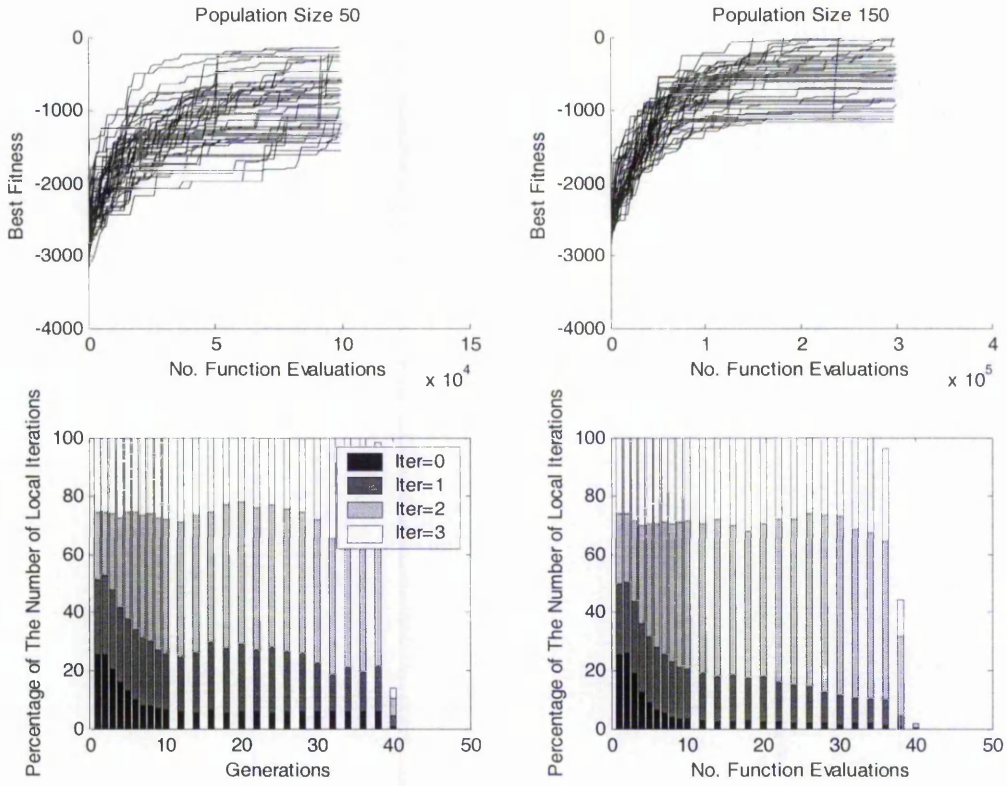

Figure 6.25: The Co-evolutionary Self-adaptive Baldwinian Search with the Schwefel Function (Experiment 6.8).

The results of running the algorithm on the Schwefel function using the 50% partial and the pure Lamarckian approaches showed that the modified algorithm slightly improved the performance of the algorithm for these two learning strategies.

The results of applying the modified algorithm on the 10-dimensional Rastrigin function in experiment 6.9 using the pure Baldwinian approach are shown in figure 6.26. By comparing the graphs of this figure with figure 6.22, it can be seen that the performance of the algorithm using the pure Baldwinian approaches is better than the previous co-evolutionary self-adaptive hybrid.

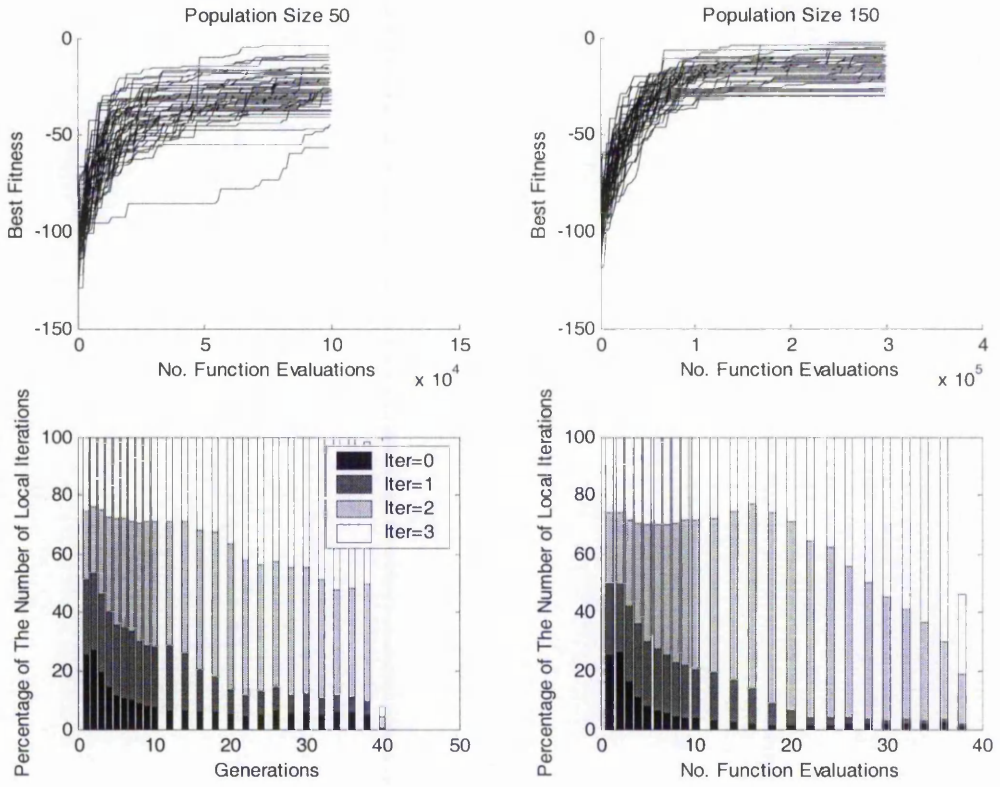

Figure 6.26: The Co-evolutionary Self-adaptive Baldwinian Search with the Rastrigin Function (Experiment 6.9).

6.5.3 Variation or Adaptation

Another set of experiments were conducted to test whether the improvement in solution quality produced was a result of the control parameter adaptation or a result of the variation in its values. In these experiments, instead of using the number of local iterations encoded into each individual to specify the duration of a local search, each individual is allowed to perform a random number of local search iterations in the range of [0, 3].

The 10-dimensional Schwefel and Rastrigin functions were optimised using this hybrid in experiments 6.10 and 6.11 respectively. The results illustrate that the use of random number of local iterations combined with the pure Baldwinian approach produces solutions with a quality higher than that produced by any of the proposed self-adaptive hybrids (figure 6.27).

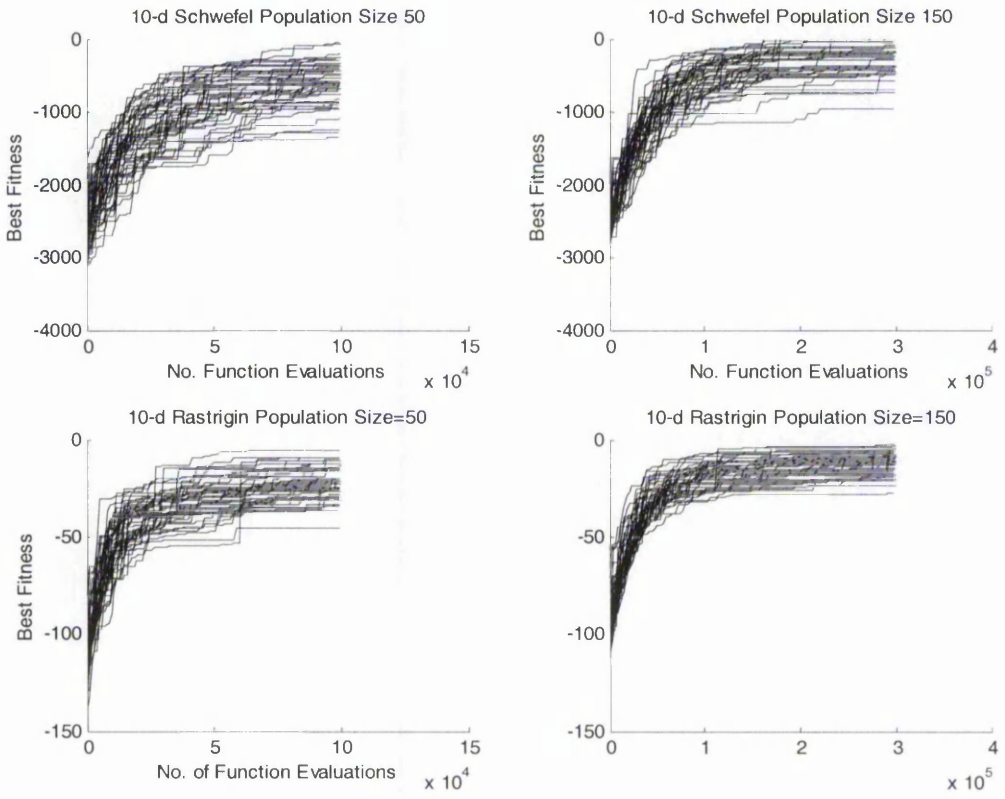

Figure 6.27: Using a Random Number of Local Iterations with a Baldwinian Hybrid (Experiments 6.10 and 6.11).

However, utilising other learning strategies can improve the search speed at the expense of the solution quality. In other words, the solutions quality of the tested self-adaptive hybrids was better than that produced using random values of the control parameters.

The experiments presented in this chapter demonstrate that the criteria used to discriminate between solutions may be suitable to efficiently and effectively adapt the control parameters of the pure genetic algorithms and specific class of hybrids. This class includes hybrids, whose individuals' fitnesses cost an equal number of function evaluations, in addition to hybrids that utilise a low cost local search method. However, the use of the evolutionary self-adaptive metaphor to adapt the control parameters of other classes of hybrids, where the individuals consumed different numbers of function evaluations, can produce effective hybrid algorithms when utilising the pure or partial Lamarckian learning strategies. These experiments also showed that the SADH algorithm performed poorly when combined with the pure Baldwinian approach due to the hindering effect problem.

The performance of this combination can be improved by combating the hindering effect through using local search methods with very short durations.

These experiments also demonstrated that the improvement in performance of the non-pure Baldwinian approaches becomes more significant as the dimension of the fitness function increases. This can be explained in terms of the cost of the adaptation process. The cost of adapting a single control parameter becomes less significant as the number of variables of the function to be optimised increases.

The experiments also illustrate that the co-evolutionary self-adaptive hybrid algorithm outperforms the evolutionary self-adaptive hybrid algorithm. Ensuring that the control parameter is subjected to the mutation and the crossover operations at the same rate of other search variables significantly improves the self-adaptive hybrid performance using the pure Baldwinian approach. It also improves the performance of hybrids that adopted other learning strategies. The co-evolutionary self-adaptive algorithm accelerates the rate of adaptation of the control parameter and maintains diversity in its values.

Chapter 7 Self-adaptive learning approach

The way of utilising gained information through local search within a hybrid genetic algorithm has a great impact on the performance of the search process (Whitley et al. 1994). Due to the similarities in the role of the local search within the genetic search and the role of learning within the evolution process, the local search is usually viewed as a learning process. For this reason, hybrid genetic algorithms use local search information utilisation approaches that are inspired by biological learning models. These approaches are the Lamarckian and the Baldwinian (Hinton and Nolan 1987) learning strategies. In addition to these basic models of learning, a third model, which is referred to as partial Lamarckianism (Orvosh and Davis 1993) (Houck et al. 1997) (Joines et al. 2000b) (Espinoza et al. 2001) (Ishibuchi et al. 2003) and Baldwinian-Lamarckian hybrid (Sung-Soon and Byung-Ro 2005), has been widely used. This model is a hybrid of the Lamarckian and the Baldwinian strategies in order to get the best out of both.

The effectiveness of using the Lamarckian approach, the Baldwinian approach, or a mixture of them in a hybrid genetic algorithm is affected by the fitness landscape, the genetic algorithm setup (Michalewicz and Nazhiyath 1995) (Ishibuchi et al. 2005), the percentage of population performing local search (see chapter 4), the duration of local search (see chapter 3), and the local search method used (Ku and Mak 1997). With the restricted amount of theory currently available for choosing the learning strategy that best matches a given black box problem in a hybrid search, it is reasonable to ask whether the effects of this choice on performance might be reduced via some intelligent means while the search is progressing. Houck et al. (1997) suggested that applying different mixtures of the Lamarckian and the Baldwinian approaches over the course of the genetic run can be more beneficial than applying a single basic learning model or a fixed mixture of learning models during the entire run.

The aim of this chapter is to investigate the use of an adaptive approach to decide on the learning mechanism. Assigning different learning strategies for the population's individuals over the course of the run via some intelligent means is investigated through applying evolution to self-adapt the learning mechanism within a hybrid genetic algorithm. This chapter examines the effect of this form of adaptation on the hybrid's performance in order to get some insight into its advantages and disadvantages. It also investigates the interactions between this form of adaptive learning and the Self-Adaptive local-search-

Duration Hybrid (SADH) (chapter 6) and the Adaptive Staged Hybrid (ASH) (Espinoza et al. 2001) algorithms.

This chapter starts with a very brief review of the learning approaches used in hybrid genetic algorithms. Then, it goes on to describe the proposed adaptation mechanism and the way it works. This chapter concludes by presenting and discussing the results of the experiments that have been conducted using the Self-Adaptive local-search-Duration Hybrid (SADH) and the Adaptive Staged Hybrid (ASH) algorithms on a selected set of test functions.

7.1 Utilising local search information

Local search methods are incorporated into genetic algorithms in order to improve the algorithm's performance through learning. The utilisation of local knowledge of a sampled solution through learning can improve the chances of good building-blocks to propagate into the next generation even in the case of being sampled by solutions of under average fitness. Learning can also refine sampled solutions in order to build better building-blocks. The way by which gained information is utilised within a hybrid genetic algorithm influences the performance of the search process. Using an appropriate learning mechanism can accelerate the search towards the global optimum. On the other hand, employing an inappropriate mechanism can either cause a premature convergence problem or decelerate the search towards the global optimum.

Two basic learning models, which are the Lamarckian and the Baldwinian approaches, have been used to utilise local information. In the former approach, both the genetic structure and its fitness merit are changed to reflect the improvement in individual traits as a result of performing local search. In the Baldwinian approach, only the fitness is changed to reflect this improvement. A third approach, which is known as partial Lamarckian, has also been used widely. In this approach, the structures of only a part of the individuals that performed a local search are updated. The reader can refer to section 2.4 for more details on these learning models.

The adoption of any form of learning in a hybrid genetic algorithm has a great impact on its performance. Several researchers have investigated how these different leaning strategies affect the performance of hybrid genetic algorithms by comparing them with pure genetic algorithms. Gruau and Whitley (1993) compared Lamarckian, Baldwinian and pure genetic algorithms in evolving the architecture and the weights of neural networks that learn Boolean functions. They conclude that using either form of leaning is better than using a

pure genetic algorithm. Orvosh and Davis (1993) found that 5% partial Lamarckian is the optimal learning strategy to solve the survival network design problem and the graph colouring problem. Michalewicz and Nazhiyath (1995) replaced 20% of the repaired solutions in their hybrid algorithm to solve numerical optimisation problems with nonlinear constraints. Bala et al. (1996) showed how the Baldwin effect can improve the performance of a genetic algorithm when integrated with a decision tree in order to evolve useful subsets of discriminatory features for recognizing complex visual concepts. However, Ku and Mak (1997) found that only using Lamarckian evolution improved the performance of genetic algorithm in evolving recurrent neural networks. They also concluded that effective hybridisation depends on the local search method used and learning frequency. Houck et al. (1997) used seven problems to compare the performance of different learning strategies. Their investigation concluded that neither the pure Lamarckian nor pure Baldwinian strategy was found to be consistently effective. It was discovered that the 20% and 40% partial Lamarckian search strategies yielded the best mixture of solution quality and computational efficiency based on a minmax criterion (i.e. minimising the worst case performance across all test problems instance). Sasaki and Tokoro (1997) found that adaptation by Lamarckian evolution was much faster for neural networks than Darwinian evolution in a static environment. But when the environment changed from generation to generation, the Darwinian evolution was superior. Julstrom (1999) reported that Baldwinian strategies are performing poor in solving the 4-Cycle problem compared to a pure genetic algorithm and their effectiveness deteriorating with an increasing use of learning in contrast to Lamarckian strategies He also found that leaning all the individuals using Lamarckian strategy produced the most effective results. Joines et al. (2000b) found that using the pure Lamarckian approach (100% Lamarckian) produced the best convergence speed to the best known solution when solving the cell formation problem. Espinoza et al. (2001) used 75% partial Lamarckian as the optimal learning strategy in their hybrid to optimise two simple continuous functions. Ishibushi et al. (2005) found that the 5% partial Lamarckian worked well on the multi-objective 0/1 knapsack problem using a single population model, however, the 50% partial Lamarckian was the optimal choice using the island model.

7.2 The evolutionary self-adaptation of learning approach

It is almost impossible to know which learning strategy is most suitable for a problem when there is only limited knowledge of the fitness landscape available. With the restricted amount of theory currently available for choosing the learning strategy that best matches a problem with no knowledge of its fitness topology in a hybrid search, the use of

an effective adaptive technique to decide on learning strategy while the search is processing would clearly be of benefit.

The idea behind the adaptive strategies is that, as the search progresses, the effectiveness of each learning strategy in dealing with the current problem can be learnt. Knowledge about the current population of solutions and each learning strategy can be built dynamically online, so identifying the strengths and weaknesses of the learning approach for the problem currently being worked on, given its current state.

The use of evolution to self-adapt the learning mechanism can help to discover the effectiveness of each learning approach in dealing with a given problem online. This adaptation can improve the hybrid's chances to find good solutions by enabling the different learning approaches to compete and cooperate with each other. By encoding the learning strategy used by an individual into its chromosome, the global genetic algorithm can promote competition among the different learning strategies based on its ability to improve the fitness of its associated solution. A good learning strategy will lead to good individuals and these will probably have more chances to survive and propagate the encoded learning approach. Applying the evolutionary self-adaptation metaphor to decide on the learning strategy can also promote cooperation between the two basic learning models in order to improve the search's performance. The use of a suitable learning approach depending on the genetic structure of an individual, and the current search state, may lead to a search algorithm that makes use of the available learning strategies to improve the whole population's performance. By ensuring the participation of the two basic learning models in the problem search, the strategy promotes joint operation and hence cooperation between learning models.

7.2.1 The algorithm

The evolutionary self-adaptive learning mechanism was incorporated into the SADH and the ASH algorithms. An additional bit was appended to the end of an individual's chromosome. The association of the learning strategy with a solution through binding them into the same chromosome can help to associate the success or the failure of a learning technique to a specific solution or solutions of similar genetic structures.

In the case of the SADH algorithm, the bit that represents the learning strategy is located after the genes that represent the number of local search iterations. However, for the ASH algorithm, the learning strategy bit is located after the genes that represent the function variables. After performing a local search operation and before returning to the global

genetic algorithm, the algorithm reads the value of the learning strategy bit to decide whether to change the genetic structure and the fitness score of the initial solution to match that of the improved solution or keep its genetic structure unchanged and modify the fitness score only. In these algorithms, the value of 1 was used to represent the Lamarckian approach, while the Baldwinian learning was represented by the value of 0. Depending on the value of the learning strategy gene, the hybrid decides on the learning strategy to use in order to utilise local search information of a given solution.

7.3 Experiments

For the purpose of evaluating the proposed learning adaptation mechanism, it was incorporated within the two adaptive hybrid algorithms. The performance of the resulting algorithms was compared with the performance of the two hybrids using fixed percentages of partial Lamarckian. The quality of the solutions produced by each algorithm and the speed of convergence were used to assess the algorithm's performance. The percentage of experiments that converged to the global optimum was used as an indication of the ability of the proposed adaptive learning mechanism to produce high quality solutions. The performance was compared using different population sizes in order to evaluate the ability of the proposed mechanism to adapt to different search environments. The speed of finding the global optimum was also used to evaluate the performance of the self-adaptive learning strategy.

A set of test functions has been chosen to evaluate the use of evolution to self-adapt the learning strategy. Four test functions have been used to evaluate the ability of this form of adaption to improve the search performance compared to that of the pure Lamarckian, the pure Baldwinian and fixed partial Lamarckian approaches.

The test functions suite includes the 10-dimensional ellipsoidal (Deb et. al 2002), the 10-dimensional Rastrigin (Törn and Zilinskas 1989), the 10-dimensional Schwefel (Mühlenbein et. al 1991), and the 10-dimensional Griewank (Mühlenbein et. al 1991) test functions. The reader can refer to chapter 6 for more details on these functions.

The results of optimising the test functions using the self-adaptive learning mechanism were evaluated against the results obtained by using fixed learning strategies. The fixed learning strategies tested were the pure Baldwinian (0% partial Lamarckian), the 25% partial Lamarckian, the 50% partial Lamarckian, the 75% partial Lamarckian and the pure Lamarckian (100% partial Lamarckian).

The hybrids use the simple elitist genetic algorithm with binary tournament selection, two-point crossover, and simple mutation. For all experiments, the probability of crossover was set to 0.7 and the probability of mutation was $1/N$ where N is the population size. For the ASH algorithm, the maximum number of local iterations was 3, e was 0.2, and the local threshold value was 0.6. The ASH algorithm tested using different values of initial local search probability, which are of 0.1, 0.2, and 0.99. For the SADH algorithm, the maximum number of local search iterations was set to 3.

The stopping criterion for all experiments was a maximum number of function evaluations. The value of this parameter was set to 2000 times the population size for the Rastrigin, the Schwefel, and the Griewank test functions, and to 500 times the population size for the ellipsoidal test function due to its simplicity. Each experiment was repeated 50 times.

A local search method, which combines the steepest descent method and Brent's method (Press et al. 1993) to estimate the best step size to climb to the local optimum from the current position in the basin of attraction, was used. The steepest descent algorithm uses the derivatives of the fitness function to estimate the best step size to climb to the local optimum from the current position in the basin of attraction. Brent's method fits a parabola to three initial solutions and uses the maximum of the parabola as the next potential solution of the overall function (chapter 2).

In these experiments, the self-adaptive learning strategy mechanism was evaluated in terms of quality of the solutions produced, convergence speed and in terms of its ability to adapt to different fitness landscapes.

7.3.1 Search effectiveness

The percentage of times a hybrid algorithm found a global optimum using the proposed adaptation mechanism is compared with that of using fixed learning strategies. These percentages were used to evaluate the effectiveness of the proposed adaptive learning mechanism when combined with different hybrids in solving the test problems.

In experiment 7.1, the proposed adaptation mechanism and fixed learning strategies were utilised within the two hybrids to find the global optimum of the ellipsoidal test function. Figure 7.1 compares the percentage of times that the different hybrids found this optimum. The graphs show that combining the adaptive learning technique with the ASH algorithm produced better performance than that produced by combining it with the pure Baldwinian approach for different initial probabilities of local search and different population sizes.

The combination produced a performance that is similar to that produced using the 25% partial Lamarckian. The graphs of the SADH algorithm illustrate that the adaptive learning mechanism produced the best performance for population sizes of 150, 200 and 250 compared with that produced by using different fixed learning strategies.

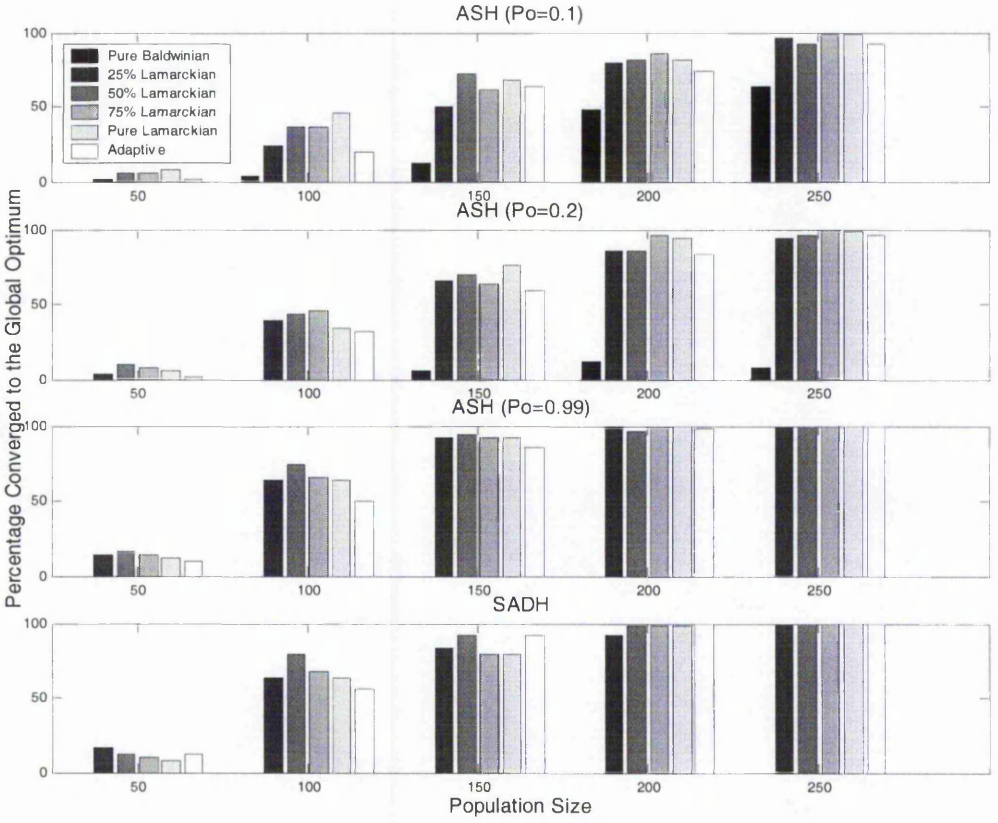

Figure 7.1: Percentages Converged to the Global Optimum of the Ellipsoidal Function (Experiment 7.1).

The results of the experiment 7.2, which evaluated the effect of the self-adaptive learning mechanism on the search effectiveness on the 10-dimesional Griewank test function, are shown in figure 7.2. The plots illustrate that the performance of the ASH algorithm was improved when combined with the self-adaptive learning compared to that when combined with fixed learning techniques for most of the tested population sizes and probabilities of local search. The results of applying the adaptive learning mechanism to the SADH algorithm show that this mechanism outperformed the pure Baldwinian and the 25% Lamarckian approaches for all the tested population sizes.

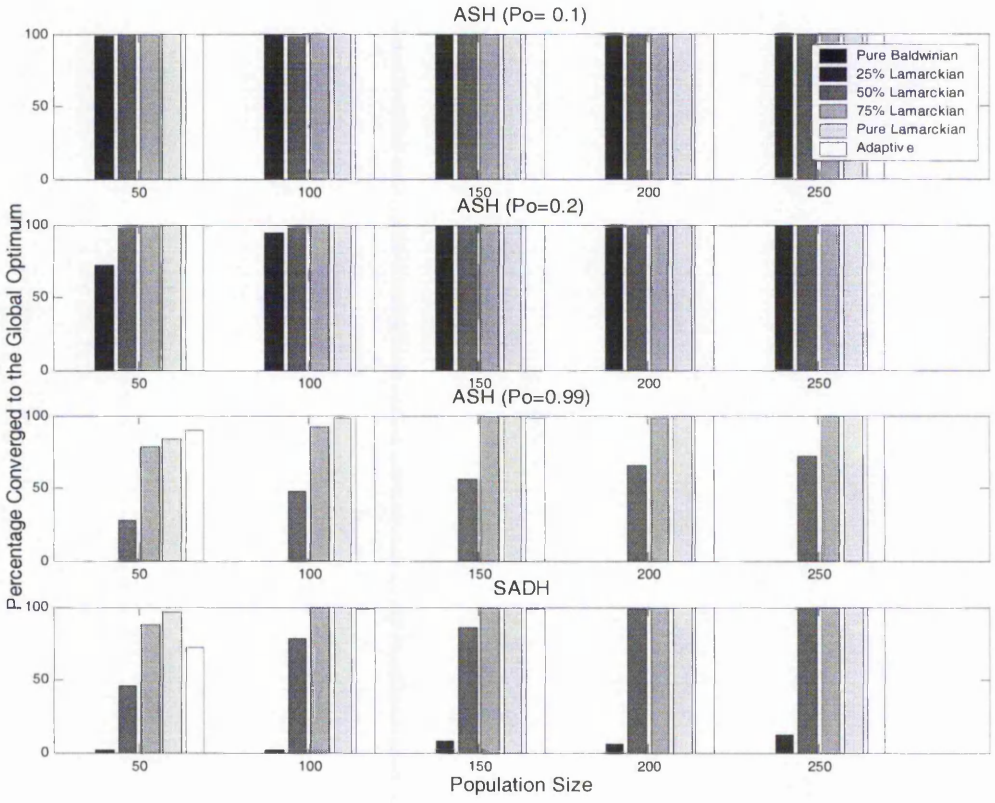

Figure 7.2: Percentages Converged to the Global Optimum of the Griewank Function (Experiment 7.2).

The self-adaptive learning mechanism was used to optimise the 10-diemsional Rastrigin function using the two hybrids in experiment 7.3. The proposed mechanism produced a similar performance using the two hybrids in terms of the number of experiments that found the function's global optimum as shown in figure 7.3. The two hybrids when combined with the self-adaptive learning produced a performance that is similar to that is produced by the partial 25%, 50%, 75% and 100% Lamarckian approaches, which produced the best performance for most of the tested population sizes.

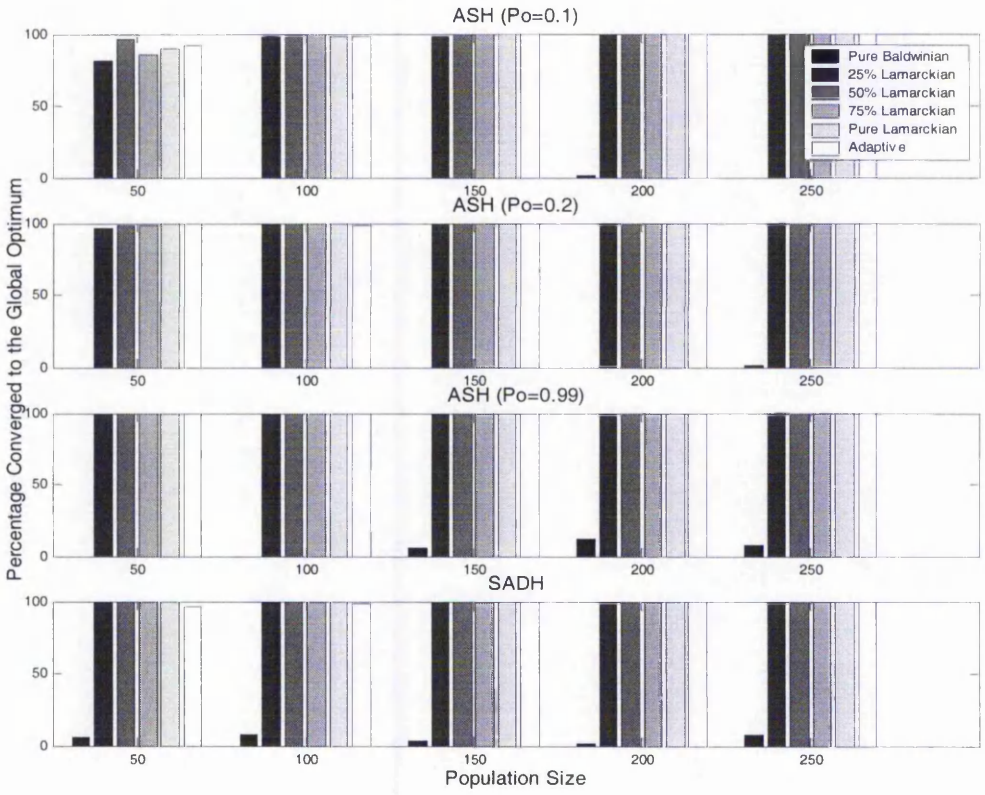

Figure 7.3: Percentages Converged to the Global Optimum of the Rastrigin Function (Experiment 7.3).

The self-adaptive learning technique outperformed the pure Baldwinian approach in terms of solution quality when applied to solve the 10-dimensional Schwefel problem using the different adaptive hybrid algorithms in experiment 7.4 as depicted in figure 7.4. The self-adaptive mechanism when combined with the ASH algorithm outperformed the partial 25% Lamarckian approach in all experiments except the one which combined a population size of 200 with a local search probability of 0.1. The graphs for the ASH algorithm show that there is no significant difference between the performance of this adaptation mechanism and the best found fixed learning strategy in about half of the tested combinations of population sizes and local search probabilities.

The plots of the SADH algorithm show that the adaptive learning strategy technique outperformed the pure Baldwinian learning strategy, which produced the worst performance in these experiments. This can be explained based on the fact that allowing a small fraction of the population to evolve according to the Lamarckian learning can help to alleviate the hindering effect which, in turn, improves the possibility of finding the global optimum.

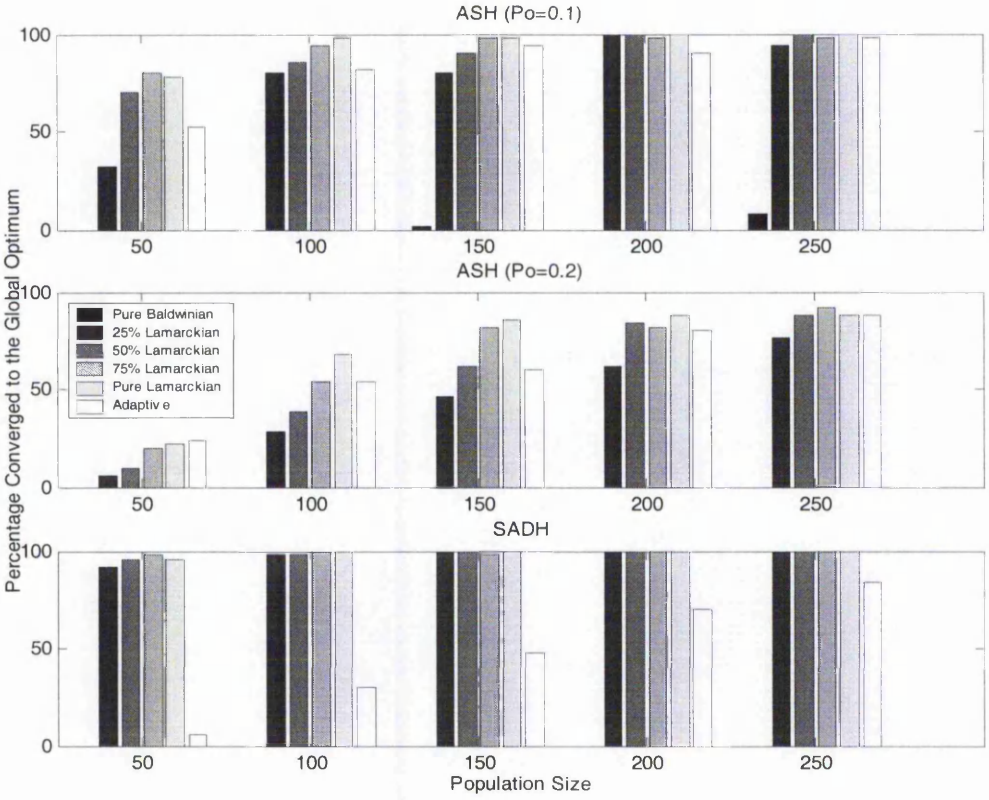

Figure 7.4: Percentages Converged to the Global Optimum of the Schwefel Function (Experiment 7.4).

7.3.2 Search efficiency

The number of function evaluations needed by a hybrid algorithm to find the global optimum of a specific function was used to measure the effect of the self-adaptive learning mechanism on the search efficiency. The convergence speed of the hybrids that use the adaptive learning technique was compared with those using fixed learning strategies.

The convergence speed of different hybrids in finding the global optimum of the different test functions in the previous experiments are shown in figures 7.5-7.8. The graphs compare the speed of the self-adaptive learning technique with the selected set of fixed learning strategies. However, the graphs of the pure Baldwinian learning strategy were excluded from these figures since in most of the cases it failed to find the global optimum.

Figure 7.5 shows the results of comparing the convergence speed of the two hybrids in finding the global optimum of the ellipsoidal function using the proposed mechanism with that of using fixed learning strategies. The graphs for the ASH algorithm show that the adaptive learning technique found the global optimum of the ellipsoidal function faster than

those using fixed learning strategies. The difference in the convergence speed becomes apparent as the population size increases. However, the self-adaptive learning technique, when combined with the SADH algorithm, produced the worst performance compared with the fixed learning techniques excluding the pure Baldwinian approach.

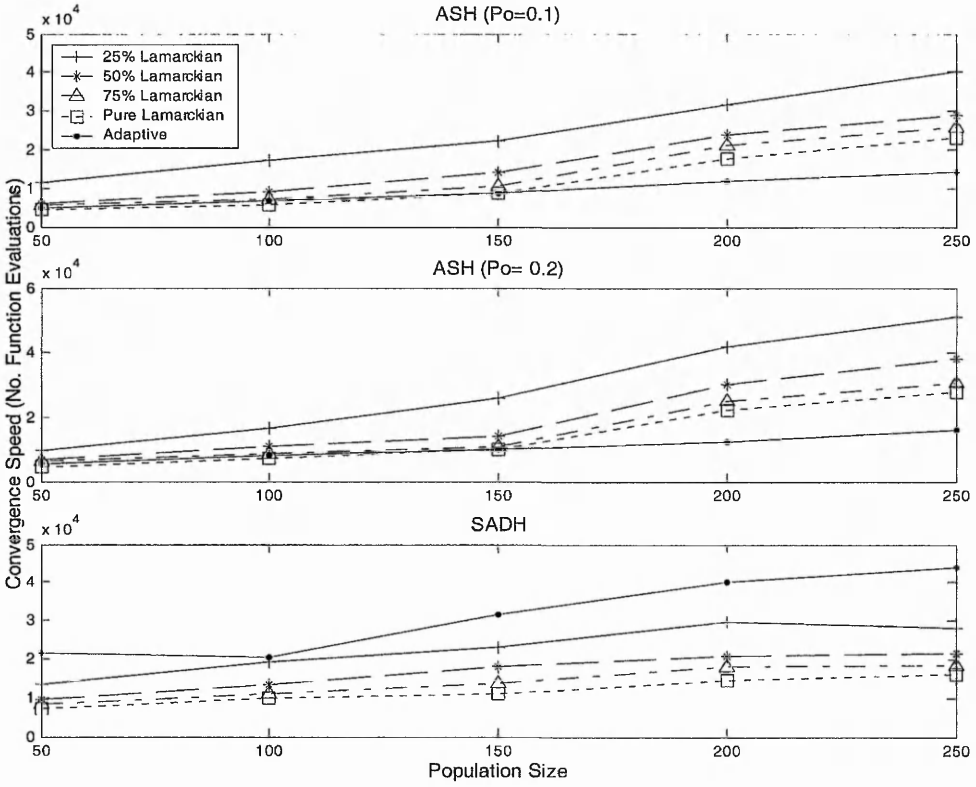

Figure 7.5: Convergence Speed of the Ellipsoidal Function (Experiment 7.1).

The results of comparing the convergence speed of the adaptive learning with the fixed learning approaches on the Griewank function in experiment 7.2 are shown in figure 7.6. These graphs show no significant difference between the adaptive and the fixed learning strategies when combined with the ASH algorithm regardless of the local search probability. However, the adaptive learning technique produced the worst performance when combined with the SADH algorithm compared with the fixed learning strategies excluding the pure Baldwinian approach.

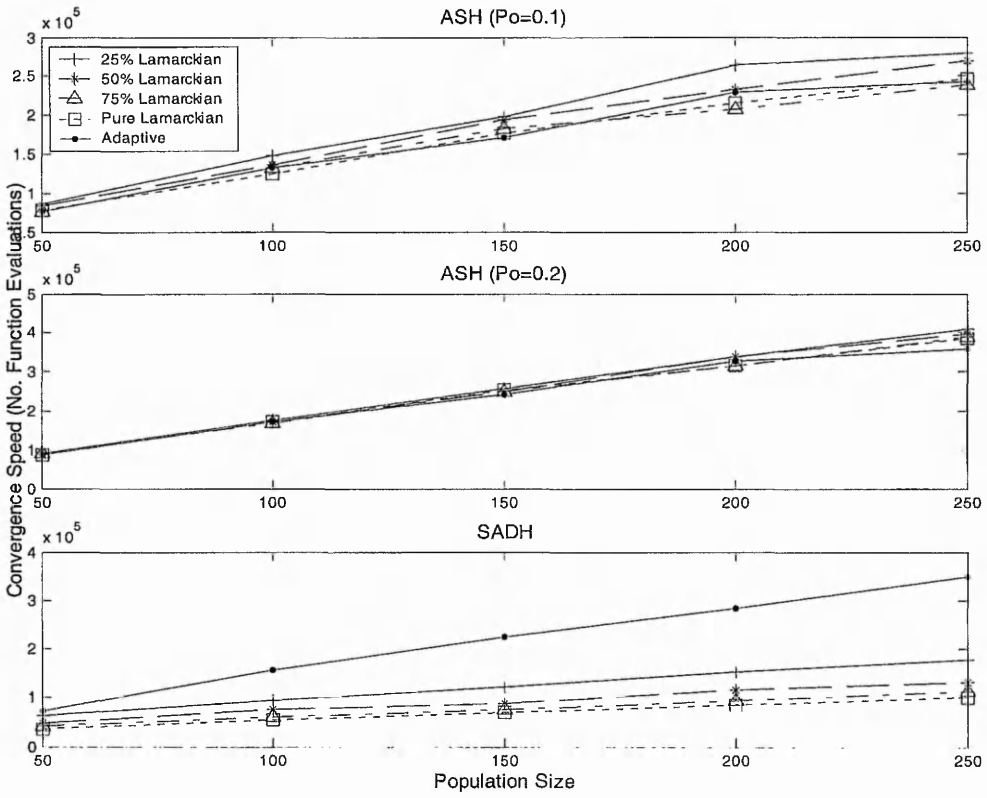

Figure 7.6: Convergence Speed of the Griewank Function (Experiment 7.2).

The graphs in figure 7.7 show the speed of finding the global optimum of the Rastrigin function in experiment 7.3. It can be seen that the adaptive learning strategy is the second fastest learning mechanism when combined with the ASH algorithm using a local search probability of 0.2. However, it is the second slowest when used with a local search probability of 0.99. The curves also illustrate that the self-adaptive learning mechanism produced the worst performance when combined with the SADH algorithm.

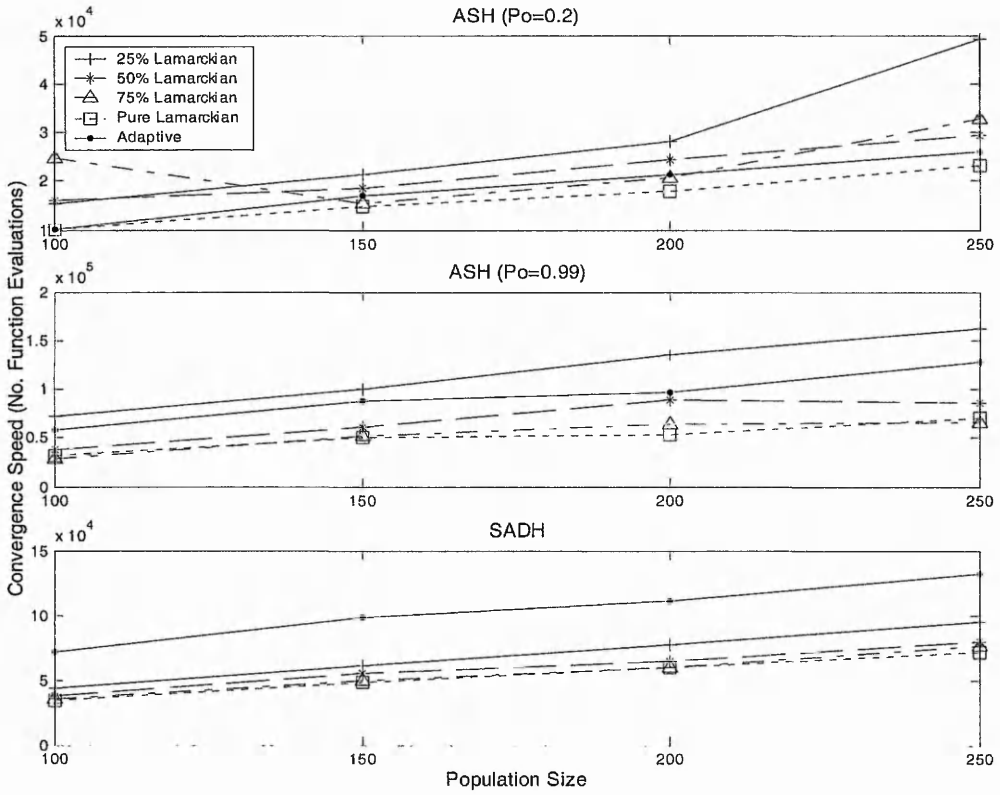

Figure 7.7: Convergence Speed of the Rastrigin Function (Experiment 7.3).

The results of experiment 7.4 showed that the adaptive learning mechanism was almost the fastest in finding the global optimum of the Schwefel function when combined with the two adaptive hybrid algorithms and using different population sizes, as illustrated in figure 7.8. These graphs also show that the proposed learning mechanism was the fastest when combined with the ASH algorithm regardless of the probability of local search used.

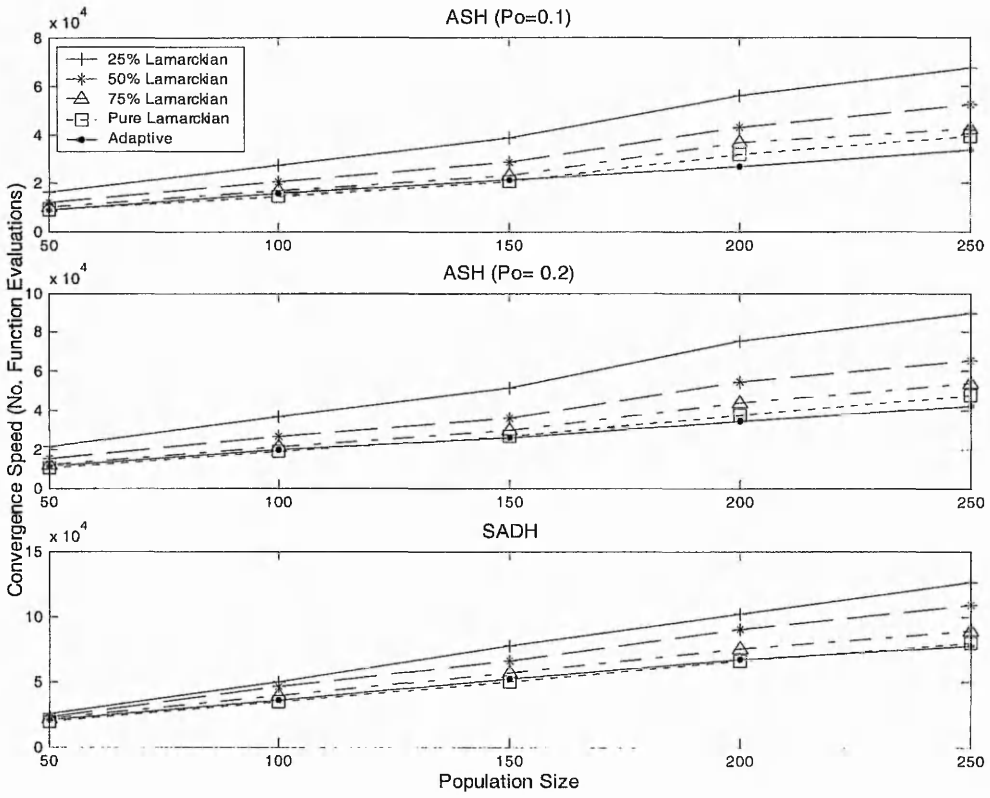

Figure 7.8: Convergence Speed of the Schwefel Function (Experiment 7.4).

7.3.3 Evolution of learning strategy

The ability of the self-adaptive learning strategy to adapt to different fitness landscapes and population sizes was evaluated through monitoring the changes in the learning strategy over time. The graphs of the evolution of the learning strategy and the best fitness for two test functions and using two population sizes are presented and discussed in this section. The changes in the percentages of the population that used the different local search iterations over generations are also presented in the case of the self-adaptive hybrid algorithm. Each graph shows the results of 50 experiments.

Figure 7.9 shows the evolution of the best fitness, the learning strategy and the duration of local search of the SADH algorithm when used to solve the ellipsoidal problem in experiment 7.1. The graphs illustrate that, at the early stages of search, the fraction of the population that evolved using the Baldwinian learning increased slightly. After that, the percentage of the population that used Lamarckian learning increased as the search progressed until the whole population became using the Lamarckian approach. The figure

clearly shows that the increase in the number of individuals that using the Lamarckian approach is accompanied by an increase in using long durations of local search.

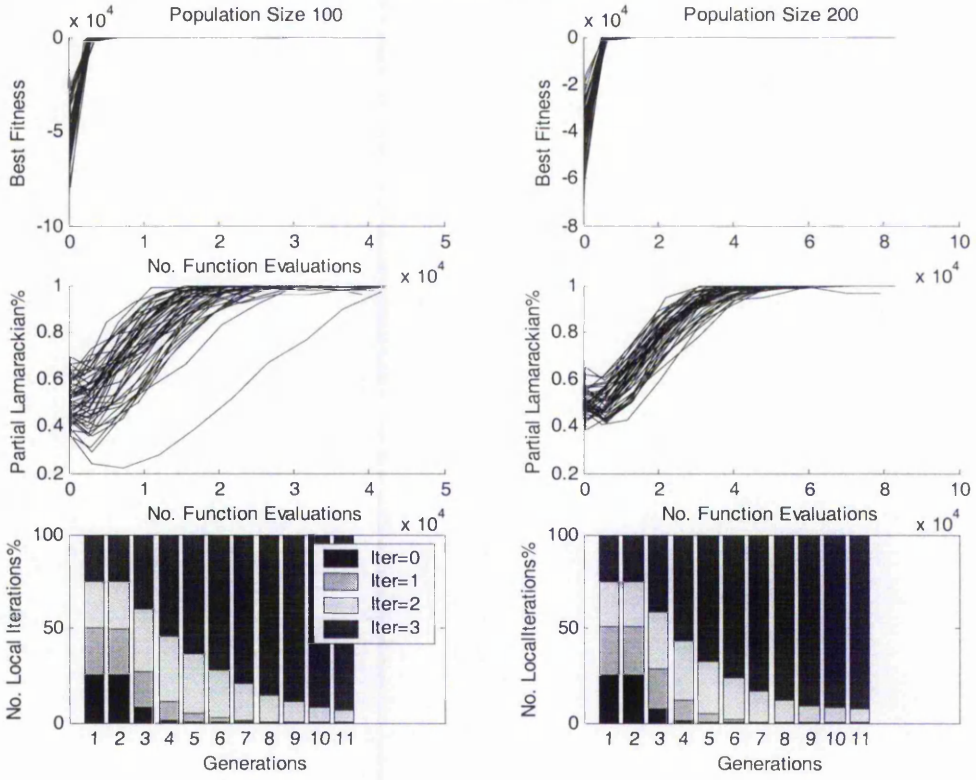

Figure 7.9: The Evolution of Learning Strategy when Solving the Ellipsoidal Problem (Experiment 7.1).

The graphs of the Rastrigin function in figure 7.10 show a similar trend to that found in figure 7.9. They show that the fraction of population that evolved using the Lamarckian search increased as the search progressed. The graphs in figure 7.10 also show that there is a trend to use short durations of local search accompanied with the use of the Baldwinian search. This trend is apparent at the final stages of the search using a population size of 100, where individuals tended to use short durations of local search to overcome the hindering effect problem associated with the Baldwinian strategy. This clearly shows the ability of the self-adaptation mechanism to discover the relations between different control parameters such as the relations between the learning strategy and the duration of local search. The difference in the number of function evaluations consumed at each local search process caused the algorithm to evolve to different number of genetic generations. For example, the graphs for the population size of 100 in figure 7.10 show that most the experiments consumed their budget of function evaluations by the 11^{th} generation. The graphs also

illustrate that the evolution trajectory of the learning strategy in the Rastrigin function is more complicated than that of the ellipsoidal function.

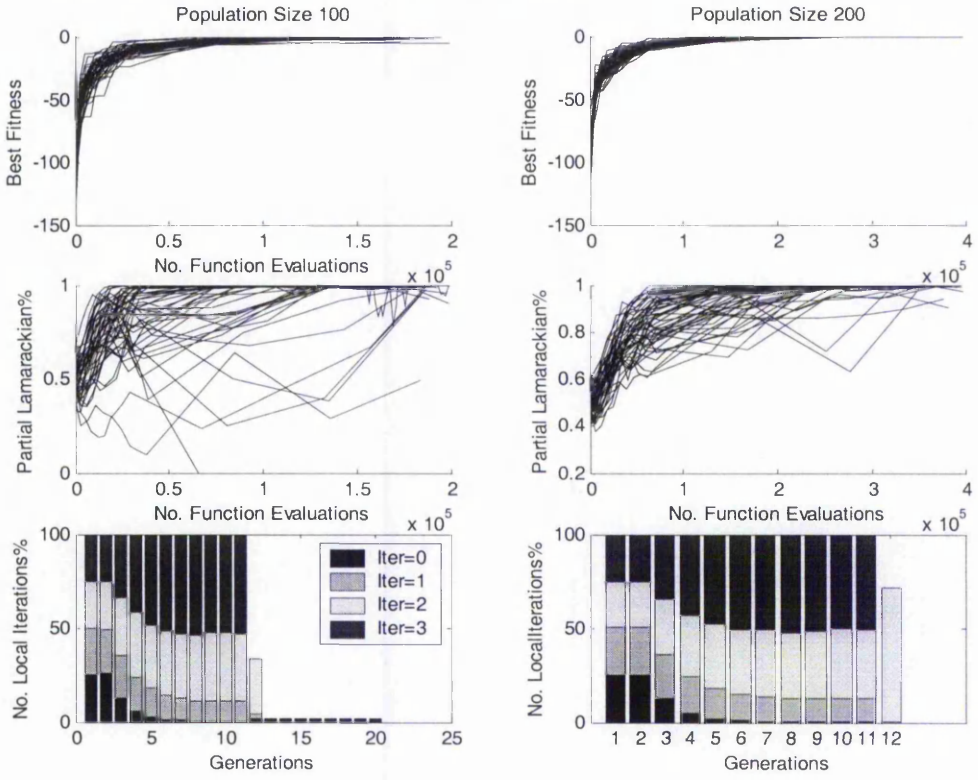

Figure 7.10: The Evolution of Learning Strategy when Solving the Rastrigin Function (Experiment 7.3).

Figure 7.11 shows how the learning strategy evolved over time when combined with the ASH algorithm to solve the ellipsoidal function using a population size of 100 in experiment 7.1. The figure shows the same trend that the figures of the SADH algorithm show. As the search progressed, the number of solutions that evolved according to the Lamarckian learning increased. The plots show that, for a local search probability of 0.1, the percentage of the population that evolved using the Lamarckian approach was in the range between 70 and 100. However, for a local search probability of 0.99, this percentage approached 100, as the search progressed. This can be explained based on that using a small probability of local search can help to fight the hindering effect which, in turn, enables the algorithm to find the global optimum even when combined with the Baldwinian approach. On the other hand, the probability of finding the global optimum increases with the increase in the partial Lamarckian approach for high probabilities of local search once the algorithm guided the search to the global optimum's basin of attraction. These graphs also

demonstrate that the hybrid that used a local search probability of 0.1 was faster than the one using a probability of 0.99 in optimising the learning strategy control parameter since it has more chances to evolve in the first case than in the latter one. This is due to the differences in the number of function evaluations consumed at each local search process.

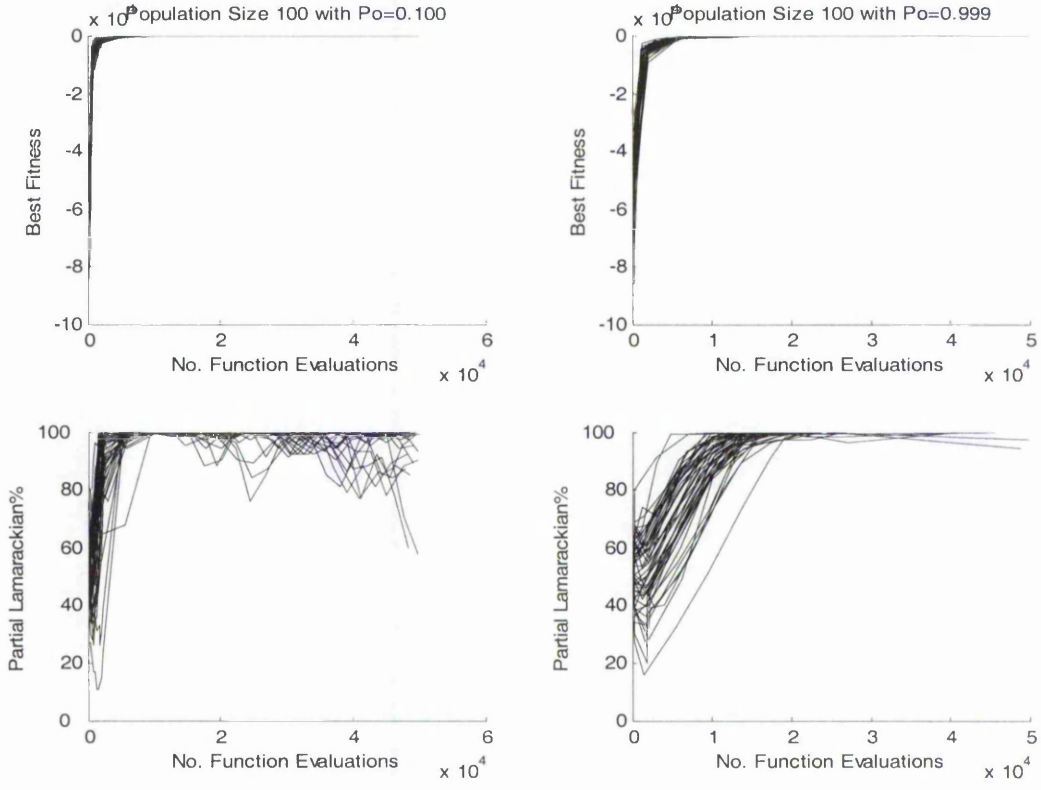

Figure 7.11: The Evolution of Learning in the ASH Algorithm Solving the Ellipsoidal Function (Experiment 7.1).

The figure 7.12 shows the results of optimising the Rastrigin function using a population size of 200 in experiment 7.3. The fitness landscape of this function is more complicated than that of the ellipsoidal function. This, in turn, makes the evolution path of the learning strategy more complicated. The graphs also show that as the local search probability increases, the trend towards using more Lamarckian increases.

7.4 Conclusions

The experiments conducted illustrate that the use of the self-adaptive learning strategy can be beneficial. It can improve the search ability of finding solutions of high quality and can accelerate the search. They also show that this mechanism was able to adapt with different environments. That was illustrated by testing this mechanism on a set of

different test functions using two different adaptive hybrid algorithms with different control parameters.

These experiments demonstrate that combing the self-adaptive with the ASH algorithm produced an algorithm that is faster than the tested fixed learning strategies on most of the tested functions. However, combining the SADH algorithm with this mechanism produced a slow search algorithm. The combination of both was able to find the global optimum of the whole set of test function more often and faster than that of the fixed pure Baldwinian approach.

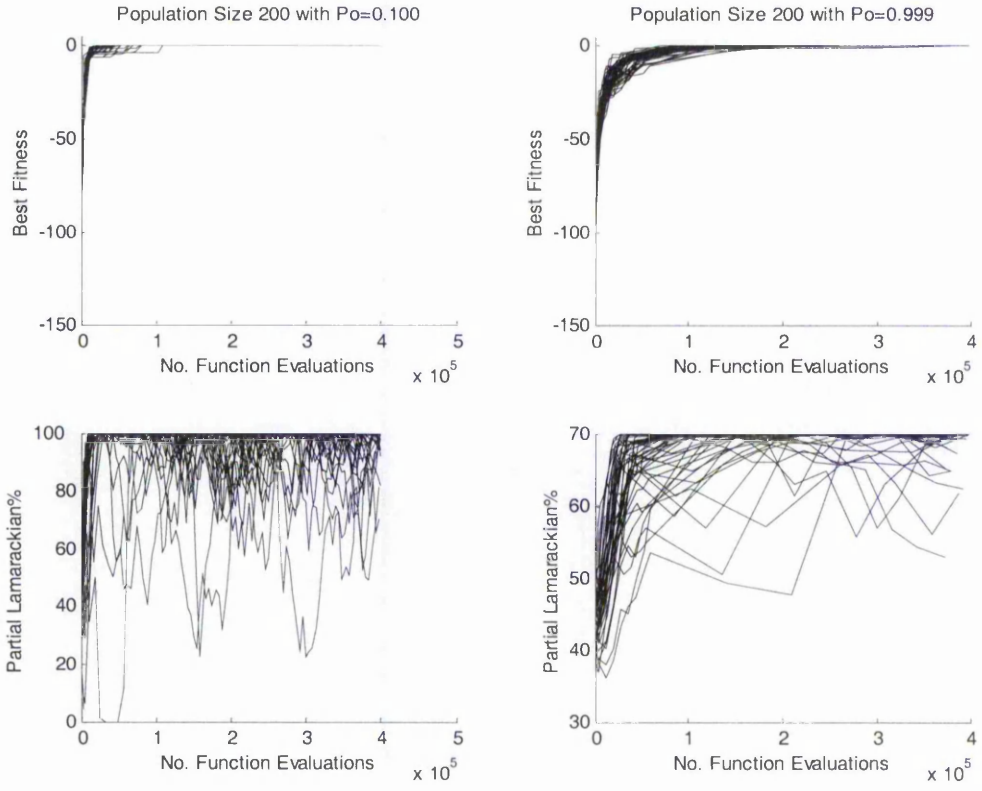

Figure 7.12: The Evolution of Learning in ASH Algorithm Solving the Rastrigin Function (Experiment 7.3).

Chapter 8 An ant-based algorithm to self-adapt genetic-local hybrids

The pheromone trail metaphor (Dorigo et al. 1991) is a simple and effective way to accumulate the experience of the past solutions in solving a given problem. Ant-based optimisation algorithms (Dorigo and Di Caro 1999) have successfully employed this metaphor to solve a large number of hard optimisation problems.

The problem of achieving an optimal utilisation of a hybrid's search time can be viewed as a problem of finding an optimal set of control parameters for that hybrid. In chapter 6, the use of evolution to self-adapt the duration of local search, as a way to strike a balance between exploration and exploitation, was investigated. The results of the experiments showed that evolutionary self-adaptation can produce an effective search algorithm but not necessarily an efficient one. They also showed that the impact of the hindering effect on obscuring genetic differences can obstruct the Baldwinian search's self-adapting ability. In chapter 7, the evolution metaphor was applied to self-adapt the learning strategy in a hybrid and tested using the self-adaptive hybrid algorithm of chapter 6 and the adaptive staged hybrid (ASH) algorithm. The results of the experiments showed that self-adapting the learning strategy can be beneficial. It can improve the search ability of finding solutions of high quality and can accelerate the search.

In this chapter, a novel form of hybridisation between an ant-based algorithm and a genetic-local hybrid algorithm is proposed. In this hybrid, an ant colony optimisation algorithm is used to monitor the behaviour of a genetic-local hybrid algorithm in order to dynamically adjust the probabilities of using the genetic operators, the local search operator, its duration, and the learning strategies to adapt the hybrid's performance to a given problem.

This chapter starts by introducing ant colony optimisation. Then, the different combinations of ant-based optimisation techniques and genetic algorithms are reviewed. The basic idea of the proposed hybrid is explained in the section that follows. That section also explains how the pheromone trail metaphor can be applied to adapt the control parameters of a genetic-local hybrid algorithm in order to strike a balance between exploitation and exploration based on the nature of a given problem. This chapter ends by presenting and discussing the results of a set of experiments that compare the ant-based and the evolutionary self-adapting techniques.

8.1 Ant colony optimisation

Ant-based optimisation algorithms are bio-inspired population-based optimisation techniques that have been applied successfully to solve a large number of hard optimisation problems. These search techniques simulate the collective behaviour of ants, which exchange information using a simple form of indirect communication mediated by pheromone formation, known as stigmergy (Dorigo et al. 1999). This form of stigmergtic communication plays a crucial role in ant foraging behaviour. This behaviour is a kind of distributed optimisation mechanism in which each single ant contributes to the finding of the shortest path to food sources. Although a single ant is capable of finding a path between nest and food source, it is the ant colony which finds the shortest path. The stigmergtic communication also explains the ant's ability to adapt to changes in the environment, such as new obstacles interrupting the currently shortest path.

The first ant-based optimisation algorithm was the ant system (Dorigo et al. 1991), which is a model of positive feedback, distributed computations, and a constructive greedy heuristic, to solve combinatorial problems. The ant colony system (Dorigo and Gambardella 1997) is an improvement to the ant system, where the level of exploration undertaken by the ants can be controlled. These variations on the original ant system led to the development of the Ant Colony Optimisation meta-heuristic (ACO) (Dorigo and Di Caro 1999), which describes a class of ant-inspired optimisation algorithms.

Ant colony optimisation algorithms are very effective in solving discrete optimisation problems such as the travelling salesperson problem (TSP) (Dorigo et al. 1991) (Dorigo and Di Caro 1999), the quadratic assignment problem (Maniezzo et al. 2004), vehicle routing (Maniezzo et al. 2004) (Dorigo et al. 1999), sequential ordering (Maniezzo et al. 2004), graph colouring (Shawe-Taylor and Zerovnik 2001), E-learning presentation problem (Semet et al. 2003) and routing in communications networks (Di Caro and Dorigo 1998) (Dorigo and Di Caro 1999). They have also been used to solve real-parameter optimisation problems. Bilchev and Parmee (1995) proposed an ant-based model for continuous space optimisation problems. The continuous neighbourhood was represented by a finite number of directions as a set of vectors starting from a base point. The vectors were evolving according to the ants' fitness. Different algorithms have been proposed as extensions of ACO for continuous search spaces to overcome the difficulties of directly applying the pheromone trail metaphor for continuous spaces. The Pachycondyla **api**calis (API) algorithm (Monmarche et al 2000) and the Continuous Interacting Ant Colony (CIAC) (Dr'eo and Siarry 2002) used some form of direct communication that does not exist in

regular ACO algorithms to optimise continuous functions. Socha (2004) used a normal distribution function, whose centre was updated according the solution of the best fitness value, to represent the pheromone density to solve real-parameters problems. In the Aggregation Pheromone System (APS) (Tsutsui 2004), the pheromone trail was replaced by aggregation pheromone, whose density was represented by a mixture of multivariate normal distributions.

8.2 Ant colony optimisation and genetic algorithms

Genetic algorithms and ant colony optimisation algorithms can be combined to improve the combination's performance in different ways. However, most of the proposed hybrids use only three different ways of hybridisation. The first set of hybrids is based on viewing the genetic algorithm as a global search method and the ant colony algorithm as a local search method. The second set is based on the ability of the genetic algorithms to adapt the control parameters of other techniques. In the last form of intergeneration, some genetic concepts and operators are incorporated into ant colony optimisation algorithms.

Incorporating an ant colony optimisation algorithm as a local search method within a genetic algorithm can improve the search performance. Bilchev and Parmee (1995) used their ant colony model for continuous search spaces to improve the quality of the solutions produced by a genetic algorithm in order to solve a heavily constrained real-world engineering design problem. Chen and Lu (2005) combined a genetic algorithm and an ant colony algorithm to solve the TSP. The hybrid starts with the ACO algorithm and switches to the genetic algorithm using the n optimal results from the ACO algorithm, as an initial population in the case of a decrease in the convergence speed and the diversity of the solutions of the ant algorithm. This decrease indicates that the ACO algorithm reaches a local optimum and utilising a genetic algorithm can help to avoid being trapped in a local optimum. The optimal result of a genetic iteration is used to update the pheromone trail of the ACO algorithm in order to improve the diversity of the solutions of the ACO algorithm. The hybrid switches back to the ACO algorithm if the diversity of the population falls below a specific threshold, where the use of the ACO algorithm can be more effective. The hybrid keeps a list of the n best found solutions and updates its contents at each search iteration.

Since genetic algorithms are in practice a very effective optimisation technique, it has been incorporated within ant colony optimisation algorithms to optimise their control parameters, which are characterised by being highly problem specific and dependent on the required solution accuracy (Caertner and Clark 2005). A genetic algorithm can be applied

to optimise the control parameters of ant colony optimisation algorithms in a variety of ways. The ASGA algorithm (White et al. 1998) was the first algorithm that used a genetic algorithm to evolve the control parameters of an ant-based algorithm. The parameters controlling the sensitivity to the pheromone concentration, and the sensitivity to the cost of a network link, were evolved to solve three path finding problems in networks. Botee and Bonabeau (1998) used a genetic algorithm as a meta-algorithm to find the best set of 11 control parameters of an ant colony optimisation algorithm to solve the TSP. Pilat and White (2002) used a genetic algorithm to evolve the control parameters of an ant colony optimisation algorithm to solve TSP at two different levels. A genetic algorithm was used as a meta-algorithm to evolve the control parameters of the ant colony system algorithm at a global level. The genetic algorithm was also used to evolve a population of genetically modified ants with their own control parameters encoded into their chromosomes. The encoded control parameters were used at an ant level. In their algorithm to plan a path for unmanned robotic vehicle in combat mission, Sauter et al. (2002) used genetic algorithms for automatically tuning the behaviour of the pheromone equation. Caertner and Clark (2005) also proposed a hybrid algorithm, where genetic algorithms were used to evolve ants with their encoded control parameters, in order to find optimal values of these parameters based on the state of ant search to solve the TSP.

Genetic models and operators can incorporated in many ways into ant colony optimisation algorithms to improve their performance. Different selection mechanisms that are used in genetic algorithms were implemented and tested with an ant colony optimisation algorithm to dynamically optimise the structure of an online teaching website based on the recommended structure, the collective experience of students, and the particularities of each student (Semet et al. 2003). Fitness ranking together with the steady state evolutionary model was incorporated into the aggregation pheromone system (APS) to solve a set of continuous uni-modal and multimodal problems (Tsutsui et al. 2005).

8.3 Ant optimisation and genetic-local hybrid self-adaptation

The success of a hybrid algorithm in solving a given problem efficiently depends on its success in achieving a balance between exploration and exploitation (see chapter 2 and 6). The appropriate balance of exploration and exploitation required for good performance depends on the amount of diversity in the population, the details of the genetic and the local operators, the learning strategies and the problem to be optimised. This balance is usually achieved by finding an optimal set of the hybrid's control parameters for a given problem. The use of a mechanism to dynamically identify the effectiveness of

different genetic and local operators and learning strategies for the problem currently being worked on can help to improve the hybrid's performance. Knowledge about previous and current solutions, the operators, and the learning strategies used to produce them, can be utilised as a base to identify the strengths and the weakness of these operators and strategies.

The basic idea of the proposed hybrid is that, as the search progresses, the effectiveness of the genetic operators, the local search method, the duration of local search, and the learning strategies, on the performance of a hybrid genetic algorithm in dealing with the current problem can be learnt by using an ant-based algorithm as a reinforcement learning approach. The pheromone trail metaphor can be used to accumulate the experience of the past solutions on the efficiency and the effectiveness of using different operators to find a solution of the current problem.

Pheromone trail behaviour can be applied to solve the problem of dynamically adjusting the probabilities of using the different genetic and local operators and learning strategies.

A population or a colony of ants collectively searches for a sequence of genetic operations, local search operator with a suitable duration, and a learning strategy, that produces an effective and efficient solutions to the problem under consideration. The search space and the neighbourhood notion of the problem of adapting the performance of a genetic-local hybrid to a given problem can be viewed as shown in figure 8.1. Each ant performs a sequence of local moves between the different states of its search task in order to find a sequence of operations that improves the solutions in efficient way. Each state of the problem's search space has a complementary state. For example, the complementary state of the "crossover" state is the "no crossover" state, while the complementary of the "Lamarckian" state is the "Baldwinian" state.

An ant moves through adjacent states starting from the selection state, which is the only state without a complementary state, and ending its tour with either the "Lamarckian" or the "Baldwinian" state. Each movement of an ant is accompanied by performing one of the two alternatives on a solution. The path that is followed by an ant defines the sequence of genetic operations, local operation, its duration, and the learning strategy that are applied on a solution. This path is assigned a merit score, which is equal to the fitness improvement in the solution as a result of performing this sequence of operations. At the end of the ant's tour, it releases an amount of pheromone on the edges of the path it used to build a solution based on the merit score of the tour. The density of pheromone on the paths that lead to

high improvements in solutions' fitness will be higher than the others which lead to less or no improvement. This will probably encourage other ants to follow the paths with a high density of pheromone. This means that the sequence of genetic and local operators and learning strategies that lead to solution improvements will be preferred by most of the new candidate solutions. These preferred sequences of operations will be dynamically built based on the fitness landscape of a given problem. This can promote competition amongst the different operators and learning strategies based on its ability to improve the fitness. It can also promote cooperation between the different operators and learning strategies in order to discover more effective sequences of operations. This technique can produce a hybrid genetic algorithm that able to adapt itself to a given problem without the need for external control.

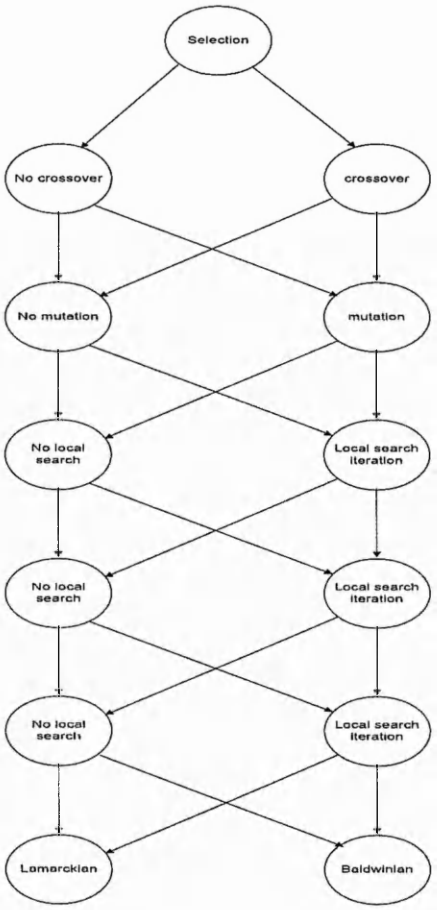

Figure 8.1: The Search Space and the Neighbourhood Notion.

An ant selects the next state from its adjacent states using a probabilistic decision policy. An ant decides to move from its current state to one of the available next two states, which

will be referred to as the Next Do state or the Next Alternative state. The decision policy, as given in equation 8.1, is based on the density of pheromone on the two branches that connect the current state to these states.

$$P_{C-NDo} = \frac{\tau_{C-NDo}}{\tau_{C-NDo} + \tau_{C-NAlter}}$$
$$P_{C-NAlter} = 1 - P_{C-NDo} \tag{8.1}$$

where P_{C-NDo} is the probability of moving from the current state to the Next Do state, $P_{C-NAlter}$ is the probability of moving from the current state to the Next Alterative state, τ_{C-NDo} is the trail density on the edge connecting the current state to the Next Do state, and $\tau_{C-NAlter}$ is the trail density on the edge connecting the current state to the Next Alterative state. For example, if an ant is at the "crossover" state, the probability of moving to the "mutation" state is given by equation 8.2.

$$P_{X-M} = \frac{\tau_{X-M}}{\tau_{X-M} + \tau_{X-NM}} \tag{8.2}$$

where P_{X-M} is the probability of moving from the "crossover" state to the "mutation" state, τ_{X-M} is the trail density on the edge connecting the "crossover" state to the "mutation" state, and τ_{X-NM} is the trail density on the edge connecting the "crossover" state to the "no mutation" state.

Initially all the edges of the possible operations paths are assigned an equal trail density. Therefore, all the adjacent states have an equal opportunity to be visited. All the ants start from the selection state. After the genetic algorithm performs the selection operation, each ant is randomly assigned an individual of the mating pool. That ant will decide on the sequence of operations that individual should perform. The decision is taken locally based on the current ant's state and using the decision policy given in equation 8.1. The ant's tour ends by choosing one of the available learning strategies. At the end of that tour, the ant deposits an amount of pheromone on the edges of the path it followed. The amount of pheromone deposited is made equal to the improvement in the fitness of the associated solution. This can induce the ants towards promising search regions of effective sequences of operators. The change in trail density on each edge of the followed path is given by equation 8.3.

$$\Delta\tau_{(i,C-N)} = \begin{cases} \Delta fitness_i & if \quad \Delta fitness > 0 \\ 0 & otherwise \end{cases} \tag{8.3}$$

where $\Delta\tau_{(i,C-N)}$ is the change in the trail density of the edge connecting state C to state N as a result of following the path constructed by ant i. The aim of rewarding sequences of

operations that produce an improvement in fitness and not penalising paths that reduces the performance is to encourage exploring the search space.

The algorithm updates the pheromone at the end of the ant colony iteration (i.e. after every member of the colony has completed its tour), which is known as offline updating, to bias the search from a global perspective. The amount of pheromone on the edge connecting the *C* state with the *N* state after updating it according to the results of the ant tour is given by equation 8.4.

$$\tau_{(i+1,C-N)} = \tau_{(i,C-N)} + \Delta\tau_{(i,C-N)} \tag{8.4}$$

8.4 Experiments

For the purpose of evaluating the proposed **Ant**-based Self-Adaptive Hybrid Genetic (AntSAHG) algorithm, its performance was compared with an Evolutionary Self-Adaptive Hybrid Genetic (ESAHG) algorithm, which uses evolution to select the genetic operators, the local operator, its duration and the learning strategy that should be performed on each individual. For each operation (strategy), a digital bit is encoded into the individual's genetic structure which determines whether to perform that operation (use that strategy) or its alternative (see figure 8.1). The duration of local search is represented by two bits that specify the number of local iterations (chapter 6).

The quality of the solutions produced by each algorithm was used as the main measure of the algorithm's performance. The percentage of experiments that converged to the global optimum was used as an indication of the ability of the algorithms to produce high quality solutions. The performance was compared using different population sizes in order to evaluate the ability of these algorithms to adapt to different search environments. The performance of the two algorithms is compared in terms of the speed of finding a global optimum. These algorithms were also evaluated in terms of their ability to adapt to different fitness landscapes and population sizes.

A set of test functions has been chosen to evaluate the performance of the two self-adaptive algorithms. Five test functions have been used as a test suite. This test functions suite includes the 20-dimensional ellipsoidal (Deb et. al 2002), the 20-dimensional Rastrigin (Törn and Zilinskas 1989), the 20-dimensional Schwefel (Mühlenbein et. al 1991), the 20-dimensional Griewank (Mühlenbein et. al 1991), and the 20-dimensional ridge (Deb et. al 2002) test functions. The reader can refer to chapter 6 for more details on the first four functions.

However, the n-dimensional inverted ridge function (Deb et al. 2002) is defined as

$$f(x) = -\sum_{i=1}^{n}(\sum_{j=1}^{i} x_j)^2 \qquad \text{For } -64 \le x_i \le 64 \qquad (8..5)$$

The ridge function is a unimodal function, but it has an epistasis among its variables.

The hybrids use the simple elitist genetic algorithm with binary tournament selection, two-point crossover operator, and a mutation operator with a probability of 0.01. The hybrids use a local search operator, which combines the steepest descent method and Brent's method to estimate the best step size to climb to the local optimum from the current position in the basin of attraction (Press et al. 1993). In addition to that, an adaptive initial step size based on the changes in the standard deviation of the population fitness was used. The use of such an adaptive step size can add an exploring role to the local search method at the early stages of search. This adaptive initial step size can improve the Baldwinian search since it can improve the genetic sampling ability at the early stages and can combat the hindering effect as the search approaches the fitness-convergence-state.

Population sizes of 100, 150, 200, 250 and 300 were used to optimise the test functions using the two self-adaptive hybrids. Each variable was represented by a string of 10 bits. The stopping criterion for all experiments was a maximum number of function evaluations. The value of this parameter was set to 5000 times the population size. Each experiment was repeated 50 times.

In the AntSAHG algorithm, the number of ants was set equal to the number of individuals of the genetic population. The amount of pheromone released was made equal to the fitness improvement, as given in equation 8.4, in order to ensure fair comparison with the evolutionary self-adaptive technique, which uses the individual's fitness to assess the effectiveness of a control parameter in solving a given problem (see chapter 6). However, the ant algorithm divides the whole tour into two stages. The first one is the genetic stage, where the ants decide on the genetic operators to apply on its associated solution. The second one is the learning stage, where the ants decide on the local operator, its duration, and the learning strategy. In order to evaluate these stages fairly, each stage is evaluated separately due to the big differences in the number of function evaluations used at these stages.

Instead of the evaporation mechanism, the AntSAHG algorithm adaptively modifies the trail density of all the possible paths to ensure that the probability of each of the alternative operations or strategies does not exceed a specific threshold. In the case of exceeding this

threshold an equal amount of pheromone is added to all the possible paths. The value of the threshold probability was set to 0.999 and the amount of pheromone added in the case of exceeding this threshold was set 0.01 of the current highest trail density.

Initially all the edges of the possible operations paths are assigned an equal trail density, which was set to the absolute value of the average fitness of the initial genetic population.

8.4.1 Search effectiveness and efficiency

The percentages of times each hybrid algorithm found a global optimum using different population sizes were compared. These percentages were used to evaluate the effectiveness of the two self-adaptive mechanisms in solving the test problems.

The two adaptive hybrids have been used to optimise the ellipsoidal and the Griewank test functions in experiments 8.1 and 8.2, respectively. The results of these experiments, not shown here, showed that both the self-adaptive techniques were able to find the exact global optimum of each function in every experiment. This clearly shows the effectiveness of both algorithms in solving these types of problems, which can obstruct the self-adaptive ability of the Baldwinian search (chapter 6). The combination of the adaptive initial step size of local search and adaptive ability of the two hybrids can explain the improvement in the hybrids' performance.

However, the results of comparing the speed of finding the global optimum of the ellipsoidal function, as shown in figure 8.2, show that the AntSAHG algorithm was slightly faster than the ESAHG algorithm. The graphs for the results of comparing the convergence speed to the global optimum of the Griewank function of both adapting techniques are shown in figure 8.3. The plots show that the AntSAHG algorithm was much faster than the ESAHG algorithm. The difference in convergence speed can be explained based on the fact that the genetic algorithm as a global search method needs a number of genetic iterations to evolve the control parameters in order to find the optimal ones. However, the ant-based algorithm needs a number of ant iterations to find these values. Each genetic iteration consumes a number of function evaluations which is equal to the population size times the number of function evaluation consumed in a single ant iteration.

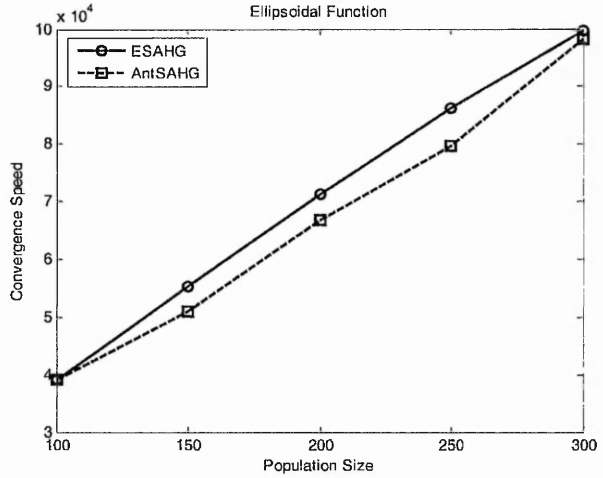

Figure 8.2: The Speed of Finding the Global Optimum of the Ellipsoidal Problem (Experiment 8.1).

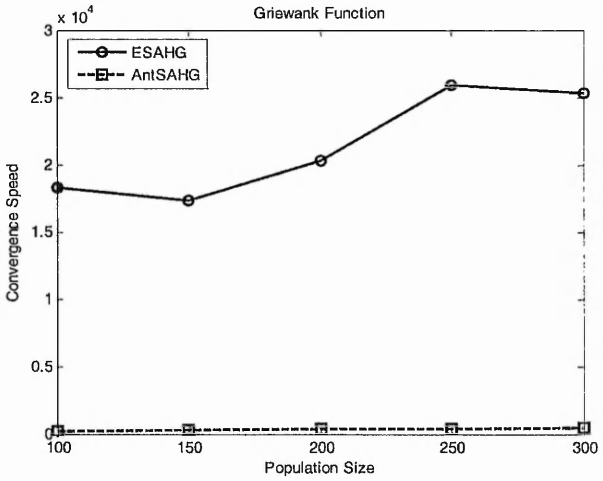

Figure 8.3: The Speed of Finding the Global Optimum of the Griewank Function (Experiment 8.2).

Figure 8.4 compares the percentage of times that the two self-adaptive hybrid algorithms found the global optimum of the Schwefel function in experiment 8.3. The graphs show that the AntSAHG hybrid algorithm was able to find the global optimum of the Schwefel function more often than the ESAHG hybrid algorithm. This can be explained by the fact that the ESAHG hybrid algorithm uses the fitness of an individual for evaluating both the quality of solutions and effectiveness of different operators and learning strategies in producing these solutions. On the other hand, the AntSAHG uses the improvements in fitness to judge the effectiveness of operators and strategies and at the same time the

genetic algorithm uses the fitness to evaluate the solution's quality. The AntSAHG hybrid algorithm can easily discriminate between sequences of operators that improve the performance and sequences that do not improve it, whereas, the ESAHG algorithm cannot distinguish between them.

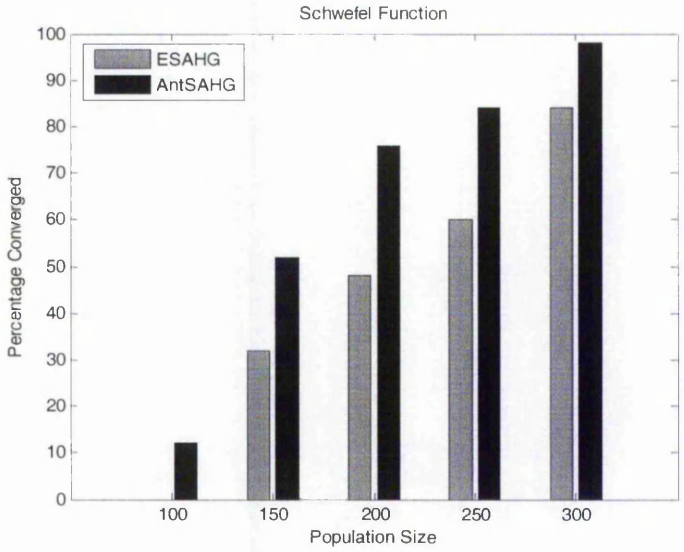

Figure 8.4: The Ability to Find the Global Optimum of the Schwefel Function (Experiment 8.3).

Figure 8.5 compares the convergence speed to the global optimum of the Schwefel function of both self-adaptive algorithms. These graphs show that the ESAHG algorithm was much faster than the AntSAHG algorithm in finding the global optimum of this function. However, the AntSAHG was able to find that optimum more frequently than the ESAHG algorithm.

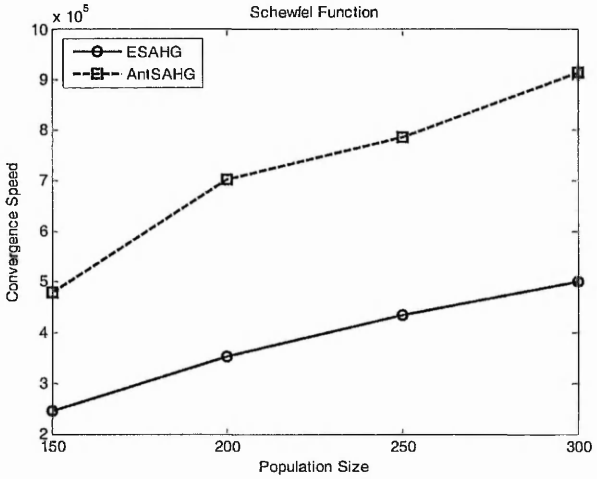

Figure 8.5: The Speed of Finding the Global Optimum of the Schwefel Function (Experiment 8.3).

The results of experiment 8.4, which aimed to optimise the Rastrigin function, showed that the AntSAHG algorithm outperformed the ESAHG algorithm in terms of the percentage that converged to the global optimum using different population sizes, as depicted in figure 8.6. The ESAHG was unable to find the global of the Rastrigin function of most of the experiments in contrast to the AntSAHG algorithm. This difference can be explained based on the fact the evolutionary self-adaptive behaviour can lead to the disappearance of useful genes of some of the control parameters and can face some difficulties in restoring them (see chapter 6). However, by ensuring that the probability of selecting one of the two alternatives does not exceed a threshold value in the AntSAHG algorithm, there is always a chance to select operations that did not improve the solutions' fitness in the past search iterations. This enables the AntSAHG to escape local optima and improves its ability to recover from premature convergence. The nature of the fitness landscape of the Rastrigin function, where the optima are close to each other, makes the AntSAHG algorithm able to recover from premature convergence. However, such a recovery is more difficult in the case of the Schwefel function, where the second best optimum is far from the global optimum, once the whole population has converged to a non-global optimum.

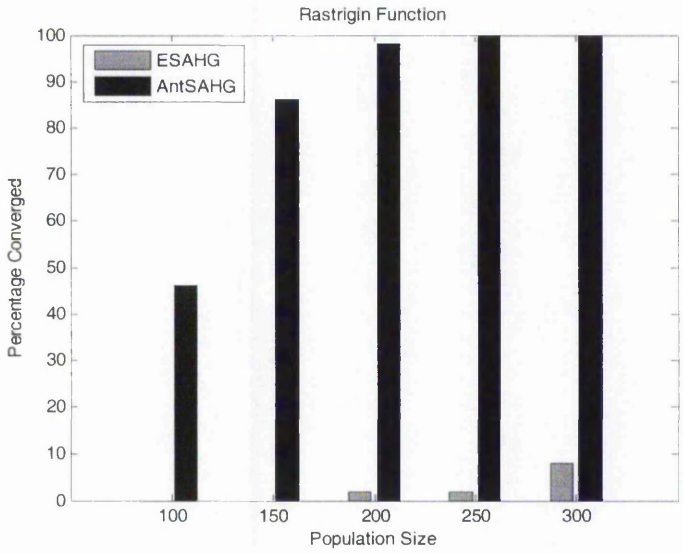

Figure 8.6: The Ability to Find the Global Optimum of the Rastrigin Function (Experiment 8.4).

The results of applying the two self-adaptive techniques to optimise the ridge function in experiment 8.5 are shown in figure 8.7. Here, it is also clear that the ant-based self-adaptive technique outperforms the evolutionary self-adaptive technique. The correlation between the fitness function variables, epistasis, makes the ridge function difficult to be solved using the genetic algorithm. The use of evolutionary self-adaptive mechanism can aggravate this problem due to the introduced correlation between the different control parameters and the fitness function variables. The use of an ant-based algorithm, on the other hand, does not add any correlation to the genetic algorithm.

The ESAHG algorithm was able to find the global optimum of the Griewank function in all the conducted experiments despite of being characterised by having a correlation between its variables. This correlation is a result of the product term (chapter 6) which decreases as the number of variables increases. The increase in the number of variables makes the function surface flat and easy to be solved.

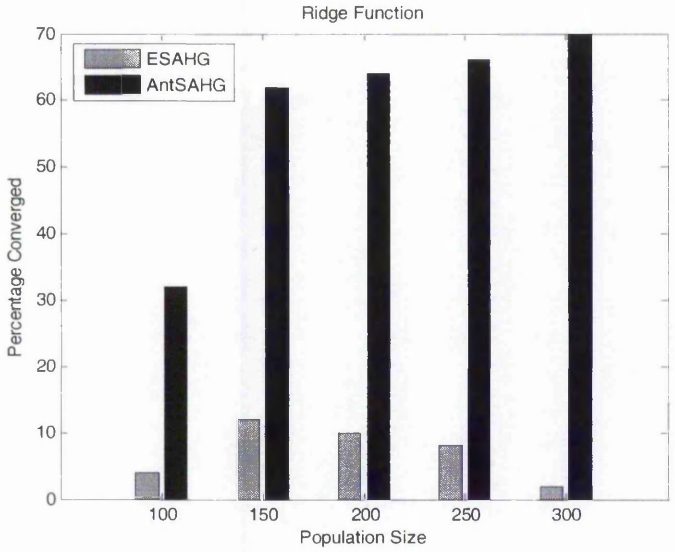

Figure 8.7: The Ability to Find the Global Optimum of the Ridge Function (Experiment 8.5).

The differences in the number times between the AntSAHG and the ESAHG algorithms in finding the global optimum for both the ridge and the Rastrigin functions eliminate the need for comparing their convergence speed.

8.4.2 The ability of the AntSAHG algorithm to adapt to different environments

The previous experiments clearly show that the AntSAHG was able to adapt to different problems using different population sizes. However, this ability was also evaluated through monitoring the changes in the probabilities of using different operators and strategies over time. The graphs of the changes in these probabilities for the five test functions are presented and discussed in this section. The ability of the AntSAHG algorithm to adapt to different population sizes is evaluated through monitoring the changes in the operators' probabilities when used to solve the ridge function using different population sizes. Each graph in the following figures represents the average of 50 experiments.

Figure 8.8 shows the changes in the probabilities of using the crossover operator, the mutation operator, the local search operator for the first iteration and the Baldwinian learning strategy, when used to solve the ridge function using population sizes of 100, 200 and 300. These graphs in general show that these probabilities followed different trajectories depending on the population size used. The graphs of the probability of

crossover show that in the early stages of search, its value dramatically increased. The maximum value of this increase is a function of the population size. The diversity in large population sizes makes the use the crossover operator beneficial. This can explain the trend of the increase in the probability of the crossover operator as the population size increases. The graphs of the probability of the first local iteration and the probability of using the Baldwinian approach show clearly the relation between these two control parameters. The increase in the probability of the first local search iteration is accompanied with a decrease in the probability of using the Baldwinian approach. This is expected since the use of small probabilities of local search can help to combat the hindering effect associated with the Baldwinian learning approach (see chapter 4).

The graphs of the mutation operator probabilities show that these probabilities increase as the population size increases. This is in contrast to the expected behaviour which is based on the fact that for small population sizes the use of high mutation rate in the pure genetic algorithms is more beneficial. However, since the relationship between control parameter values and search performance is complex, not completely understood, and problem dependent (Eiben et al. 1999), the interactions between these parameters can make such a trend beneficial as they were able to produce good performance.

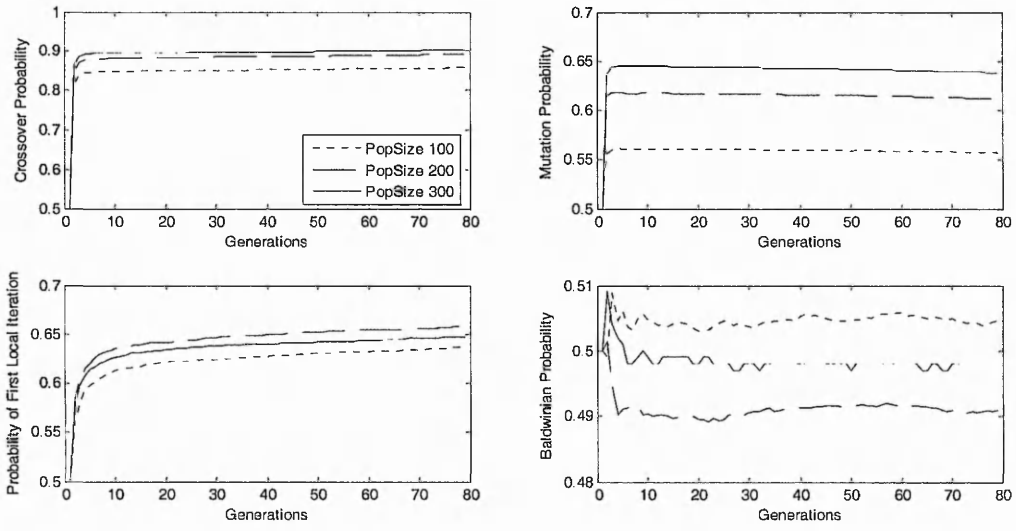

Figure 8.8: The Ability to Adapt to Different Population Sizes (Experiment 8.5).

The plots in figure 8.9 show the changes in the different operators' probabilities over time for different test functions using a population size of 150. These graphs clearly show that the operators' probabilities follow different paths depending on the fitness landscape of the

solved problem. The graphs of the Schwefel and the Rastrigin functions are similar. On the other hand, there are similarities in the graphs of the ellipsoidal and the Griewank functions. This is due to the similarities in each group of these functions. The Schwefel and the Rastrigin are both multimodal functions. However, the 20-dimensional Griewank function is similar to the 20-dimensional ellipsoidal function. These graphs also show that there is always a chance to choose any of the hybrid's operators and strategies since none of these probabilities reaches 1 or zero. This can make the AntSAHG algorithm suitable for optimising problems whose fitness changes with time.

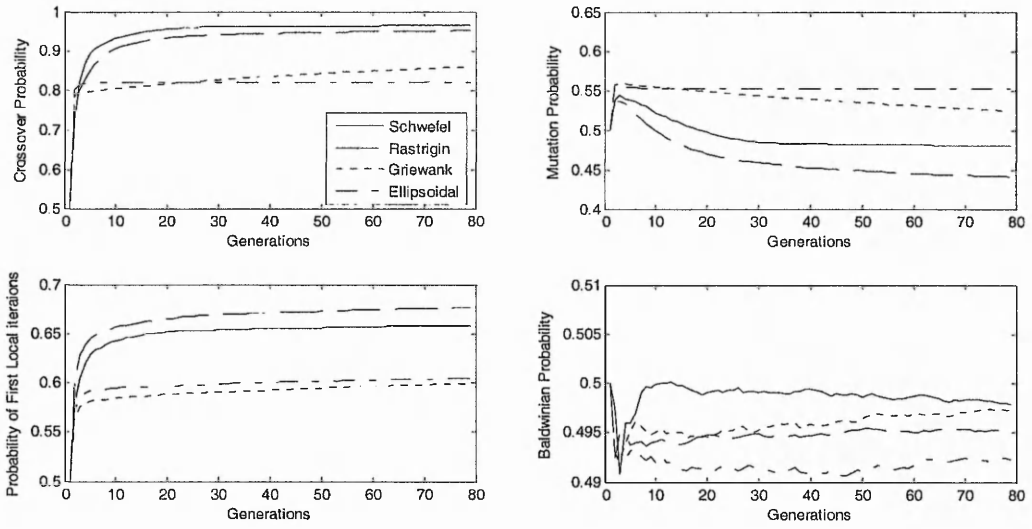

Figure 8.9: The Ability to Adapt to Different Optimisation Problems (Experiments 8.1, 8.2, 8.3 and 8.4).

The experiments conducted clearly show that the use of the pheromone trail metaphor to utilise the experience of the past solutions for online learning of the effectiveness of the different combinations of operators and learning strategies in solving a given problem is effective. The ant-based self-adaptive mechanism was able find high quality solutions for the test problems. It outperformed the evolutionary self-adaptive algorithm in terms of the solution quality and the convergence speed. The experiments suggested the suitability of the AntSAHG algorithm for dynamic environments.

Chapter 9 Conclusions and further work

Hybrid genetic algorithms have received significant interest in recent years and are being increasingly used to solve real-world problems. A genetic algorithm is able to incorporate other techniques within its framework to produce a hybrid that reaps the best from the combination. Incorporating a local search method within a genetic algorithm can improve the search performance on the condition that their roles cooperate to achieve the optimisation goal. There is an opportunity in hybrid optimisation to capture the best of both schemes. This opportunity depends on the design details of the hybrid genetic algorithm. There are several issues that need to be taken into consideration when designing a hybrid genetic algorithm.

The main aim of this thesis is to investigate these design issues. The approach followed was that through designing, developing and testing hybrid genetic algorithms based on the available knowledge of these hybrid issues, new key features and relations can be discovered. The discovered relations and features can be used to develop better hybrids which, in turn, can uncover new relations and features. The goal was to develop a hybrid genetic algorithm that employs learning to direct its search operations and to adapt its control parameters to find high quality solutions for a wide range of optimisation problems efficiently through evolving the hybrid's design based on the analysis of the search's behaviour.

In this chapter, a summary of the research findings is given and the main contributions of the thesis are evaluated in some depth. This will also include the evaluation of this thesis and suggestions for experiments that are needed and further development of some methods. Finally, a section on further work describes key directions of interesting further study to the research in the thesis.

9.1 Research findings and contributions

The research presented in this thesis has contributed towards an improved understanding of hybrid genetic algorithm design issues and their effect on the hybrid's performance. This research demonstrated the direct and indirect influences of the design choices on the utilisation of the search time. It has shed some light on the relations between the design choices and their effect on improving the hybrid's performance in terms of solutions quality, convergence speed, and population size requirements.

This research also suggested and developed many solutions to improve the effectiveness and the efficiency of the hybrid search. Those solutions are based on the richness of the genetic information, which can be utilised to improve the hybrid's performance. The first approach was to develop a search method that can utilise some of the genetic information in order to gain the efficiency of the Lamarckian search with minimum interference with the genetic schema processing. The second approach was to utilise the genetic information to optimise the hybrid search time through adapting the duration of the local search method while the hybrid seeks the global optimum. The third approach was to make use of the genetic information in order to enable the hybrid to learn the effectiveness of the different learning strategies in dealing with the current problem online. The evolution metaphor was applied as a mechanism to utilise the genetic information in the second and the third approach. The forth solution was to apply the pheromone trail metaphor as a mechanism to utilise the genetic information in order to produce an effective, efficient and control parameter-less hybrid genetic algorithm.

The results of the investigations and the contributions of this research are discussed in the following subsections. These subsections will also evaluate the conducted investigations in terms of achieving their goals, the need for more experiments, and point to areas where more investigations are required.

9.1.1 Duration, probability of local search, learning strategy and hybrid's performance

The in-dept review of hybrid genetic algorithms, the analysis, and the experiments conducted in the third and the fourth chapters, help to reveal some relations between the duration of local search, its probability, the learning strategy used, and the hybrid's performance. They emphasise the effect of incorporating a local search method within a genetic algorithm on overcoming some of the obstacles that arise as a result of using finite population sizes. This has been illustrated through investigating the combined effect of the probability of local search and the learning strategy used on the population size requirements of hybrid genetic algorithms. The investigation demonstrates that the minimum population size required for a pure genetic algorithm can be reduced by incorporating a local search method, which influences both the standard deviation of the population and the signal difference between the best and second-best solutions.

The investigations conducted in chapters 3 and 4 show that the two main drawbacks of the basic learning models can be combated through controlling the duration and the probability of local search. The hindering effect associated with the Baldwinian learning approach can

be alleviated by controlling the duration and the probability of local search. The diversity limitation associated with the Lamarckian learning model can also be combated through using appropriate durations and probabilities of local search.

The combination of high probabilities of local search and long durations can aggravate the hindering effect, which makes reliance of the selection operator on the acquired fitness alone insufficient in directing the search towards the global optimum. However, moving into the other direction of using short durations or small probabilities can help to alleviate this problem. When the probability of local search is small, the probability of applying a local search on the same solution in consecutive local iterations is significantly small, giving the algorithm a better chance to distinguish between innate and acquired fitness. The use of short durations of local search can delay the fitness-convergence state. This enables the algorithm to find a solution very near the global optimum.

However, the use of short durations may not improve the sampling ability of the global genetic algorithms, in contrast to long durations, which can improve it. This means that the use of a local search method with short durations cannot help the genetic algorithm to recover from sampling errors and as a result can face premature convergence. The solution to this problem can be the use of an adaptive duration of local search since long durations can cause problems only at the fitness-convergence state and they can improve the genetic algorithm's sampling ability before reaching this state. The adaptive mechanism should enable the Baldwinian search to use long durations at the early stages of the search in order to direct the search towards the most promising search regions. It should also enable the search to use very short durations of local search as the population converged to the basin of attraction of the global optimum in order to find a solution very near the global optimum. A simple adaptive rule for the duration of local search is to make the duration of local search proportional to the variation in the population fitness so that the algorithm uses the shortest duration as the population reaches the fitness-convergence state while it uses the longest duration at the start of the genetic search.

The investigation into the hybrid design issues helps to uncover the relation between the Lamarckian learning approach, the duration, and the probability of local search. The investigation shows that the diversity loss due to the use of the Lamarckian approach can be significantly reduced by using a very short durations and small probabilities of local search. The use of a complete (i.e. exhaustive) local search can map the whole population to local optima of the search space which can badly affect the population diversity in the case that the basin of attraction of the global optimum is not represented in the population. In this

case, the diversity limitation can significantly reduce the chances of redirecting the search towards the global optimum. However, the diversity loss as a result of using short durations is limited compared with that of the complete local search since using short durations maps the population to new points and not to the local optima.

Using a small probability of local search also can help to solve problems without causing diversity loss since the local search will modify the genetic structure of a small fraction of the population only. In contrast, the use of high probabilities can affect the genetic structure of a large number of individuals in the population, which can disrupt the genetic schema processing, and, hence, might result in premature convergence.

In contrast to the Baldwinian search, the use of short durations at the early stages and long durations at the latter stages of the Lamarckian search can be beneficial. Any technique to adapt the duration of local search should take into account the learning strategy used.

In addition to the influence of the interactions between the duration, probability of local search and the learning strategy on the hybrid's performance, the duration and probability of local search have a direct effect on the exploring ability of the hybrid. The use of short durations and small probabilities of local search increases the chances of the global genetic algorithm effectively exploring the search space.

The investigations also showed that using a frequency of local search, i.e. the number of uninterrupted genetic iterations before performing a local iteration (Espinoza et al. 2001), with a value of more than 1 can alleviate the hindering effect since that enables the algorithm to discriminate between acquired and innate fitness.

These findings guided the research to investigate three different possible ways that can improve the hybrid's performance. The first way is to utilise the genetic search information to benefit from the efficiency of Lamarckian learning approach without sacrificing the solution's quality. This can be achieved by minimising the conflict between the genetic schema processing and the utilisation of local knowledge. The second possible way is to find a mechanism that adapts the durations and/or the probabilities of the local search method according to current state of the search and the learning strategy used. The third way is to utilise the search information to change the learning strategy while the search progresses as a mechanism to strike a balance between exploration and exploitation.

9.1.2 Avoiding interference with the genetic search

The richness of the genetic information can be utilised to improve the solution quality in accordance with genetic schema processing and using minimum resources. A search method was proposed as an example of a genetic information utilisation mechanism. The proposed search method is a probabilistic method that works on the genotype space by making use of a group of the current population of solutions to estimate the structure of the improved solution. In this way, it aims to make use of some of the valuable genetic search information, which is inherently available in the gene pool. It also aims to avoid disrupting the genetic schema processing by improving the solution in accordance with the global genetic search. The modification of the initial solution based on a sub-group of solutions of the genetic population can provide the secondary search method with a partial global view of the problem at hand. Based on this view the search method can produce a solution in the context of the global view captured by the genetic algorithm. The partial global aspect of the search method can be controlled by the sub-group size, the mechanism of selecting the group members, and their effect on the initial solution (i.e. the probability effect). This method is also characterised by its low costs. Its costs are equal to the costs of evolving a solution for a single iteration of the genetic search (i.e. one function evaluation per solution). This can help to minimise the loss of the hybrid's time in the case of any undesirable interference between the two search methods.

The result of evaluating the search method as a secondary method on a set of test functions with different marginal fitness contribution of their genes shows that the proposed method, when used with suitable group size and probability effect, could improve the genetic performance in terms of the population size required, convergence speed, or both, to produce high quality solutions. This improvement in the performance was as a result of the ability of the proposed method to integrate the global genetic search with minimum conflicts. By concentrating on the differences in the population's structures and fitnesses to modify the non-identical genes, the proposed algorithm was able to complement the genetic search even in the case of non-uniformly scaled problems, where the genes converge at different rates. In such problems, the proposed algorithm modifies the genes that converge at a slower rate while the genetic algorithm modifies the others.

The proposed algorithm also showed good performance on uniformly scaled problems as a stand-alone search algorithm. This is in accordance with its basic assumption, which states that each gene in the chromosome contributes uniformly to the fitness of the solution.

The ability of the proposed algorithm to reduce the probability of disrupting the genetic schema processing depends on the chosen samples of the genetic population. Each sample involves the initial solution to be optimised and a selected sub-group of solutions. This sample is used to determine the genes that have not converged yet and, hence, can be modified without any conflicts with the genetic search. Modifying them can also accelerate the search towards the global optimum.

The sizes of these samples and the way of selecting their members have a great impact on the hybrid's performance in terms of the solution quality and the convergence speed. Increasing the sample's size increases its accuracy, however it can also reduce its speed as it improves the search from a global perspective. The size of the sample and its members should be chosen in such a way to provide a partial view of the search space around the initial solution in accordance with the global genetic view. The experiments conducted in chapter 5 showed that the size of the sample and the probability factor depend on the fitness landscape. More investigations are required on the possible ways of deciding on optimal group size and probability factor in order to improve the effectiveness and the efficiency of the search.

9.1.3 Adapting the duration of local search

The investigations show, as mentioned above, that the interactions between the duration, the probability of the local search method, and the learning strategy, have a great impact on the hybrid's performance. The duration and the probability of a local search method have a direct influence on the exploration and exploitation trade-off. However, the learning strategy has an indirect impact on this trade-off through its interactions with these two factors.

Finding a mechanism to adapt either the duration or the probability of local search can help to find an optimal utilisation of the search time. Adapting the duration of the local search, and allowing that control parameter to have the value of 0, will implicitly adapt the probability of local search. The decision was taken to investigate the use of evolution to self-adapt the duration of local search in order to find an optimal utilisation of search time for a given problem. The first reason for choosing the evolution metaphor to self-adapt the duration of local search is to gain insight into the ability of this mechanism, which has successfully applied to self-adapt pure genetic algorithms, to adapt the control parameters related to incorporating a local search method. The second reason is the association of the control parameter with a solution through binding them into the same chromosome, which can help to associate the success or the failure of a control parameter value to a specific

solution or solutions of similar genetic structures. The third reason is the simplicity of this mechanism and its implementation cost, which is low compared to other adaptation techniques.

Applying this adaptation mechanism to optimise the performance of a hybrid genetic algorithm to different search environments has demonstrated its ability to produce an effective search. This mechanism uses the individual's fitness as a metric to evaluate the suitability of the encoded duration of local search for solving a given problem. Selecting individuals based on their fitnesses only can bias the search towards an effective algorithm and cannot guarantee its efficiency since these individuals can consume different numbers of function evaluations to achieve their fitnesses.

In addition to that, the hindering effect can obstruct the ability of Baldwinian search to self-adapt the duration-of-local-search control parameter. The possibility of obstructing this ability increases as the dimensionality of the fitness function increases as it may be easier for the algorithm to optimise a single control parameter than optimising a large number of function variables. The use of a local search method with very small durations can help to alleviate the hindering effect and hence improve the performance of the Baldwinian search in terms of solution quality and convergence speed. The performance of the Baldwinian search can be further improved when the local search duration, which is already encoded into the chromosome, is used alongside the acquired fitness to discriminate between effective solutions.

The co-evolutionary mechanism, where the control parameters and function parameters are treated as two subpopulations, improved the self-adapting ability of the hybrid which was reflected as an improvement in its performance. The co-evolutionary mechanism can accelerate the evolution process of the control parameter compared with the evolutionary mechanism where the chance to modify the genetic structure of the control parameter depends on the ratio of the length of its representation to the length of the whole chromosome.

When the decision was taken to self-adapt the duration of local search as a mechanism to strike a balance between the global genetic and the local search methods, it was expected that the frequency of the local search method could be adapted implicitly though adapting its duration. However, adapting the frequency of local search requires switching from the state where duration of local search for all the individuals of the population has a value of 0

and back to that state. Since the experiments showed that reaching that state was not possible, adapting the duration is not a practical way to adapt the frequency of local search.

9.1.4 Adapting the learning strategy

The learning strategy interacts with the duration and the probability of local search and their interaction influences the hybrid's exploration and exploitation trade-off. A mechanism that can dynamically define the strengths and weaknesses of different learning strategies in dealing with a given problem and its environment based on the past solutions experience in using them can help to improve the hybrid's performance. Such a mechanism can strike a balance between exploitation and exploration through dynamically deciding on the learning strategy which has indirect influence on this balance through its interaction with the duration and the probability of local search.

Based on the richness of the genetic information and for the same reasons of selecting evolution to self-adapt the duration of local search, the evolution metaphor was chosen as a mechanism to make use of the past solutions' experience with learning strategies to decide online on the learning strategy to use to solve a given problem. The aim was to gain some insight into the effectiveness and efficiency of self-adapting the learning strategy. It was intended to explore the effect of using evolution to self-adapt both the learning strategy and the duration of local search on the hybrid's performance.

The experiments conducted illustrate that the use of the self-adaptive learning strategy can be beneficial. It can improve the search ability of finding solutions of high quality and can accelerate the search. The experiments also show that this mechanism was able to adapt to different environments. That was illustrated by testing this mechanism on a set of different test functions using two different adaptive hybrid algorithms and different population sizes. The evolutionary self-adaptive mechanism can promote competition and cooperation between the basic learning models in the direction of improving the search performance.

Combining the evolutionary self-adaptive learning mechanism with the adaptive staged hybrid algorithm produced an algorithm that is faster than the tested fixed learning strategies on most of the tested functions. However, combining the evolutionary self-adaptive local-search-duration with this mechanism produced a slow search algorithm. The combination was able to find the global optimum of the whole set of test functions more often and faster than that of the fixed pure Baldwinian approach.

The slow convergence speed of the algorithm that uses evolution to self-adapt both the learning strategy and the duration of local search can be explained based on the fact that the evolutionary mechanism introduces a strong correlation between genes, i.e. epistasis. Encoding the control parameters into the chromosome means that the fitness of a solution is defined based on the interactions between the genes of the control parameters and the fitness function variables' genes. As the number of the encoded control parameters increases, the complexity of the interactions between their genes and the genes of the fitness variables increases. According to the building block hypothesis, one of the basic requirements for a genetic algorithm to be successful is that there is low epistasis (Beasley et al. 1993b). The existence of a strong correlation can affect the ability of the global genetic algorithm to simultaneously explore both the problem search space and the control space.

9.1.5 Ant-based algorithm to self-adapt the hybrid's control parameters

Based on the findings of the previous investigations and to achieve the goal of this thesis in developing a hybrid algorithm that learns form the available search information to utilise its operators in an effective and efficient way without the need for any forms of human intervention, the research has been redirected to find a mechanism that is able to use the available search information without complicating the task of the genetic search.

The simplicity and the effectiveness of the pheromone trail metaphor as a way to accumulate the experience of the past solutions in solving a given problem, and being applied successfully to solve a large number of hard optimisation problems, make it a strong candidate to be applied to achieve an optimal utilisation of the hybrid's search time.

A simple search space with the neighbourhood notion of the problem of adapting the performance of a genetic-local hybrid to a given problem has been defined (figure 8.1). Based on the defined search space, an ant-based optimisation method was used to find an optimal sequence of genetic operators, a local search operator with a suitable duration, and a learning strategy to solve a given problem.

The results of evaluating the performance of the ant-based self-adaptive and the evolutionary self-adaptive techniques showed the superiority of the ant-based self-adaptive mechanism over the evolutionary self-adaptive mechanism in terms of the solution quality and the convergence speed. The experiments conducted clearly showed the effectiveness of using this mechanism to adapt the hybrid's performance to a given problem. The ant-based mechanism was able to adapt the probabilities of selecting the different operators and

strategies according to the relations between these operators, strategies and their environment. The ant-based self-adaptive mechanism was able find high quality solutions for the test problems in an efficient way. The experiments also suggested the suitability of the ant-based self-adaptive algorithm for dynamic environments as there is always a chance to select any operator or strategy.

9.2 Future Directions

The empirical investigations in this thesis suggest many possible directions for future research.

9.2.1 Avoiding interference with the genetic search

The aim of the proposed algorithm in chapter 5 was to utilise the richness of the genetic information from which local information can be simply extracted in order to enhance the genetic search. The proposed algorithm shed some light on the need for more cooperation between the global genetic algorithm and the local search method in exchanging the available information to produce effective and efficient algorithms. Through such cooperation, the proposed search method was able to utilise the efficiency of the Lamarckian search to find high quality solutions.

The main difficulty of applying the proposed algorithm to solve a problem is how to choose the sub-group size, its members, and the probability factor, that produce the best performance. Chapter 5 suggested the use of a variable group size and setting the values of the probability factor based on that size in accordance with the findings of the experiments of that chapter. Further work is required on finding a mechanism to decide on the members of the sub-group, its size, and the probability factor.

There are other search techniques that may be used as a secondary method to achieve such cooperation and can be more effective and efficient than the proposed method. For example, an ant colony optimisation algorithm can be incorporated as a secondary method using a group of the genetic population to improve the quality of an initial solution. Further work could investigate the effectiveness of incorporating such techniques to avoid the genetic schema processing.

9.2.2 Optimal utilisation of search time

The aim of self-adapting the duration of local search within a hybrid was to adapt the duration, the probability, and frequency of local search, in order to achieve optimal utilisation of the search's time. However, as mentioned above, the experiments showed the

difficulty of adapting the frequency of local search following that mechanism. Due to the great impact of the frequency of local search on the balance between exploration and exploitation, and on minimising the interference between local and global search, future work can begin by finding a mechanism to adapt the frequency of local search based on the available genetic information. Then, the possibilities of combining this mechanism with the effectiveness of the co-evolutionary self-adaptive local-search-duration algorithm in order to produce an efficient algorithm can be investigated.

9.2.3 Ant-based algorithm as a self-adaptive mechanism

There are different possible ways to improve the success of the ant-based algorithm in self-adapting the performance of the genetic-local hybrid to a given problem. The efficiency of this mechanism can be further improved by explicitly introducing the cost of the operators into the criteria for selecting the next operation or strategy. The mechanism's effectiveness can be further improved by hybridising it with the co-evolutionary self-adaptive mechanism. The mechanism can be also extended to decide between different sets of genetic operations and different local search methods to solve a given problem. It can also be applied to solve problems that change with time.

9.2.3.1 Improving the effectiveness and the efficiency

Although the experiments showed the effectiveness and the efficiency of the proposed **Ant**-based **S**elf-**A**daptive **H**ybrid **G**enetic (AntSAHG) algorithm to self-adapt the genetic-local hybrids, its performance can be improved in different ways. There is a possibility of improving the AntSAHG algorithm by using an explicit evaluation of the cost of its operations instead of the implicit evaluation. It also can be improved by using the genetically evolved probabilities of the operations in addition to the pheromone density to decide on an optimal sequence of operations.

In the AntSAHG algorithm proposed in chapter 8, the cost of operations was evaluated implicitly through dividing the ant's tour into two stages. These stages are the genetic stage, which consumes a maximum of one function evaluation, and the learning stage, which usually consumes a large number of function evaluations depending on the local search method and its duration. By making the ant deposit an amount of pheromone in proportion to the fitness improvement at the end of each stage, the algorithm implicitly evaluates the relative cost of the operations.

However, the cost of the operations can be explicitly evaluated through including it in the pheromone released equation. The amount of pheromone deposited by an ant can be made

proportional to the improvement in the fitness of its associated solution, and inversely proportional to the number of function evaluations used to produce this improvement. This can induce the ants towards operations sequences that do not only improve the fitness but improve it in an efficient way. This can improve the efficiency of the AntSAHG algorithm. According to this, the change in trail density on each edge of the followed path is modified as given in equation 9.1.

$$\Delta\tau_{(i,C-N)} = \frac{(\Delta fit_i)^{\alpha}}{(\cos t_i)^{\beta}} \quad \text{if } \Delta fit_i > 0$$
$$\Delta\tau_{C-N} = 0 \quad \text{Otherwise} \tag{9.1}$$

where $\Delta\tau_{(i,C-N)}$ is the change in trail density of the edge connecting state C with state N as a result of following the path constructed by ant i. α is a parameter that controls the relative influence of fitness improvement on the trail density. $cost_i$ is the number of function evaluations consumed through this path. β is a parameter that controls the relative influence of the cost of the fitness improvement on the trail density. Through adjusting α and β, the effectiveness and the efficiency of the search can be controlled.

An investigation in the effectives and the efficiency of introducing the operation's cost in the pheromone equation of the AntSAHG algorithm is required. The stability of that algorithm against the changes in the values of the control parameters introduced also requires exploring.

The other direction that can improve the AntSAHG algorithm is to combine it with the Evolutionary Self-Adaptive Hybrid Genetic (ESAHG) (see chapter 8) algorithm. Instead of relying on the pheromone density only to decide on the next operation to perform, the algorithm can use the evolved genetic probability of that operation in addition to pheromone density. This can mix the global perspective of the genetic evolved probabilities with the local perspective of the ant algorithm. The decision policy, as given in equation 9.2, will be based on the density of pheromone on the two branches that connect the current state to these states and the global genetic evolved probability.

$$P_{C-NDo} = \left(\frac{\tau_{C-NDo}}{\tau_{C-NDo} + \tau_{C-NAlter}} \right)^{\delta} Pg^{\varepsilon}_{C-NDo}$$
$$P_{C-NAlter} = 1 - P_{C-NDo} \tag{9.2}$$

where P_{C-NDo} is the probability of moving from the current state to the Next Do state, $P_{C-NAlter}$ is the probability of moving from the current state to the Next Alterative state, τ_{C-NDo} is the trail density on the edge connecting the current state with the Next Do

state, and $\tau_{C-NAlter}$ is the trail density on the edge connecting the current state with the Next Alterative state. $Pg_{C\text{-}NDo}$ is the global evolved genetic probability of moving from the current state to the Next Do state, δ is the influence of pheromone density on the probability of selection, and ε is the influence of global genetic probability on the probability of selection.

The global genetic probability is the evolved encoded probability of that operation at the population level. This probability can be calculated by dividing the number of individuals whose encoded bit indicates performing this operation by the number of individuals in the population.

Researches on in the effectiveness and the efficiency of combining the evolutionary and the ant-based self-adaptive techniques are required. The influence of the δ and ε control parameters on the hybrid's performance also needs investigations. The impact of explicit use of the cost of operations on the search's behaviour also requires more analysis and study.

9.2.3.2 Deciding on local search methods

Due to the effect of the choice of the local search method on the genetic-local hybrid performance, some hybrid genetic algorithms have relied on the use of a variety of different search methods as local search methods (Magyar et al. 2000) (Krasnogor and Simth 2001) (Ong and Kean 2004). The AntSAHG algorithm can be extended and applied to decide, at run time, on the search local method that can used to locally improve the current solution. This can reduce the probability of employing inappropriate local search methods in a hybrid algorithm and can yield robust and improved search performance.

Figure 9.1 depicts a possible search space for an ant-based algorithm to solve the problem of funding an optimal sequence of genetic operations, local operations and strategies.

9.2.3.3 Searching in changing environments

There are many applications in which the fitness function may change over time (Grefenstette 1992). The ability of a hybrid genetic algorithm to respond to a changing fitness function depends on the diversity in its population and the ability of the mechanism that decides on its operators to adjust itself to the changes in its environment. The experiments in chapter 8 showed that there is always a chance to choose any of the hybrid's operators and strategies in the AntSAHG algorithm and suggested the suitability of the ant-based self-adaptive mechanism for dynamic environments. It would be interesting to explore the applicability of the AntSAHG algorithm to such non-stationary environments.

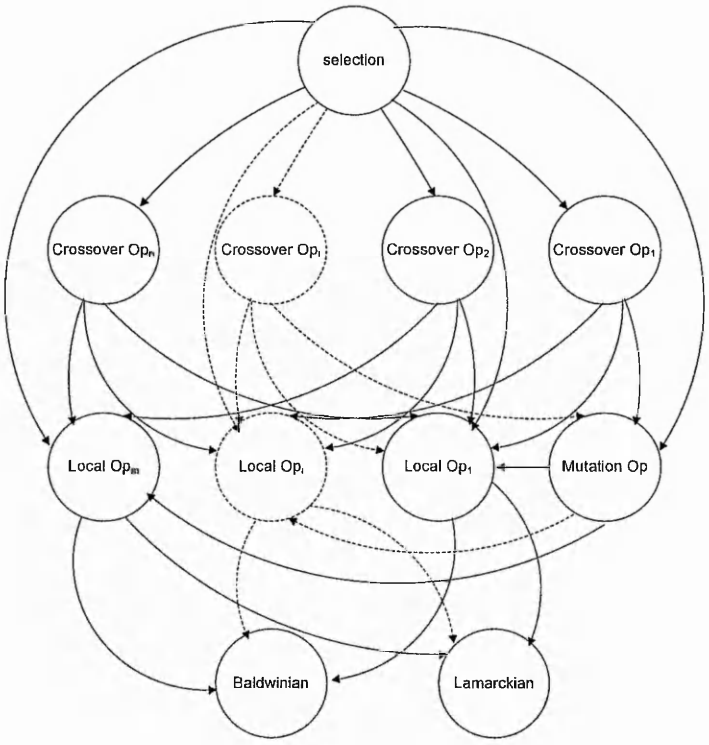

Figure 9.1: Search Space for the Problem of Finding an Optimal Combination of Operators and Strategies

9.3 Summary

The richness of the genetic search information can be utilised in different ways to improve the performance of genetic algorithms. The ability of the genetic algorithm to incorporate different search mechanisms within its framework promotes its cooperation

with different search techniques to make use of the genetic information in order to enhance the search performance. The research of this thesis helps to gain some insight into some of the possible ways of utilising the genetic information effectively and efficiently. It also sheds some light on the key directions of future work that can further improve the hybrid's performance.

The research demonstrated that through adjusting the durations and/or the probability of local search, the hindering effect associated with the pure Baldwinian search, and the problem of disrupting the schema processing of the Lamarckian search, can be alleviated. It also presented an effective model for utilising the genetic information to improve the solution's quality of the Lamarckian search. More investigations are required into the possibilities of using population-based search techniques to improve the effectiveness of the Lamarckian search.

The research also showed the effectiveness of the co-evolutionary self-adaptive local-search-duration mechanism in achieving a balance between exploration and exploitation. It also analysed the difficulties of utilising this technique to produce an efficient search, and the influence of the hindering effect on the self-adaptive ability of the Baldwinian search. Possible ways of improving the self-adaptive ability of the Baldwinian search were suggested and tested. Further work is required to explore the possibility of using other techniques to adapt the frequency of local search and combine it with the evolutionary self-adaptive local-search-duration algorithm in order to improve its efficiency.

The efficiency of the evolutionary self-adaptive learning strategy mechanism in finding high quality solutions is also demonstrated. The increase in the chances of introducing a strong correlation between the control parameters and the variables of the fitness function as the number of the encoded control parameters increases explains the slow convergence speed of the algorithms that self-adapt more than one control parameter.

The ability of the ant-based algorithms to adapt the probabilities of using different operators and strategies of the hybrid algorithm in producing an efficient and effective search was demonstrated. This ability can be enhanced further by introducing the cost of the operations explicitly into the mechanism and combining it with the effective evolutionary self-adaptive mechanism. This mechanism may be applied to decide between different genetic and local search operators. It can also be applied to solve problems in real-world environments that exhibit dynamic and unpredictable characteristics. These ways of

enhancing and extending the success of the ant-based algorithms to solve similar problems require more exploring and investigations.

Bibliography

Abela, J., Abramson, D., Krishnamoorthy, M., Selva, A. D., and Mills, G. (1993): Computing Optimal Schedules for Landing Aircraft, pp. 71-90: *the 12th Conference of the Australian Society for Operations Research*, Adelaide.

Affenzeller, M. (2001): A New Approach to Evolutionary Computation: Segregative Genetic Algorithms (SEGA)". pp. 594-601: *Connectionist Models of Neurons, Learning Procasses, and Artificial Intelligence*, Springer-Verlag.

Anderson, R. W., Fogel, D. B., and Schutz, M. (1997): Other Operators, pp. C3.4:1-C3.4:15. In T. Bäck, D. B. Fogel, and Z. Michalewicz (Eds): *Handbook of Evolutionary Computation*, IOP Publising Ltd and Oxford University Press.

Andre, J., Siarry, P., and Dognon, T. (2000): An Improvement of the Standard Genetic Algorithm Fighting Premature Convergence in Continuous Optimization. *Advances in Engineering Software* **32**, 49 -60.

Angeline, P. J. (1995): Adaptive and Self-adaptive Evolutionary Computations, pp. 152-163. In M. Palaniswami, and Y. Attikiouze (Eds): *Computational Intelligence: A Dynamic Systems Perspective*, IEEE press.

Ankenbrandt, C. A., Buckles, B., Petry, F. E., and Lybanon, M. (1989): Ocean Feature Recognition Using Genetic Algorithms with Fuzzy Fitness Functions, pp. 679-685: *the Third Annual Workshop on Space Operations, Automation and Robotics*, Houston, USA.

Annunziato, M., I., I. B., Elisei, G., Pannicelli, A., and Pizzuti, S. (2002): Adaptive Parameterization of Evolutionary Algorithms Driven by Reproduction and Competition. *Advances in Computational Intelligence and Learning*, 17-134.

Arabas, J., Michalewicz, Z., and Mulawka., J. (1994): GAVaPS - a Genetic Algorithm with Varying Population Size, pp. 306-311: *the First IEEE Conference on Evolutionary Computation*, IEEE, Orlando, USA.

Areibi, S., Moussa, M., and Abdullah, H. (2001): A Comparison Of Genetic/memetic Algorithms And Other Heuristic Search Techniques, pp. 660-666: *International Conference on Artificial Intelligence*, Las Vegas, USA.

Areibi, S., and Vannelli, A. (1994): Advanced Search Techniques for Circuit Partitioning, pp. 77-98. In P. Pardalos, and H. Wolkowicz (Eds): *Quadratic Assignment and Related Problems*.

Areibi, S., and Yang, Z. (2004): Effective Memetic Algorithms for VLSI Design = Genetic Algorithms + Local Search + Multi-Level Clustering. *Evolutionary Computation* **12**, 327 -353

Arena, P., Caponetto, R., Fortuna, I., and Xibilia, M. G. (1993): MLP Optimal Topology via Genetic Algorithms, pp. 670-674. In A. Dobnikar, N. Steele, D. Pearson, and R. F. Albrecht (Eds): *the International Conference on Artificial Neural Nets and Genetic Algorithms*, Springer-Verlag, Portoroz, Slovenia.

Asoh, H., and Mühlenbein, H. (1994): On the Mean Convergence Time of Evolutionary Algorithms Without Selection and Mutation, pp. 88–97. In Y. Davidor, H.-P. Schwefel, and R. Manner (Eds): *Parallel Problem Solving from Nature, PPSN III*, Springer-Verlag, Berlin, Germany.

Bäck, T. (1992): The Interaction of Mutation Rate, Selection, And Self-Adaptation Within a Genetic Algorithm., pp. 85-94. In R. Manner, and B. Manderick (Eds): *Parallel Problem Solving from Nature II*, Elsevier Science, Brussels, Belgium.

Bäck, T., Eiben, A. E., and Vaart, N. L. v. d. (2000): An Empirical Study on GAs without Parameters, pp. 315-324. In M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. Merelo, and H.-P. Schwefel (Eds): *Parallel Problem Solving from Nature PPSN V*, Springer, Amsterdam, The Netherlands.

Bäck, T., Fogel, D. B., and Michalewicz, Z. (1997): *Handbook of Evolutionary Computation*. IOP Publishing and Oxford University Press.

Bäck, T., Hammel, U., and Schwefel, H.-P. (1997): Evolutionary Computation: Comments on the History and Current state. *IEEE Transactions on Evolutionary Computation* **1**, 3-17.

Bäck, T., Heisternmann, J., Kappler, C., and Zamparelli., M. (1996): Evolutionary Algorithms: Support Refueling of Pressurized Water Reactors., pp. 104-108: *the Third IEEE Conference on Evolutionary Computation*, IEEE, Nagoya, Japan.

Bäck, T., and Hoffmeister (1991): Extended Selection Mechanisms in Genetic Algorithms, pp. 92-99: *the Fourth International Conference on Genetic Algorithms and their Application*, Morgan Kaufman, San Mateo, USA.

Bäck, T., Hoffmeister, F., and Schewefel, H. (1991): A Survey of Evolution Strategies, pp. 2–9. In K. Belew, and L. B. Booke (Eds): *the Fourth International Conference on Genetic Algorithms*, Morgan Kaufman, San Mateo, USA.

Bäck, T., and Schütz., M. (1996): Intelligent Mutation Rate Control in Canonical Genetic

Algorithms., pp. 158–167. In W. Ras, and M. Michalewicz (Eds): *In ninth International Syposium on Methodologies for Intelligent Systems*, Springer, Zakopane, Poland.

Baker, J. E. (1985): Adaptive Selection Methods for Genetic Algorithms, pp. 101-111. In J. J. Grefenstette (Ed.): *the First International Conference on Genetic Algorithms and their Applications*, Lawrence Erlbaum Associates, Hillsdale, USA.

Bala, J., De Jong, K. A., Huang, J., Vafaie, H., and Wechsler, H. (1996): Using Learning To Facilitate The Evolution Of Features For Recognizing Visual Concepts. *Evolutionary Computation* **4**, 297–311.

Baldwin, J. (1896): A New Factor In Evolution. *The American Naturalist 30*, 41-451.

Ballester, P., and Carter, J. (2004): An Effective Real-Parameter Genetic Algorithm for Multimodal Optimization, pp. 359-364. In V. I. C. Parmee (Ed.): *Adaptive Computing in Design and Manufacture*, Springer, Bristol, UK.

Baluja, S. (1994): Population-Based Incremental Learning: A Method for Integrating Genetic Search Based Function Optimization and Competitive Learning: *Technical Report: CS-94-163*, Carnegie Mellon University.

Barr, R. S., Golden, B. L., Kelly, J. P., Resende, M. G. C., and Stewart, W. R. (1995): Designing and Reporting On Computational Experiments With Heuristic Methods. *Journal of Heuristics* **1**, 9-32.

Beasley, D., Bull, D. R., R, and Martin, R. (1993a): An Overview Of Genetic Algorithms : Part 1, Fundamentals. *University Computing* **15**, 58-69.

Beasley, D., Bull, D. R., R, and Martin, R. (1993b): An Overview Of Genetic Algorithms : Part 2, Research Topics. *University Computing* **15**, 170-181.

Belew, R. K. (1989): When Both Individuals and Populations Search: Adding Simple Learning To The Genetic Algorithm, pp. 34-41. In H. Schaffer (Ed.): *the Third International Conference on Genetic Algorithms*, Morgan Kaufmann, San Mateo, USA.

Belew, R. K., McInerney, J., and Schraudolph, N. N. (1991): Evolving Networks: Using the Genetic Algorithm with Connectionist Learning, pp. 511-547: *Artificial Life II*, Addison-Wesley, New York, USA.

Besnard, E., Cordier-Lallouet, N., Schmitz, A., Kural, O., and Chen, H. P. (1999): Design/Optimization with Advanced Simulated Annealing: *AIAA Paper No. 99-0186*, American Insitute of Aeronautic and Astronautics.

Beyer, H.-G., and Deb, K. (2001): On Self-adaptive Features in Real-Parameter Evolutionary Algorithms. *IEEE Transactions on Evolutionary Computation* **5**, 250-270.

Bilchev, G., and Parmee, I. C. (1995): The Ant Colony Metaphor for Searching Continuous Design Spaces, pp. 25-39. In T. C. Fogarty (Ed.): *AISB Workshop on Evolutionary Computing*, Springer Verlag, Sheffield, UK.

Bommel, P. v., and Weide, T. P. v. d. (1992): Towards Database Optimization by Evolution, pp. 273-287. In A. K. Marumdar, and N. Prakash (Eds): *the International Conference on Information Systems and Management of Data*, Bangalore, India.

Booker, L. (1987): Improving Search in Genetic Algorithms, pp. 61-73: *Genetic Algorithms and Simulated Annealing*, Pitman.

Botee, H. M., and Bonabeau, E. (1998): Evolving Ant Colony Optimization. *Advanced Complex Systems* **1**, 149-159.

Bridges, C. L., and Goldberg, D. E. (1987): An Analysis of Reproduction and Crossover in a Binary-Coded Genetic Algorithm, pp. 9-13. In J. J. Grefenstette (Ed.): *the Second International Conference on Genetic Algorithms*, Lawrence Erlbaum Associates, Hillsdale, USA.

Buckham, B. J., and Lambert, C. Simulated annealing applications. *Retrieved September 2003 from the World Wide Web: http://www.me.uvic.ca/~zdong/courses/mech620/SA_App.PDF.*

Burdsall, B., and Giraud-Carrier, C. (1997 b): GA-RBF: A Self-Optimising RBF Network, pp. 348-351. In G. Smith, N. Steele, and R. Albrecht (Eds): *the Third International Conference on Artificial Neural Networks and Genetic Algorithms*, Springer-Verlag, Vienna, Austria.

Burdsall, B., and Giraud-Carrier, C. (1997a): Evolving Fuzzy Prototypes for Efficient Data Clustering, pp. 217-223: *Second International ICSC Symposium on Fuzzy Logic and Applications*, Zurich, Switzerland.

Burke, E. K., Elliman, D. G., and Weare, R. F. (1995): A Hybrid Genetic Algorithm for Highly Constrained Timetabling Problems, pp. 605-610. In L. J. Eshelman (Ed.): *the sixth International Conference on Genetic Algorithms*, Morgan Kaufmann Pittsburgh, USA

Cantú-Paz, E. (1998): A Survey of Parallel Genetic Algorithms. *Calculateurs Parallele,*

Reseaux et Systems Repartis **10**, 141-171.

Ceroni, A., Pelikan, M., and Goldberg, D. E. (2001): Convergence-Time Models for the Simple Genetic Algorithm with Finite Population: *IlliGAL technical report 2001028.*

Chaiyaratana, N., and Zalzala, A. M. (2000): Hybridisation of Neural Networks and a Genetic Algorithm for Friction Compensation, pp. 22-29: *The 2000 Congress on Evolutionary Computation*, San Diego, USA.

Chalmers, D. (1990): The Evolution of Learning: An Experiment in Genetic Connectionism, pp. 81–90. In D. Touretzky, J. Elman, T. Sejnowski, and G. Hinton (Eds): *Connectionist Models,1990 Summer School*, Morgan Kaufmann, San Diego, USA.

Chen, M., and Lu, Q. (2005): A Hybrid Model Based on Genetic Algorithm and Ant Colony Algorithm. *Journal of Information & Computational Science* **2**, 647-653.

Chen, Y., and Goldberg, D. (2005): Convergence Time for the Linkage Learning Genetic Algorithms. *Evolutionary computation* **13**, 279-302.

Corno, F., Reorda, M. S., and Squillero, G. (1998): The Selfish Gene Algorithm: A New Evolutionary Optimization Strategy, pp. 349-355 *the 1998 ACM symposium on Applied Computing*, ACM, Atlanta, USA.

Davidor, Y. (1991): *Genetic Algorithms and Robotics: A Heuristic Strategy for Optimization*. World Scientific Publishing.

Davis, L. (1989): Adapting Operator Probabilities in Genetic Algorithms, pp. 61–69: *the Third International Conference on Genetic Algorithms*, Morgan Kaufmann, Fairfax, USA.

Davis, L., and Steenstrup, M. (1987): Genetic Algorithms and Simulated Annealing: An Overview, pp. 1-11: *Genetic Algorithms and Simulated Annealing*, Pitman.

Deb, K. (1997): Limitations of Evolutionary Computation Methods, pp. B2.9. In T. Bäck, D. B. Fogel, and Z. Michalewicz (Eds): *Handbook of Evolutionary Computation*, IOP Publishing and Oxford University Press.

Deb, K. (1999): Evolutionary Algorithms for Multi-Criterion Optimization in Engineering Design, pp. 135-161. In K. Miettinen, P. Neittaanmäki, M. M. Mäkelä, and J. Périaux (Eds): *Evolutionary Algorithms in Engineering and Computer Science (EUROGEN-99)*, Wiley, Finland.

Deb, K., Anand, A., and Joshi, D. (2002): A Computationally Efficient Evolutionary

Algorithm for Real-Parameter Optimization. *Evolutionary Computation* **10**, 371-395.

Deb, K., and Goel, T. (2001): A Hybrid Multi-Objective Evolutionary Approach to Engineering Shape Design, pp. 385 -399. In E. Zitzler, K. Deb, L. Thiele, C. Coello, and D. Corne (Eds): *the First International Conference on Evolutionary Multi-Criterion Optimization*, Zurich, Switzerland.

De Jong, K. A., and Jayshree, S. (1992): Generation Gaps Revisited, pp. 19-28. In L. D. Whitley (Ed.): *Foundations of Genetic Algorithms 2*, Morgan Kaufmann, San Mateo, USA.

De Jong, K. A., Potter, M. A., and Spears, W. (1997): Using Problem Generators to Explore the Effects of Epistasis, pp. 338-345. In T. Bäck (Ed.): *the Seventh International Conference on Genetic Algorithms (ICGA97)*, East Lansing, USA.

De Jong, K. A., and Spears, W. M. (1992): A Formal Analysis of the Role of Multi-Point Crossover in Genetic Algorithms. *Annals of Mathematics and Artificial Intelligence* **5**, Jan-26.

De Jong, K. (1975): An Analysis of the Behavior of a Class of Genetic Adaptive Systems: *Computer and Communication Sciences*, The University of Michigan, Ann Arbor.

De Jong, K. (2005): Genetic algorithms: A 30 year perspective. In L. Booker, S. Forrest, M. Mitchell, and R. Riolo (Eds): *Perspectives on Adaptation in Natural and Artificial Systems*, Oxford University Press.

De Jong, K., and Spears, W. (1993): On The State of Evolutionary Computation, pp. 618-623: *the Fifth International Conference on Genetic Algorithms*, Urbana-Champaign, USA.

Di Caro, G., and Dorigo, M. (1998): AntNet: Distributed Stigmergetic Control for Communications Networks. *Journal of Artificial Intelligence Research* **9**, 317-365.

Digalakis, J. G., and Margaritis, K. G. (2002): An Experimental Study Of Benchmarking Functions For Genetic Algorithms. *International Journal of Computer Mathematics* **79**, 403-416.

Dijk, S. v., Thierens, D., and Berg, M. d. (2004): On the Design and Analysis of Competent Selecto-Recombinative GAs. *Evolutionary Compution.* **12**, 243-67.

Dorigo, M., and De Caro, G. (1999): The Ant Colony Optimization Meta-Heuristic, pp. 11-32. In D. Corne, M. Dorigo, and F. Glover (Eds): *New Ideas in Optimization*, McGraw-Hill.

Dorigo, M., De Caro, G., and Gambardella, L. M. (1999): Ant Algorithms for Discrete Optimization. *Artificial Life* **5**, 137-172.

Dorigo, M., and Gambardella, L. (1997): Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem. *IEEE Transactions on Evolutionary Computation* **1**, 53-66.

Dorigo, M., Maniezzo, V., and Colorni, A. (1991): Positive Feedback as a Search Strategy: *Technical Report No 91-016*, Politecnico di Milano.

Dorigo, M., Maniezzo, V., and Colorni, A. (1996): The Ant System: Optimization by a Colony of Cooperating Agents. *IEEE Transaction on Systems, Man, and Cybernetics-Part B* **26**, 1-13.

Dr'eo, J., and Siarry, P. (2002): A New Ant Colony Algorithm Using the Heterarchical Concept aimed at optimization of multiminima continuous functions, pp. 216-221: *Third International Workshop on Ant Algorithms (ANTS 2002)*, Springer Verlag, Brussels, Belgium.

Eiben, A. E., Hinterding, R., and Michalewicz, Z. (1999): Parameter Control In Evolutionary Algorithms. *IEEE Transactions on Evolutionary Computation* **3**, 24-141.

Eshelman, J., Caruana, R., and Schaffer, J. (1989): Biases In the Crossover Landscape, pp. 10 -19 In H. Schaffer (Ed.): *the third International Conference on Genetic Algorithms*, Morgan Kaufmann, San Mateo, USA.

Eshelman, L. J. (1991): The CHC Adaptive Search Algorithm : to have safe search when engaging in nontraditional genetic recombination, pp. 265-283. In G. J. E. Rawlins (Ed.): *the First Workshop on Foundations of Genetic Algorithms*, Morgan Kaufmann, Bloomington Campus, USA.

Eshelman, L. J. (1997): Genetic Algorithms, pp. B1.2. In T. Back, D. B. Fogel, and Z. Michalewicz (Eds): *Handbook of Evolutionary Computation*, IOP Publising Ltd and Oxford University Press.

Espinoza, F., Minsker, B. S., and Goldberg, D. (2003b): Local Search Issues for the Appliction of a Self-Adaptive Hybrid Genetic Algorithm in Groundwater Remediation Design: *American Society of Civil Engineers (ASCE) Environmental & Water Resources Institute (EWRI) World Water & Environmental Resources Congress 2003 & Related Symposia*, Philadelphia, USA.

Espinoza, F. B., Minsker, B., and Goldberg, D. (2001): A Self Adaptive Hybrid Genetic

Algorithm, pp. 759: *the Genetic and Evolutionary Computation Conference (GECCO 2001)*, Morgan Kaufmann Publishers, San Francisco, USA.

Espinoza, F. B., Minsker, B., and Goldberg, D. (2003a): Performance Evaluation And Population Size Reduction For Self Adaptive Hybrid Genetic Algorithm (SAHGA), pp. 922-933: *the Genetic and Evolutionary Computation Conference*, Springer, San Francisco, USA.

Fogel, D. (1997): Why Evolutionary Computation?, pp. A1.1. In T. Bäck, D. B. Fogel, and Z. Michalewicz (Eds): *Handbook of Evolutionary Computation*, IOP Publising Ltd and Oxford University Press.

Fontanari, J., and Meir, R. (1991): Evolving a Learning Algorithm for the Binary Perceptron. *Network* **2**, 353–359.

Freisleben, B., and Merz, P. (1996): New Genetic Local Search Operators for the Travaling Salesman Problem, pp. 890–899. In H.-M. Voigt, W. Ebeling, I. Rechenberg, and H.-P. Schwefel (Eds): *the Fourth Conference on Parallel Problem Solving from Nature* Springer-Verlag, Berlin, Germany.

Gaertner, D., and Clark, K. L. (2005): On Optimal Parameters for Ant Colony Optimization Algorithms, pp. 83-89: *the 2005 International Conference on Artificial Intelligence, ICAI 2005*, Las Vegas, USA.

Gao, Y. (2003): Population Size and Sampling Complexity in Genetic Algorithms, pp. 178-181. In A. M. Barry (Ed.): *Proceedings of the Bird of a Feather Workshops, Genetic and Evolutionary Computation Conference*, AAAI, Chigaco, USA.

Gendreau, M. (2003): An Introduction to Tabu Search, pp. 37-54. In F. Glover, and G. A. Kochenberger (Eds): *Metaheuristic Handbook*, Kluwer Academic Publishers.

Giraud-Moreau, L., and Lafon, P. (2002): A Comparison of Evolutionary Algorithms for Mechanical Design Components. *Engineering Optimization* **34**, 307-322.

Glover, F. (1989): Tabu Search- part I. *ORSA Journal on Computing* **1**, 190-260.

Glover, F. (1990): Tabu Search: A Tutorial. *Interfaces* **20**, 74-94.

Goldberg, D. E. (1987): Simple Genetic Algorithms and the Minimal, Deceptive Problem, pp. 74-88: *Genetic algorithms and simulated annealing*, Pitman.

Goldberg, D. E. (1989a): *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley.

Goldberg, D. E. (1989b): Sizing Population for Serial and Parallel Genetic Algorithms, pp.

70-79. In J. D. Schaffer (Ed.): *the Third International Conference on Genetic Algorithms*, Morgan Kaufmann, San Mateo, USA.

Goldberg, D. E. (1999): Using Time Efficiently: Genetic-Evolutionary Algorithms and the Continuation Problem, pp. 212-219: *the Genetic and Evolutionary Computation Conference*, Orlando, USA.

Goldberg, D. E. (2003): Foreward. *EURASIP Journal on Applied Signal Processing* **8**, 731-732.

Goldberg, D. E., and Deb, K. (1991): A Comparative Analysis of Selection Scheme Used in Genetic Algorithms, pp. 69-93: *Foundations of Genetic Algorithms*, Morgan Kaufmann, San Mateo, USA.

Goldberg, D. E., Deb, K., and Clark, J. H. (1992): Genetic Algorithms, Noise, and the Sizing of Populations. *Complex Systems* **6**, 333-362.

Goldberg, D. E., and K.Sastry (2001): A Practical Schema Theorem For Genetic Algorithm Design And Tuning, pp. 328-348: *the Genetic and Evolutionary Computation Conference (GECCO 2001)*, Morgan Kaufmann, San Francisco, USA.

Goldberg, D. E., and Lingle, R. (1985): Alleles, Loci, and the Traveling Salesman Problem, pp. 154-159: *the International Conference on Genetic Algorithms and their Applications*, Lawrence Erlbaum, Hillsdale, USA.

Goldberg, D. E., Sastry, K., and Latoza, T. (2001): On the Supply of Building Blocks, pp. 336-342: *the Genetic and Evolutionary Computation Conference (GECCO 2001)*, Morgan Kaufmann, San Francisco, USA.

Goldberg, D. E., and Voessner, S. (1999): Optimizing Global-Local Search Hybrids, pp. 222-228: *the Genetic and Evolutionary Computation Conference (GECCO 1999)*, Morgan Kaufmann, Orlando, USA.

Grefenstette, J. (1992): Genetic Algorithms for Changing Environments, pp. 139-146. In R. Männer, and B. Manderick (Eds): *Parallel Problem Solving from Nature II*, Brussels, Belgium.

Grefenstette, J. J. (1986): Optimization of Control Parameters for Genetic Algorithms. *IEEE Transactions on Systems, Man and Cybernetics* **16**, 122–128.

Grefenstette, J. J. (1987): Incorporating Problem Specific Knowledge into Genetic Algorithm, pp. 42-60: *Genetic Algorithms and Simulated Annealing*, Pitman.

Grefenstette, J. J. (1997): Rank-Based Selection, pp. c2.5:1-4. In T. Back, D. B. Fogel, and Z. Michalewicz (Eds): *Handbook of Evolutionary Computation*, IOP Publishing

and Oxford University Press.

Grefenstette, J. J., Gopal, R., Rosmaita, B., and Gucht, D. v. (1985): Genetic Algorithms for the Traveling Salesman Problem, pp. 160-165. In J. J. Grefenstette (Ed.): *the First International Conference on Genetic Algorithms and Their Applications*, Lawrence Erlbaum, Pittsburgh, USA.

Griewank, A. O. (1981): Generalized Descent for Global Optimization. *Journal of Optimization Theory and Applications* **34**, 11--39.

Gruau, F., and Whitley, D. (1993): Adding Learning To The Cellular Development Of Neural Network: Evolution And Baldwin Effect. *Evolutionary Computation* **1**, 213-233.

Hacker, K. A., Eddy, J., and Lewis, K. E. (2002): Efficient Global Optimization Using Hybrid Genetic Algorithms. 9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, pp. AIAA 2002-5429.

Hagema, J. A., Wehren, R., Sprang, H. A. v., and Buydens, L. M. C. (2003): Hybrid Genetic Algorithm-Tabu Search Approach For Optimizing Multilayer Optical Coating. *Analytica Chimica Acta* **490**, 211-222.

Han, K.-H., and Kim, J.-H. (2002): Quantum-Inspired Evolutionary Algorithm For A Class Of Combinatorial Optimization. *IEEE Transactions On Evolutionary Computation* **6**, 580- 593.

Han, K.-H., and Kim, J.-H. (2004): Quantum-Inspired Evolutionary Algorithm With A New Termination Criterion, He Gate, And Two-Phase Scheme. *IEEE Transactions On Evolutionary Computation* **8**, 156-169.

Harik, G., Cantu-Paz, E., Goldberg, D. E., and Miller, B. l. (1999): The Gambler's Ruin Problem, Genetic Algorithms, and the Sizing of Populations. *Evolutionary Computation* **7**, 231 - 253.

Hart, W. E. (1994): Adaptive Global Optimization With Local Search: *Computer Science & Engineering*, University of California San Diego

Hart, W. E., and Belew, R. K. (1996): Optimization with Genetic Algorithm Hybrids that Use Local Search, pp. 483-496. In R. Belew, and M. Mitchell (Eds): *Adaptive individuals in evolving populations: Models and algorithms*, Addison-Wesley.

Hart, W. E., Kammeyer, T. E., and Belew, R. K. (1995): The Role of Development in Genetic Algorithms. the Third Workshop on Foundations of Genetic Algorithms, pp. 315-332.

Hart, W. E., Rosin, C. R., Belew, R. K., and Morris, G. M. (2000): Improved Evolutionary Hybrids for Flexible Ligand Docking in AutoDock, pp. 209-230. In C. A. Floudas, and P. M. Pardalos (Eds): *Optimization in Computational Chemistry and Molecular Biology*, Springer

Hedar, A., and Fukushima, M. (2003): Simplex Coding Genetic Algorithm For The Global Optimization Of Nonlinear Functions, pp. 135-140. In T. Tanino, T. Tanaka, and M. Inuiguchi (Eds): *Multi-Objective Programming and Goal Programming*, Springer-Verlag.

Herrera, F., and Lozano, M. (1996): Heuristic Crossovers for Real-Coded Genetic Algorithms Based on Fuzzy Connectives, pp. 336 - 345: *the 4th International Conference on Parallel Problem Solving from Nature*, Springer-Verlag Berlin, Germany.

Herrera, F., and Lozano, M. (2001): Adaptive Genetic Operators Based on Coevolution with Fuzzy Behaviors. *IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION* **5**, 149-165.

Hertz, A., Taillard, E., and Werra, D. d. (1995): A Tutorial on Tabu Search, pp. 13-24: *Giornate di Lavoro AIRO'95 (Enterprise Systems: Management of Technological and Organizational Changes)*, Italy.

Hesser, J., and Manner, R. (991): Towards an Optimal Mutation Probability for Genetic Algorithms pp. 23-32. In H. P. S. a. R. M"anner (Ed.): *Parallel Problem Solving from Nature - Proceedings of 1st Workshop, PPSN 1*, Springer-Verlag, Dortmund, Germany.

Hinterding, R. (1997): Self-Adaptation Using Multi-Chromosomes, pp. 87-91: *IEEE International Conference on Evolutionary Computation*, IEEE, Indianapolis, USA.

Hinterding, R., Michalewicz, Z., and Eiben, A. E. (1997): Adaptation in Evolutionary Computation: A Survey. IEEE International Conference on Evolutionary Computation, pp. 65-69.

Hinterding, R., Michalewicz, Z., and Peachey, T. C. (1996): Self-Adaptive Genetic Algorithm for Numeric Functions, pp. 420-429: *the Fourth International Conference on Parallel Problem Solving from Nature*, Springer-Verlag, Berlin, Germany.

Hinton, G., and Nowlan, S. J. (1987): How Learning Can Guide Evolution. *Complex Systems* **1**, 495-502.

Holland, J. (1975): *Adaptation in natural and artificial systems*. The University of Michigan.

Holland, J. H. (1992): Genetic Algorithms. *Scientific American* **267**, 66-72.

Homaifar, A., Guan, S., and Liepins, G. E. (1992): Schema Analysis of the Traveling Salesman Problem using Genetic Algorithms. *Complex Systems* **6**, 533-552

Hopgood, A. A. (2001): *Intelligent Systems for Engineers and Scientists*. CRC Press.

Horst, R., and Pardalos, P. M. (1995): Preface: *Handbook of Global Optimization*, Kluwer Academic.

Houck, C., Joines, J., Kay, M., and Wilson, J. (1997): Empirical Investigation of The Benefits Of Partial Lamarckianism. *Evolutionary Computation* **5**, 31- 60.

Houck, C. R., Joines, J. A., and Kay, M. G. (1996): Comparison of Genetic Algorithms, Random Restart and Two-Opt Switching for Solving Large location-Allocation Problems. *Computers and Operations Research* **23**, 587 - 596.

Ibaraki, T. (1997): Combinations with Other Optimization Methods, pp. D3:1-. In T. Back, D. B. Fogel, and Z. Michalewicz (Eds): *Handbook of Evolutionary Computation*, IOP Publishing and Oxford University Press.

Iorio, A., and Li, X. (2002): Parameter Control within a Co-operative Co-evolutionary Genetic Algorithm, pp. 247-256 *the Seventh International Conference on Parallel Problem Solving from Nature*, Springer-Verlag Granada, Spain.

Ishibuchi, H., Kaige, S., and Narukawa, K. (2005): Comparison Between Lamarckian and Baldwinian Repair on Multiobjective 0/1 Knapsack Problems, pp. 370-385. In Carlos A. Coello Coello, A. H. Aguirre, and E. Zitzler (Eds): *Evolutionary Multi-Criterion Optimization*, Guanajuato, Mexico.

Ishibuchi, H., Yoshida, T., and Murata, T. (2003): Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling. *IEEE Transactions on Evolutionary Computation,* **7**, 204- 223.

Jain, B. J., Pohlheim, H., and Wegener, J. (2001): On Termination Criteria of Evolutionary Algorithms, pp. 768: *the Genetic and Evolutionary Computation Conference (GECC01)*, Morgan Kaufmann Publishers, San Francisco, USA.

Jin, Y. (2005): A Comprehensive Survey Of Fitness Approximation In Evolutionary Computation. *Soft Computing* **9**, 3-12.

Jin, Y., and Sendhoff, B. (2004): Reducing Fitness Evaluations Using Clustering

Techniques And Neural Network Ensembles, pp. 688-699: *Genetic and Evolutionary Computation Conference (GECCO 2004)*, Springer, Seattle, USA.

Joborn, M., Crainic, T. G., Gendreau, M., Holmberg, K., and Lundgren, J. T. (2004): Economies of Scale in Empty Freight Car Distribution in Scheduled Railways. *Transportation Science* **38**, 121–134.

Jog, P., Suh, J. Y., and Gucht, D. V. (1991): Parallel Genetic Algorithms Applied to the Traveling Salesman Problem. *SIAM Journal of Optimization* **1**, 515-529

Joines, J. A., Houck, C. R., and Kay, M. G. (2000a): Characterizing Search Spaces for Tabu Search and Including Adaptive Memory into a Genetic Algorithm. *Journal of the Chinesse Institute of Industrial Engineers* **17**, 527-536.

Joines, J. A., and Kay, M. G. (2002): Hybrid Genetic Algorithms and Random Linkage, pp. 1733-1738: *the 2002 Congress on Evolutionary Computation*, IEEE, Honolulu, USA.

Joines, J. A., Kay, M. G., King, R., and Culbreth, C. (2000b): A Hybrid Genetic Algorithm for Manufacturing Cell Design. *Journal of the Chinese Institute of Industrial Engineers* **17**, 549-564.

Julstrom, B. (1995): What Have You Done for Me Lately? Adapting Operator Probabilities in a Steady-State Genetic Algorithm, pp. 81-87: *the sixth International Conference on Genetic Algorithms*, Pittsburgh, USA.

Julstrom, B. (1999): Comparing Darwinian, Baldwinian, And Lamarckian Search In A Genetic Algorithm For The 4-Cycle Problem, pp. 134-138. In S. Brave, and A. S. Wu (Eds): *the 1999 Genetic and Evolutionary Computation Conference, Late Breaking Papers*, Orlando, USA.

Karr, C. L. (1991): Design of an adaptive Fuzzy Logic Controller Using a Genetic Algorithm, pp. 450-457: *the Fourth International Conference on Genetic Algorithms*, Morgan Kaufmann, San Diego, USA.

Karr, C. L., and Gentry, E. J. (1993): Fuzzy Control of pH using genetic Algorithms. *IEEE Transaction on Fuzzy Systems* **1**, 46-53.

Kennedy, J., and Spears, W. M. (1998): Matching Algorithms to Problems: An Experimental Test of the Particle Swarm and Some Genetic Algorithms on the Multimodal Problem Generator, pp. 78-83. In J. Kennedy (Ed.): *the IEEE International Conference on Evolutionary Computation*, IEEE press, Piscataway, USA.

Kim, Y., and Weck, O. d. (2004): Variable Chromosome Length Genetic Algorithm for Structural Topology Design Optimization. *Structural and Multidisciplinary Optimization* **29**, 445 - 456

Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983): Optimization by Simulated Annealing. *Science* **220**, 671-680.

Konak, A., and Smith, A. E. (1999): A Hybrid Genetic Algorithm Approach for Backbone Design of Communication Networks, pp. 1817-1823: *the 1999 Congress on Evolutionary Computation*, IEEE, Washington D.C, USA.

Koza, J. R., and Rice, J. P. (1991): Genetic Generation of Both the Weights and Architecture for a Neural Network, pp. 397-404: *Joint Conference on Neural Networks*, Seattle, USA.

Krasnogor, N. (1999): Coevolution of Genes and Memes in Memetic Algorithms, pp. 371. In A. Wu (Ed.): *the Genetic and Evolutionary Computation Conference Workshop Program*, Orlando, USA.

Krasnogor, N., and Simth, J. (2001): Emergence Of Profitable Search Strategies Based On A Simple Inheritance Mechanism, pp. 432-439: *the Genetic and Evolutionary Computation Conference*, Morgan Kaufmann, San Francisco, USA.

Krasnogor, N., and Smith, J. (2000): A Memetic Algorithm with Self-Adaptive Local Search: TSP as A Case Study, pp. 987–994: *the Genetic and Evolutionary Computation Conference*, Morgan Kaufmann, Las Vegas, USA

Krasnogor, N., and Smith, J. (2005): A Tutorial for Competent Memetic Algorithms: Model, Taxonomy and Design Issues. *IEEE Transactions on Evolutionary Computation* **9**, 474-488.

Ku, K. W., and Mak, M. W. (1997): Exploring the Effects of Lamarckian and Baldwinian Learning in Evolving Neural Networks, pp. 617-622: *International Conference on Evolutionary Computation*, Indianapolis, USA.

Land, M., SIDorowich, J. J., and Belew, R. K. (1997): Using Genetic Algorithms with Local Search for Thin Film Metrology, pp. 537-544: *the Seventh International Conference on Genetic Algorithms*, Morgan Kaufmann, East Lansing, USA.

Land., M. (1998): Evolutionary Algorithms with Local Search for Combinatorial Optimization: *Computer Science and Engrinerring* University of California

San Diego.

Lawrence, S., Tsoi, A. C., and Back, A. D. (1996): Function Approximation with Neural

Networks and Local Methods: Bias, Variance and Smoothness, pp. 16–21: *Australian Conference on Neural Networks,*, Canberra.

Leng, L. T. (1999): Guided Genetic Algorithm: *Department of Computer Science,*, University of Essex.

Levine, D. (1994): A Parallel Genetic Algorithm for the Set Partitioning Problem: *Argonne National Laboratory.*

Liang, H., Lin, Z., and McCallum, R. W. (2000): Application of Combined Genetic Algorithms with Cascade Correlation to Diagnosis of Delayed Gastric Emptying from Electrogastrograms. *Medical Engineering & Physics* **22**, 229–234.

Liang, K., Yao, X., and Newton, C. (1999): Combining Landscape Approximation and Local Search In Global Optimization, pp. 1514-1520: *the Congress on Evolutionary Computation*, IEEE Press, Washington DC, USA.

Lin, B., and Miller, D. C. (2004a): Tabu Search Algorithm for Chemical Process Optimization. *Computers & Chemical Engineering* **28**, 2287-2306.

Lin, B., and Miller, D. C. (2004b): Solving Heat Exchanger Network Synthesis Problems with Tabu Search. *Computers & Chemical Engineering* **28**, 1451-1464.

Lin, F. T., Kao, C. Y., and Hsu., C. C. (1991): Incorporating genetic algorithms into simulated annealing, pp. 290-297: *the Fourth International Symposium on Artificial Intelligence*, Cancun, Mexico.

Lobo, F. G., D, Goldberg, E., and Pelikan, M. (2000): Time complexity of genetic algorithms on exponentially scaled problems, pp. 151–158: *The genetic nd evolutionary computation conference*, Morgan-Kaufmann, Las Vegas, USA

Lobo, F. G., and Goldberg, D. E. (1997): Decision Making in a Hybrid Genetic Algorithm, pp. 122-125: *IEEE International Conference on evolutionary Computation*, IEEE Press, Piscataway, USA.

Lozano, M., Herrera, F., Krasnogor, N., and Molina, D. (2004): Real-Coded Memetic Algorithms With Crossover Hill-Climbing. *Evolutionary computation* **12**, 273 - 302

Magyar, G., Johnsson, M., and Nevalainen, O. (2000): An adaptive Hybrid Genetic Algorithm for the Three-Matching Problem. *IEEE Transaction on Evolutionary Computation* **4**, 135-146.

Mahfoud, S., and Goldberg, D. (1995): Parallel Recombinative Simulated Annealing: A genetic algorithm. *Parallel Computing* **21**, 11-28.

Mahfoud, S. W. (1997): Bltzmann Selection, pp. C2.5:1-4. In T. Back, D. B. Fogel, and Z. Michalewicz (Eds): *Handbook of Evolutionary Computation*, IOP Publising Ltd and Oxford University Press.

Maniezzo, V., Gambardella, L. M., and F. de Luigi. (2004): Ant colony optimization, pp. 101-117. In G. C. Onwubolu, and B. V. Babu (Eds): *New Optimization Techniques in Engineering*, Springer-Verlag.

Martinez-Estudillo, A., Hervas-Martnez, C., Martnez-Estudillo, F., and Garca-Pedrajas, N. (2004): Hybrid Method Based on Clustering for Evolutionary Algorithms with Local Search. *IEEE Transactions on Systems, Man and Cybernetics.*

Mathias, K., and Whitley, D. (1992): Genetic Operators, the Fitness Landscape and the Traveling Salesman Problem, pp. 219-228: *Parallel Problem Solving from Nature-PPSN 2*, North Holland-Elsevier, Brussels, Belguim.

Mathias, K., Whitley, L., Stock, C., and Kusuma, T. (1994): Staged Hybrid Genetic Search For Seismic Data Imaging, pp. 356-361: *International Conference on Evolutionary Computation*, Orlando, USA.

Mayley, G. (1996): Landscapes, Learning Costs and Genetic Assimilation. *Evolutionary Computation* **4**, 213 - 234.

De la Maza, M., and Yure, D. (1995): Seeing Clearly: Medical Imaging Now and Tomorrow: *Future Health: Computers and Medicine in the 21st Century*, St. Martin's Press.

Michalewicz, Z. (1996): *Genetic Algorithms + Data Structures = Evolution Programs* Springer-Verlag.

Michalewicz, Z., and Fogel, D. B. (2000): *How to Solve It: Modern Heuristics.* Springer-Verlag.

Michalewicz, Z., Hinterding, R., and Michalewicz, M. (1997): Evolutionary Algorithms, pp. chapter 2. In W. Pedrycz (Ed.): *Fuzzy Evolutionary Computation*, Kluwer Academic.

Michalewicz, Z., and Nazhiyath, G. (1995): Genocop III: A Co-Evolutionary Algorithm For Numerical Optimization Problems With Nonlinear Constraints, pp. 647-651: *2nd IEEE International Conference on Evolutionary Computation*, IEEE, Perth, Australia

Michel, R., and Middendorf, M. (1998): An Island Model Based Ant System with Lookahead for the Shortest Supersequence Problem, pp. 692–701: *the Fifth*

International Conference on Parallel Problem Solving from Nature, Springer-Verlag, Amsterdam, The Netherlands.

Miller, G. F., Todd, P. M., and Hegde, S. U. (1989): Designing neural networks using genetic algorithms, pp. 379-384. In J. D. Schaffer (Ed.): *the Third International Conference on Genetic Algorithms*, Morgan Kaufmann, San Mateo, USA.

Miller, J. A., Potter, W. D., Gandham, R. V., and Lapena, C. N. (1993): An Evaluation Of Local Improvement Operators For Genetic Algorithms. *IEEE Transactions of Systems, Man and Cybernetics,* **23**, 1340-1351.

Mitchell, M., Holland, J. H., and Forrest, S. (1993): When Will A Genetic Algorithm Outperform Hill Climbing? *Advances in Neural Information Processing Systems* **6**, 51-58.

Monmarch'e, N., Venturini, G., and Slimane, M. (2000): On How Pachycondyla Apicalis Ants Suggest a New Search Algorithm. *Future Generation Computer Systems* **16**, 937-946.

Montana, D. J. (1995): Neural Network Weight Selection Using Genetic Algorithms, pp. 85-104: *Intelligent Hybrid Systems*, John Wiley & Sons.

Morris, G. M., Goodsell, D. S., Halliday, R. S., Huey, R., Hart, W. E., Belew, R. K., and Olson, A. J. (1998): Automated Docking Using a Lamarckian Genetic Algorithm and an Empirical Binding Free Energy Function. *Journal of Computational Chemistry* **19**, 1639-1662.

Moscato, P. (1989): On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards Memetic Algorithms: *Tech. Rep. Caltech Concurrent Computation Program, Report. 826*, California Institute of Technology.

Mühlenbein, H., and Schlierkamp-Voosen, D. (1993): Predictive Models for the Breeder Genetic Algorithm. *Evolutionary Computation* **1**, 25-49.

Mühlenbein, H., Schomisch, M., and Born, J. (1991): The Parallel Genetic Algorithm as Function Optimizer. *Parallel Computing* **17**, 619-632.

Nolle, L., Armstrong, A., and Lee, S. (2000): On a Class of Non-Linear Rank Based Genetic Algorithms, pp. 101-106: *5th International Conference on Soft Computing: MENDEL 2000*, Brno, Czech Republic.

Ochoa, G., Harvey, I., and Buxton, H. (1999): On Recombination and Optimal Mutation Rates, pp. 13-17: *Genetic and Evolutionary Computation Conference*, Orlando, USA.

Oliver, I. M., Smith, D. J., and Holland, J. R. C. (1987): A Study of Permutation Crossover Operators on the Traveling Salesman Problem, pp. 224 - 230: *the Second International Conference on Genetic Algorithms on Genetic algorithms and their application*, Hillsdale, USA.

Ong, Y.-S., and Keane, A. J. (2004): Meta-Lamarckian learning in memetic algorithms. *IEEE Transactions on Evolutionary Computation* **8**, 99-110.

Orvosh, D., and Davis, L. (1993): Shall We Repair? Genetic Algorithms, Combinatorial Optimization, and Feasibility Constraints, pp. 650: *the Fifth International Conference on Genetic Algorithms*, Morgan Kaufmann, Urbana-Champaign, USA.

Pearce, R., and Cowley, P. H. (1996): Use of Fuzzy Logic to Describe Constraints Derived from Engineering Judgment in Genetic Algorithms. *IEEE Transactions on Industrial Electronics* **43**, 535-540.

Pelikan, M., Goldberg, D. E., and Cantu-Paz, E. (1999a): BOA: The Bayesian Optimization Algorithm, pp. 525-532: *the Genetic and Evolutionary Computation Conference*, Morgan Kaufmann, Orlando, USA.

Pelikan, M., Goldberg, D. E., and Lobo, F. (1999b): A Survey of Optimization by Building and Using Probabilistic Models: *Technical Report 99018*, IlliGAL.

Pilat, M. L., and White, T. (2002): Using Genetic Algorithms to Optimize ACS-TSP, pp. 282 - 287: *the Third International Workshop on Ant Algorithms*, Springer-Verlag, Berlin, Germany.

Potter, M., and De Jong, K. (1994): A Cooperative Coevolutionary Approach to Function Optimization, pp. 249-257: *Third Parallel Problem Solving From nature*, Springer-Verlag, Jerusalem.

Press, W., Teukolsky, S., Vetterling, W., and Flannery, B. (1993): *Numerical Recipes In C.* Cambridge University Press.

Preux, P., and Talbi, E.-G. (1999): Towards Hybrid Evolutionary Algorithms. *International Transactions in Operational Research* **6**, 557-570.

Radcliffe, N. J., and Surry, P. D. (1994): Formal Memetic Algorithms, pp. 1-16: *Evolutionary Computing: AISB Workshop*, Springer-Verlag, Brighton, UK.

Rana, S. (1999): The Distributional Biases Of Crossover Operators, pp. 549-556: *the Genetic and Evolutionary Computation Conference*, Morgan Kaufmann, Orlando, USA.

Reed, P., Minsker, B. S., and Goldberg, D. E. (2000): Designing A Competent Simple

Genetic Algorithm For Search And Optimization. *Water Resources Research* **36**, 3731-3741.

Reeves, C. (1993): Using Genetic Algorithms With Small Populations, pp. 92-99: *the Fifth International Conference on Genetic Algorithms*, Morgan Kaufman, Urbana-Champaign, USA.

Reeves, C. (1994): Genetic Algorithms and Neighbourhood Search, pp. 115-130. In T. C. Fogarty (Ed.): *Evolutionary Computing, AISB Workshop*, Springer-Verlag, Leeds, UK.

Richter, J. N., and Peak, D. (2002): Fuzzy Evolutionary Cellular Automata, pp. 185-191: *International Conference on Artificial Neural Networks In Engineering*, Saint Louis, USA.

Riopka, T. P., and Bock, P. (2000): Intelligent Recombination Using Individual Learning in a Collective Learning Genetic Algorithm, pp. 104-111. In D. Whitley, D. Goldberg, E. Cantu-Paz, L. Spector, I. Parmee, and H.-G. Beyer (Eds): *the Genetic and Evolutionary Computation Conference (GECCO-2000)*, Morgan Kaufmann, Las Vegas, USA.

Rogers, A., and Prügel-Bennett, A. (1999): Genetic Drift in Genetic Algorithm Selection Schemes. *IEEE Transactions on Evolutionary Computation* **3**, 298-303.

Rolland, E. (1997): A Tabu Search Method for Constrained Real Number Search: Applications to Portfolio Selection: *Technical Report*, Dept. of Accounting and Management Information Systems. Ohio State University, Columbus. U.S.A.

Rosin, C. D., Halliday, R. S., Hart, W. E., and Belew, R. K. (1997): A Comparison of Global and Local Search Methods in Drug Docking, pp. 221-228. In T. Bäck (Ed.): *the Seventh International Conference on Genetic Algorithms*, Morgan Kaufmann, Michigan, USA.

Rothlauf, F. (2002): Binary Representations of Integers and the Performance Of Selectorecombinative Genetic Algorithms, pp. 99 - 110 In J. J. Merelo, P. Adamidis, and H.-G. Beyer (Eds): *the seventh International Conference on Parallel Problem Solving from Nature*, Springer-Verlag, Granada, Spain.

Rudolph, G. (1994): An Evolutionary Algorithm for Integer Programming, pp. 139-148. In Y. Davidor (Ed.): *Parallel Problem Solving from Nature III*, Jerusalem.

Sasaki, T., and Tokoro, M. (1997): Adaptation toward Changing Environments: Why Darwinian in Nature?, pp. 145-153. In P. Husbands, and I. Harvey (Eds): *Fourth*

European Conference on Artificial Life, MIT press, Brighton, UK.

Sauter, J. A., Matthews, R. S., Parunak, H. V. D., and :, S. B. (2002): Evolving adaptive pheromone path planning mechanisms, pp. 434-440. In C. Castelfranchi, and W. Johnson (Eds): *First International Joint Conference on Autonomous Agents and Multi-Agent Systems*, ACM Press, Bologna, Italy.

Schaffer, J. D., and Morishima, A. (1987): An Adaptive Crossover Distribution Mechanism for Genetic Algorithms, pp. 36-40. In J. J. Grefenstette (Ed.): *the Second International Conference on Genetic Algorithms*, Lawrence Erlbaum, Hillsdale, USA.

Schraudolph, N. N., and Belew, R. K. (1992): Dynamic Parameter Encoding For Genetic Algorithms. *Machine Learning Journal* **9**, 9-22.

Schwefel, H.-P. (1995): *Evolution and Optimum Seeking*. Wiley. New York.

Schwefel, H.-P. (1997): Advantages (and Disadvantages) of Evolutionary Computation Over Other Approaches, pp. A1.3. In T. Bäck, D. B. Fogel, and Z. Michalewicz (Eds): *Handbook of Evolutionary Computation*, IOP Publishing and Oxford University Press.

Semet, Y., Lutton, E., and Collet., P. (2003): Ant Colony Optimisation for E-Learning: Observing the Emergence of Pedagogic Suggestions, pp. 46- 52: *IEEE Swarm Intelligence Symposium*, IEEE press, Indianapolis, USA

Shawe-Taylor, J., and Zerovnik, J. v. (2001): Ants and Graph Coloring, pp. 276-279. In V. Kurkova, N. C. Steele, R. Neruda, and M. Karny (Eds): *the International Conference on Artificial Neural Nets and Genetic Algorithms*, Springer-Verlag, Portoroz, Slovenia.

Shmygelska, A., and Hoos, H. H. (2005): An Ant Colony Optimisation Algorithm for the 2D and 3D Hydrophobic Polar Protein Folding Problem. *BMC Bioinformatics* **6**, 6-30.

Sinha, A., and Goldberg, D. E. (2001): Verification And Extension Of The Theory Of Global-Local Hybrids, pp. 592-598: *the Genetic and Evolutionary Computation Conference*, Morgan Kaufmann, Sanfrancisco, USA.

Smith, J. E., and Fogarty, T. C. (1996): Self Adaptation of Mutation Rates in a Steady State Genetic Algorithm, pp. 318 - 323: *IEEE International Conference on Evolutionary Computing*, IEEE press, Nagoya, Japan.

Smith, J. E., and Fogarty, T. C. (1997): Operator and Parameter Adaptation in Genetic

Algorithms. *Soft Computing*, 81-87.

Socha, K. (2004): ACO for Continuous and Mixed-Variable Optimization, pp. 25-36: *the Fourth International Workshop on Ant Colony Optimization and Swarm Intelligence (ANTS 2004)*, Brussels, Belgium.

Soils, F., and Wets, R. (1981): Minimization by Random Search Techniques. *Mathematical Operations Research* **30**, 19-30.

Spears, W. M. (1992): Crossover or Mutation?, pp. 221-237: *Foundations of Genetic Algorithms Workshop* Vail, USA.

Spears, W. M. (1995): Adapting Crossover in a Genetic Algorithm, pp. 367-384: *the Fifth Conference on Evolutionary Programming*, MIT press, San Diego, USA.

Spitzer, F. (2000): *Principles of random walk*. Springer.

Striz, and Sobieszczanski-Sobieski (1996): Displacement Based Multilevel Structural Optimization: *AIAA-CP-4098*, American Institute of Aeronautics and Astronautics, Inc.

Sung-Soon, C., and Byung-Ro, M. A Graph-Based Lamarckian-Baldwinian Hybrid for the Sorting Network Problem *IEEE Transactions on Evolutionary Computation* **9**, 105-114.

Syrjakow, M., and Szczerbicka, H. (1995): Combination of Direct Global and Local Optimization Methods, pp. 326-333: *IEEE Conference on Evolutionary Computation*, IEEE, Perth, Western Australia.

Syswerda, G. (1989): Uniform Crossover in Genetic Algorithms, pp. 2 - 9 *the third international conference on Genetic algorithms*, Morgan Kaufmann, Fairfax, USA.

Talbi, E. (2002): A Taxonomy of Hybrid Metaheuristics. *Journal of Heuristics* **8**, 541–564.

Talbi, H., Draa, A., and Batouche, M. (2004): A New Quantum-Inspired Genetic Algorithm for Solving the travelling Salesman Problem: *14th International Conference on Computer Theory and Applications*, Alexandria, Egypt

Tan, K. C., Li, Y., Murray-Smith, D. J., and Sharman, K. C. (1995): System Identification and Linearisation using Genetic Algorithms with Simulated Annealing. First IEE/IEEE Int. Conf. on GA in Eng. Syst.: Innovations and Appl., pp. 164-69.

Thierens, D. (1997): Selection Schemes, Elitist Recombination, and Selection Intensity, pp. 152-159: *International Conference on Genetic Algorithms*, Morgan kaufmann, East Lansing, USA.

Thierens, D., Goldberg, D., and Guimaraes, P. (1998): Domino Convergence, Drift, and The Temporal-Salience Structure Of Problems, pp. 535-540: *1998 IEEE International Conference on Evolutionary Computation* IEEE, Anchorage, USA.

Thierens, D., and Goldberg, D. E. (1994): Convergence Models of Genetic Algorithm Selection Schemes, pp. 119–129: *the parallel problem solving from nature III*, Springer-Verlag, Jerusalem.

Törn, A., and Zilinskas, A. (1989): Global Optimization, pp. 350: *Lecture Notes in Computer Science*, Springer-Verlag.

Toussaint, M., and Igel, C. (2002): Neutrality: A Necessity For Self-Adaptation, pp. 1354-1359.: *the IEEE Congress on Evolutionary Computation (CEC 2002)*, Morgan Kaufmann, Honolulu, USA.

Tsang, E. P., and Voudouris, C. (1997): Fast Local Search and Guided Local Search and their application to British Telecom's Workforce Scheduling Problem. *In Operations Research Letters* **20**, 119-127.

Tsutsui, S. (2004): Ant Colony Optimisation for Continuous Domains with Aggregation Pheromones Metaphor, pp. 207-212: *the 5th International Conference on Recent Advances in Soft Computing (RASC-04)*, Nottingham, UK.

Tsutsui, S., Pelikan, M., and Ghosh, A. (2005): Performance of Aggregation Phermone System on Unimodal and Multimodal Problems, pp. 880-887: *The 2005 IEEE Congress on Evolutionary Computation*, IEEE, Edinburgh, UK.

Turney, P. (1996): Myths And Legends Of The Baldwin Effect, pp. 135-142: *Proceedings Workshop on Evolutionary Computation and Machine Learning at the 13th International Conference on Machine Learning*, Bari, Italy.

Turney, P. (1999): Increasing Evolvability Considered As A Large-Scale Trend In Evolution, pp. 43-46: *the 1999 Genetic and Evolutionary Computation Conference (GECCO-99),Workshop on Evolvability* Morgan Kaufmann, Orlando, Florida, USA

Turney, P., Whitley, D., and Anderson, R. (1996): Evolution, Learning, And Instinct: 100 Years Of The Baldwin Effect. *Evolutionary Computation* **4**, iv-viii.

Tuson, A., and Ross, P. (1998): Adapting Operator Settings in Genetic Algorithms. *Evolutionary Computation* **6**, 161-184

Tuson, A. L., and Ross, P. (1996): Cost Based Operator Rate Adaptation: An Investigation, pp. 461-469: *the Fourth International Conference on Parallel Problem Solving*

From Nature (PPSN IV), , Springer Verlag, Berlin, Germany.

Valenzuela-Rendon, M. (1991): The Fuzzy Classifier System: Motivations and First tresults, pp. 338-342: *the International Workshop Parallel Problem*, Springer, Dortmund, Germany.

Vekaria, K., and Clack, C. (1999): Biases Introduced by Adaptive Recombination Operators, pp. 670-677: *the Genetic and Evolutionary Computation Conference*, Morgan Kaufmann, Orlando, USA.

Voudouris, C. (1998): Guided Local Search - An illustrative example in function optimisation. *BT Technology Journal* **16**, 46-50.

Voudouris, C., and Tsang, E. (1995): Function Optimization Using Guided Local Search: *Technical Report CSM-249*, Department of computer science, University of Essex.

Wang, Q., Spronck, P., and Tracht, R. (2003): An Overview of Genetic Algorithms Applied to Control Engineering Problems, pp. 1651 - 1656: *International Conference on Machine Learning and Cybernetics*, IEEE, Xi-an, China.

White, T., Pagurek, B., and Oppacher, F. (1998): ASGA: Improving the Ant System by Integration with Genetic Algorithms, pp. 610-617: *the third Conference on Genetic Programming (GP/SGA'98)*, Madison, USA.

Whitley, D. (1989): The GENITOR Algorithm and Selection Pressure: Why Rank-Based Allocation of Reproductive Trials is Best, pp. 116-121: *the Third International Conference on Genetic Algorithms*, Morgan Kaufmann, Fairfax, USA.

Whitley, D., Das, R., and Crabb, C. (1992): Tracking Primary Hyperplane Competitors During Genetic Search. *Annals of Mathematics and Artificial Intelligence* **12**, 367-388.

Whitley, D., Gordon, S., and Mathias, K. (1994): Lamarckian Evolution, The Baldwin Effect And Function Optimization, pp. 6-15. In Y. Davidor, H.-P. Schwefel, and R. Manner (Eds): *Parallel Problem Solving from Nature - PPSN III* Springer-Verlag, Jerusalem.

Whitley, D., Mathias, K., Rana, S., and Dzubera, J. (1995): Building Better Test Functions, pp. 239-246. In Eshelman (Ed.): *International Conference on Genetic Algorithms* Morgan Kaufmann, Pittsburgh, USA.

Whitley, D., Starkweather, T., and Fuquay, D. A. (1989): Scheduling Problems and Traveling Salesman: The Genetic Edge Recombination Operator. the Third International Conference on Genetic Algorithms, pp. 133 - 140.

Willmes, L., Back, T., Jin, Y., and Sendhoff., B. (2003): Comparing Neural Networks and Kriging for Fitness Approximation in Evolutionary Optimization, pp. 663-670: *IEEE Congress on Evolutionary Computation*, Canberra, Australia.

Wroblewski, J. (1996): Theoretical Foundations of Order-Based Genetic Algorithms. *Fundamenta Informaticae.*, 423–430.

Yamada, T., and Nakano, R. (1995): A Genetic Algorithm with Multi-Step Crossover for Job-Shop Scheduling Problems, pp. 146–151: *First IEE/IEEE International Conference on Genetic ALgorithms in Engineering Systems Innovations and Applications (GALESIA '95)*, Sheffield, UK.

Yamada, T., and Reeves, C. (1998): Solving the Csum Permutation Flowshop Scheduling Problem by Genetic Local Search, pp. 230-234: *International Confrenence on Evolutionary Computation*, Anchorage, USA.

Yao, X. (1999): Evolving artificial neural networks. *Proceedings of the IEEE* **87**, 1423-1447.

Yen, J., Liao, J. C., Lee, B., and Randolph, D. (1998): A Hybrid Approach To Modeling Metabolic Systems Using Genetic Algorithms And Simplex Method. *IEEE Transactions on Systems, Man, and Cybernetics* **28**, 173-191.

Yuret, D. (1994): From Genetic Algorithms To Efficient Optimization, Technical Report 1569, MIT AI Laboratory.

Yuret, D., and De la Maza, M. (1993): Dynamic Hill Climbing: Overcoming The Limitations Of Optimization Techniques, pp. 254-260: *The Second Turkish Symposium on Artificial Intelligence and Neural Networks*, Istanbul, Turkey.

Zhang, B.-T., and Kim, J.-J. (2000): Comparison of Selection Methods for Evolutionary Optimization. *Evolutionary Optimization* **2**, 55-70.

Performance of Hybrid Genetic Algorithms Incorporating Local Search

T. Elmihoub, A. A. Hopgood, L. Nolle and A. Battersby
The Nottingham Trent University, School of Computing and Technology,
Burton Street, Nottingham NG1 4BU, United Kingdom
{tarek.elmihoub, adrian.hopgood, lars.nolle, alan.battersby}@ntu.ac.uk

ABSTRACT

This paper investigates the effects of learning strategy and probability of local search on the performance of hybrid genetic algorithms. It compares the performance of two genetic–local hybrids using different learning strategies and different probabilities of local search. Two test functions are used for the comparisons. The results show that the solution quality of hybrids is not only affected by the Lamarckian or Baldwinian learning strategy, but also by the probability of local search. This probability, together with the learning strategy, has a great impact on population size requirements. These requirements are also affected by the local search method, and the fitness landscape. Reducing the population size can lead to an increase in the algorithm convergence speed.

INTRODUCTION

The ability of genetic algorithms to capture a global view of the search space, when combined carefully with the fast convergence of local search methods (Turney 1996), can often produce an algorithm that outperforms either one alone (Bobo and Goldberg 1997). Hybridizing a local search method provides the global genetic search algorithm with some local knowledge that can guide and may accelerate the search to the global optimum (Hart 1994).

The usual motivation for hybridization in optimization practice is the achievement of increased efficiency (Goldberg and Vosser 1999). The efficiency of any hybrid depends on many factors, e.g. how the hybrid decides between global and local knowledge (Bobo and Goldberg 1997), how it strikes a balance between the cost and value of local knowledge (Hart 1994), and how successfully local knowledge are utilized by the global genetic algorithm (Whitley et al. 1994). The efficiency of any hybrid can be measured by comparing its performance with that of the global genetic algorithm alone. Espinoza et al. (2001) have proposed an adaptive hybrid algorithm that can increase convergence speed to the global optimum. The same authors also show the effect of a local search method on reducing the population size of the algorithm compared with the population size of the standard genetic algorithm (Espinoza et al. 2003).

In this paper, a further step is taken in this direction by investigating the effect of the learning strategy and probability of local search on the performance on both the adaptive hybrid algorithm and the standard staged hybrid. The effect of both these factors on the population size requirements, convergence speed and solution quality has been studied.

LAMARCKIAN EVOLUTION AND BALDWIN EFFECT

One of the important issues of hybrid genetic algorithms is how the information gained during local search is used by the global algorithm. Either the Lamarckian or the Baldwinian approach can be used. In the Lamarckian approach the traits acquired during the learning process are passed from parents to their offspring. This means that both the genetic structure of an individual and its associated fitness value are modified to reflect the changes in phenotype structure as a result of performing local search (Whitley et al. 1994). The Baldwin Effect is somewhat Lamarckian in its results but using different mechanisms (Turney 1996) In the Baldwinian approach the learning process can help the individual to adapt to its environment and as a result to survive and gain more chance to pass on its traits to the next generation. In this case, only the improved fitness value is modified to reflect the effect of performing local search, thereby allowing individuals with the ability to learn to proliferate in the population.

Although Lamarckian evolution has been universally rejected as a viable theory of genetic evolution in nature, using it as learning strategy in genetic algorithms can improve their convergence speed (Whitley et al. 1994). The Lamarckian strategy can disrupt schema processing of genetic algorithms in staged hybrid algorithms and in some cases this may lead to the premature convergence problem (Whitley et al. 1994). In many real-world applications, it is not possible to use the Lamarckian approach because the inverse mapping from phenotype to genotype is computationally intractable (Turney 1996). The Baldwinian approach, in spite of being characterized by slow convergence speed compared with that of Lamarckian (Whitley et al. 1994), has a smoothing effect on the search landscape and does not disrupt the global genetic search (Gruau and Whitley 1993).

Utilizing either form of learning is more effective than the standard genetic algorithm approach without a local

Proceedings 18th European Simulation Multiconference
Graham Horton (c) SCS Europe, 2004
ISBN 3-936150-35-4 (book) / ISBN 3-936150-36-2 (CD)

improvement procedure (Whitley et al. 1994). The effectiveness of pure Lamarckian, pure Baldwinian or any mixture of them is affected by the fitness landscape, the representations, and local search method used (Whitley et al. 1994; Houck et al. 1997;Michalewicz and Nazhiyath 1995).

BALANCE BETWEEN COST AND VALUE OF LOCAL KNOWLEDGE

In any hybrid algorithm, a local search can be applied either to every individual in the population or only few individuals. Applying a local search to every individual in the population can waste resources without providing any more useful information than applying it to only a small fraction of the population. The use of a large fraction of the population can limit exploring the search space by allowing the genetic algorithm to evolve for a small number of generations. A more selective use of local search can improve the efficiency of hybrids (Hart 1994). Deciding on the optimal fraction of the population that should perform local search, and the basis on which these individuals are chosen, has been investigated by Hart (1994). The cost of local knowledge is measured by the number of function evaluations performed by a local search method to gain that knowledge and its value is measured by its effect on increasing the convergence speed and/or solution quality of the algorithm. The probability of local search can affect the minimum population size of the hybrid which, in turn, can affect the convergence speed of the algorithm. This effect should not be ignored when deciding between different local search probabilities.

POPULATION SIZE REQUIREMENTS

Efficient Population sizing is critical in genetic algorithms for getting the most out of a fixed budget of function evaluations. In (Harik et al. 1997) two factors that influence convergence quality are considered to estimate the population size of genetic algorithms. These factors are the initial supply of building blocks and the selection of the best building blocks over their competitors. The gambler ruin model is used to derive the following relation for population size of genetic algorithms

$$N = \frac{-2^{k-1}\ln(\alpha)\sigma_{bb}\sqrt{\pi(m-1)}}{d}$$

where k is the building-block order, which represents the minimum number of binary digits that have physical significance to the solution of the problem. α is the probability of failure, σ_{bb} is the standard deviation of the building blocks, d is the signal difference between the best and second-best building blocks, and m is the maximum number of building-blocks within a single string. The term $\sqrt{\pi(m-1)}$ represents the noise interference between competing building-blocks. This term can be approximated using the fitness function standard deviation, $\sigma_{fitness}$ (Reed et al 2000).

The computational complexity of a genetic algorithm is measured as the number of function evaluations that are required to attain an optimal solution. The number of function evaluations can be calculated by multiplying the population size (N) by the number of generations required for convergence (t). The number of generation required is strongly affected by the relative rates at which genes within the population converge. The lower and upper bounds for the convergence rates for genetic algorithms applications are functions of $O(\sqrt{l})$ and $O(l)$ for tournament selections, where l is the string length (Thierens et al. 1998). The building blocks of most engineering problems converge at variable rates within the population (Reed 2000). This phenomenon is known as domino convergence. The expected number of generations (t) required under domino convergence for all locations to be converged is given by the following equation.

$$t_{domino} \approx 2l$$

Another phenomenon that is closely related to domino convergence is "genetic drift" (Thierens et al. 1998). This phenomenon occurs in a population when crossover and mutation cause genes to fluctuate and converge to non-optimal values in the absence of selection pressure. Although the genes with reduced relevance to the solution experience reduced selection pressure, they may converge to non-optimal values under the crossover and mutation operations. The expected number of generations for genes to converge in the absence of selection within a randomly generated initial population is given by the following equation (Thierens et al. 1998):

$$t_{drift} \approx 1.4N$$

Domino convergence to optimal solution should occur before genetic drift can occur. The following inequality needs to be satisfied:

$$t_{domino} < t_{drift}$$

or, in terms of population size and string length,

$$N > 1.43l$$

Since carefully designed hybrid genetic algorithms often converge faster than standard genetic algorithms, their convergence to the global optimum can occur even if population size is not greater than $1.43l$. The population size for hybrid algorithm should satisfy the following relation

$$2l\frac{t_{dominо}}{t_{drift}} < 1.43N$$

where t_{hybrid} is the number of generations required to for a hybrid to converge.

The local search method affects the signal difference between the best individual and the second best, and this can either increase or decrease the population size. It can also decrease the standard deviation of the population and this leads to a decrease in the population size.

ALGORITHMS AND TEST FUNCTIONS

Two hybrids with different mechanisms for deciding between global and local search were used to gain some insight into the effect of learning strategy and probability of local search on the performance of hybrids. The standard staged hybrid genetic algorithm (SSH) (Mathias and Whitley 1994) and the adaptive staged hybrid genetic algorithm (ASH) (Espinoza et al. 2001) have been tested using two multimodal test functions.

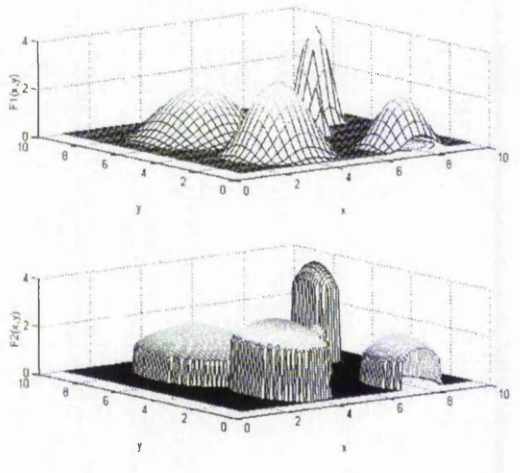

Figure 1: Fitness Landscapes for the Test Functions

In the standard staged hybrid genetic algorithm (SSH), the local search step is defined by three basic parameters: frequency of local search, probability of local search and number of local iterations. The local search frequency measures how frequently local search is performed; the probability of the local search represents the fraction of individuals in the population that undergo local search at each local search iteration; and the number of local search iterations represents the number of local search iterations performed at each local search process.

The adaptive staged hybrid genetic algorithm (ASH), uses feedback from the current state of the search process to direct the algorithm to decide between global and local methods (Espinoza et al. 2001). The algorithm works with the same operators as SSH. It performs local search only if new regions of search space are being discovered, and local knowledge can help to guide the search. The probability of the local search is controlled by a deterministic rule that keeps this probability less than a specific value. When local search no longer improves the average fitness more than the most recent global search iteration, the search goes back to the global search.

Two multimodal test functions, with multiple basins of attraction, have been used in the current work. The first function, F1, has conical basins of attraction. Its global maximum is 4 and is located at (7.0,8.5) (Goldberg and Vosser 1999; Espinoza et al. 2001). The second function, F2, has elliptical basins of attraction. This function has a global optimum of 4 located at (7.0, 8.5) (Espinoza et al. 2001). Figure 1 shows the fitness landscapes of F1 and F2.

The steepest descent method (Press et al.1993) was used as a local searcher. The steepest descent algorithm uses the derivatives of the function to estimate the best step size to climb to the local optimum from the current position in the basin of attraction.

SIMULATIONS AND DISCUSSION

In order to evaluate the effect of learning strategy and local search probability on the hybrids' performance, a set of experiments was performed. Both hybrids use the simple elitist genetic algorithm with binary tournament selection, single-point crossover, and simple mutation. For all experiments, the probability of local search was 0.4 and the probability of mutation was $1/N$ where N is the population size (Reed et al. 2000). For SSH, the frequency of local search was 3 and the number of local iterations was 3. For ASH, the maximum number of local iterations was 3, e was 0.2, and the local threshold value was 0.6. Each variable was represented by 30-bit string with a total of 60 bits for each chromosome. The stopping criterion for all experiments was that 80% of the population had converged to the solution.

Effects on Convergence Speed

In the experiment to evaluate the effect of learning strategy on convergence speed of hybrid algorithms, both the adaptive and standard staged algorithms used a probability of local search of 0.1, and population sizes of 800 and 1200 for F1 and F2 (Espinoza 2001).The stopping criterion was that 80% of the population converged within 0.000001 boundaries of the best ever found solution.

The results show, as expected, that increasing the fraction of the population that evolves according to the Lamarckian approach leads to an increase in the convergence speed. This increase is not linear. For

example, when applying ASH on F2, the speed of convergence increases sharply as the learning approach changes from pure Baldwinian (100% Baldwinian) to a mixture of 80% Baldwinian and 20% Lamarckian. In this interval the number of function evaluations decreases from 85,000 to about 37,000, while it decreases to 25,000 evaluations for the pure Lamarckian approach. Figure 2 shows the effect of learning strategy on the convergence speed of the adaptive staged hybrid. The effect of learning strategy on the convergence speed of standard staged hybrid and the adaptive staged hybrid are similar for both test functions.

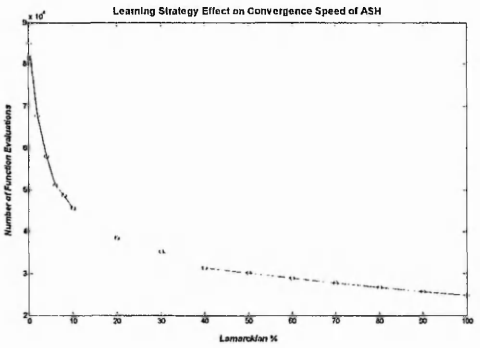

Figure 2: Effect of Learning Strategy on Convergence Speed

Effects on Solution Quality

The results of previous experiments show no clear relation between learning strategy and solution quality. This led us to consider how the local search probability interacts with the learning strategy and how this interaction affects the quality of solutions. An experiment was carried out to consider the effect of local probability on the solution quality for different population sizes (100, 400, 800, and 1200). The results of these experiments show that as probability of local search increases, the effect of learning strategy becomes apparent (figure 3). The graphs in figure 4 show that, when the probability of the local search is kept small, the quality of the solution is insignificantly affected by the learning strategy. As this probability increases, the quality of the solutions degrades with an increasing Lamarckian percentage in the learning process. This means using small local search probabilities for both algorithms, even with pure Lamarckian, can produce high quality solutions because the disruption to schema processing caused by these small probabilities is neglected and has no effect on global search process.

The results in Figure 4 show that a mixture of 20% Lamarckian and 80% Baldwinian produces the most stable solution quality for F2, regardless of the probability of the local search. A mixture of 75% Baldwinian and 25% Lamarckian produces the most stable solution quality for F1 (Figure 5) .The results from both hybrid algorithms show that a pure Baldwinian approach does not always produce the optimal solution quality and that the optimal learning strategy depends on the probability of local search.

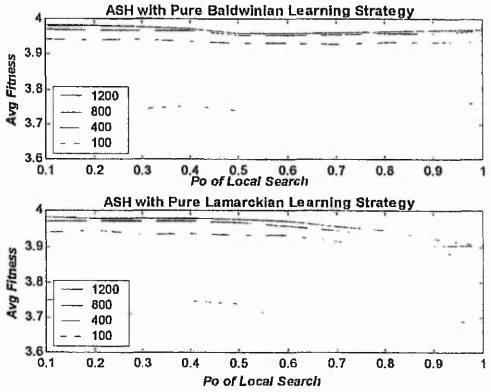

Figure 3: Effect of Learning Strategy and Search Probability on Solution Quality

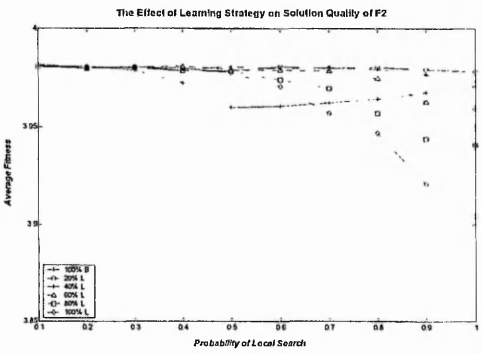

Figure 4: Solution Qualities for F2

Effect on Population Size

The aim of this experiment was to show how the probability of local search and learning strategy affect the minimum size requirements for both hybrids. The results were obtained by using bisection method. Starting with a population size of 10, the population size is doubled until the population converges to the desired solution quality. After the solution quality is attained, the population size is set midway between the current size and the last unsuccessful population size. This process is repeated until the difference between population sizes is less than or equal to 10. The stopping criterion was that 80% of the population converged within 0.000001 boundaries of the

global optimum. The settings of other parameters were as in the previous experiments.

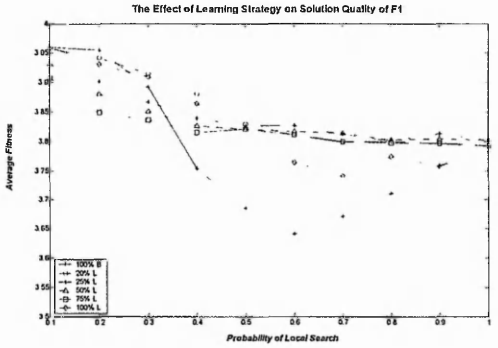

Figure 5: Solution Qualities for F1

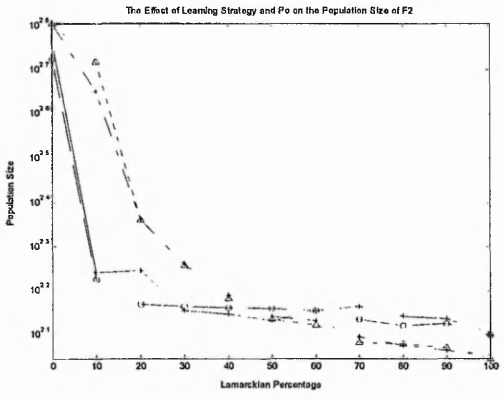

Figure 6: Effect of Learning Strategy and Search Probability on Population Size

The results of SSH and ASH on the second test function are similar. Figure 6 shows that, as the probability of local search increases, the population size decreases for a pure Lamarckian approach. On the other hand, with a pure Baldwinian strategy, the population size increases as the probability of local search increases. For a pure Baldwinian strategy with local search probability of more than 0.4, the population size exceeds that of a pure genetic algorithm (minimum population size=640). The results also show that the relationship between the local search probability and the change in the population size depends on the learning strategy used. For example, using a partial Lamarckian of 50% or more, an increase in the local search probability results in a decrease in population size. With a partial Lamarckian of less than 50%, an increase in the local search probability leads to an increase in the population size. For both hybrids, a decrease in population size leads to an increase in the convergence speed. In general, increasing the Lamarckian percentage decreases the population size and increases the convergence speed. The experiments also show that the solution quality of the pure Baldwinian approach is the optimal and the solution quality is degraded as both the Lamarckian percentage and the probability of local search increase. The solution quality for impure Baldwinian strategies, as shown in Figure 7, seems to be more dependent on the probability of local search than on the learning strategy.

The local search can decrease both the standard deviation of the population and the signal difference between the best and second-best solutions, since the population size depends directly on the standard deviation of the population and the signal difference. A decrease in the former decreases the population size and a decrease in the latter increases the population size.

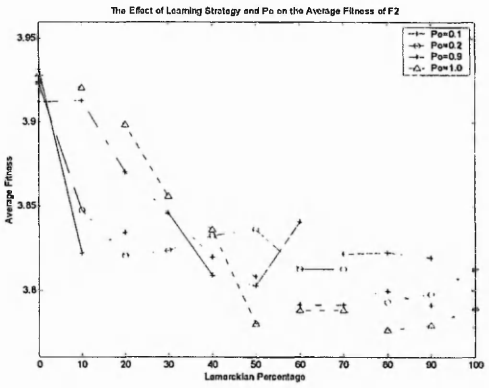

Figure 7: Effect of Lamarkian Proportion and Search Probability on Solution Quality

The increase in the population size requirements for the pure Baldwinian approach can be explained as follows. In a pure Baldwinian, the local search needs some help from evolution process to keep decreasing the ratio of standard deviation to signal difference. Pure Baldwinian can reduce this ratio at the end of the local search. However, in the next global iteration, if the value of local knowledge is insufficient to keep the global genetic algorithm reducing this ratio, the algorithm will lose some of its resources (i.e. local function evaluations) without reducing that ratio. In this case, a high probability of local search cannot lead to any reduction in the population size since it increases the probability of losing the algorithm's resources. However, a low local search probability reduces the probability of lost resources while increasing the probability of maintaining the reduction in

the above-mentioned ratio by the global genetic algorithm. In addition to the probability of local search, the effectiveness of pure Baldwinian in reducing the population size depends on the value of local knowledge and this depends on the method of local search and fitness landscape.

On the other hand, the opportunity to keep the gained reduction in this ratio is improved by using a partial Lamarckian strategy. As the percentage of Lamarckian increases, the probability of keeping this reduction increases. An increase in the probability of local search increases the probability of reducing the ratio and reducing the population size.

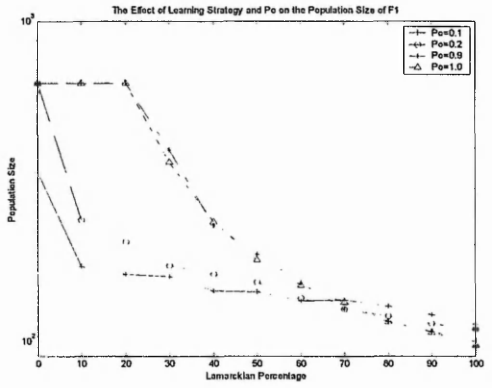

Figure 8: Effect of Learning Strategy and Search Probability on Population Size of F1

Figure 8 shows the results of running the same experiment on the first test function. For a Lamarckian percentage of 65% or more, an increase in the probability of local search results in a reduction in the population size. For other percentages, an increase in this probability leads to an increase in the population size requirements. The convergence speed depends on the population size; as the population size decreases the convergence speed increases. Comparison of Figures 6 and 8 shows that the switch point on the Lamarckian axis between increasing and reducing the population size is shifted from about 50% for F2 to about 65% for F1. This is due to the differences in the fitness landscape of both functions. While the local search can provide more significant local knowledge in F1 than in F2, an impure Lamarckian approach requires a more partial Lamarckian to accelerate the genetic assimilation process.

Additionally, the effect of the local search method on F1 is to enable any solution in a basin of attraction to climb to the exact local optimum in a single step. Consequently, increasing the probability of local search does not necessitate decreasing the signal difference between the best and second-best solutions. It also makes the selection process more difficult as the search process progresses when using a pure Baldwinian approach. In contrast, in F2 the local search method sends any point in the basin of attraction to a point near local optima and not to the local optimum itself.

The local search method can provide more significant local knowledge from the landscape of F1 than F2. This is why the reduction in the population size requirements of F1, using a pure Lamarckian approach, is greater than that of F2. This also makes the genetic assimilation process more difficult for F1 using a pure Baldwin effect compared with F2. The use of a partial Lamarckian can accelerate the genetic assimilation process. The exact value of the switch point depends on the value of the local knowledge.

CONCLUSIONS AND FUTURE WORK

The simulations show that using a low probability of local search and using a pure Lamarckian learning strategy can improve the convergence speed of the algorithm without disrupting the schema processing of the global genetic algorithms. They also show that, depending on the learning strategy used, increasing the probability of local search can decrease or increase the population size. As a result, the convergence speed is affected by the probability of local search. The results show that there is a relation between the probability of local search and the population size.

These experiments have attempted to provide an insight into how the probability of local search and learning strategy affect the population size requirements. We now plan to study how the population size can affect the optimal local search probability by developing a self-adaptive hybrid algorithm that encodes the number of local iterations within the chromosomes themselves and to study how their values propagate during the evolution process.

REFERENCES

Bobo, F. and Goldberg, D. 1997. "Decision Making in a Hybrid Genetic Algorithm", In T. Back (Ed.) Proc. the 1997 IEEE International Conference on Evolutionary Computation, pp. 121-125. IEEE Press

Espinoza, F., B; Minsker, B. and Goldberg, D. 2001. "A Self Adaptive Hybrid Genetic Algorithm". L. Spector, E. Goodman, A. Wu, W.B. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. Garzon, and E. Burke, editors. Proceedings of the Genetic and Evolutionary Computation Conference, GECCO'2001. San Francisco, Morgan Kaufmann Publishers.

Espinoza, F.; Minsker, B. and Goldberg, D. 2003. "Performance Evaluation and Population Size reduction for Self Adaptive Hybrid Genetic Algorithm (SAHGA) ", Proc. the Genetic and

Evolutionary Computation Conference, 2723, 922-933, Morgan Kaufmann

Goldberg, D. and Vosser, S. 1999. "Optimizing Global-Local Search Hybrids" Proc. the Genetic and Evolutionary Computation Conference, pp 220-228,Morgan Kaufmann.

Gruau, F. and Whitley, D. 1993."Adding Learning to the Cellular Development of Neural network: Evolution and Baldwin Effect", Evolutionary Computation, 1(3),213

Harik, G.; Cantu-Paz, E; Glodberg, D., and Miller, B. 1997. " The Gambler's Ruin Problem, Genetic algorithms, and the Sizing of Populations", In T. Back (Ed.) Proc. the 1997 IEEE International Conference on Evolutionary Computation, pp. 121-125.

Houck, C.; Joines, J.; Kay, M; and Wilson, J. 1997. "Empirical investigation of the benefits of partial Lamarckianism", Evolutionary Computation, v.5, n.1, 31- 60.

Hart, W. E. 1994 ."Adaptive Global Optimization with Local Search". Doctoral Dissertation, University of California, San Diego, CA

Mathias, K.; Whitley, D. ; Stork, C. and Kusuma, T. 1994. "Staged Hybrid Genetic Search for Seismic Data Imaging" ,1994 ,International Conference on Evolutionary Computation

Michalewicz, Z. and Nazhiyath, G., 1995. "Genocop III: A Co-evolutionary Algorithm for Numerical Optimization Problems with Nonlinear Constraints" , Proc. of the 2nd IEEE International Conference on Evolutionary Computation, Vol.2, 647-651

Press, W.; Teukolsky, S.; Vetterling, W. and Flannery, B. 1993. "Numerical Recipes in C ", Cambridge University Press

Reed, P.; Minsker, B. and Glodberg D. 2000. "Designing a Competent Simple Genetic algorithm for Search and Optimization". Water Resource Research, 36(12),3757-3761

Thierens, D., Goldberg, D. and Guimaraes P.1998.:" A. Domino Convergence, Drift, and the Temporal-Salience Structure of Problems. In The 1998 IEEE International Conference on Evolutionary Computation Proceedings, pp. 535-540, IEEE Press, New York, NY

Turney, P. 1996 "Myths and Legends of the Baldwin Effect" ,Poc. The ICML (13th International Conference on Machine Learning), 135-142.

Whitley, D.Gordon, V. and Mathias, K. 1994. "Lamarckian Evolution, the Baldwin Effect and Function Optimization", Parallel Problem Solving from Nature-PPSN III, 6-15.

BIOGRAPHIES

TAREK EL MIHOUB is a PhD student at the Nottingham Trent University. He obtained his MSc in Engineering Multimedia in 2002 from Nottingham Trent University. He worked as a teaching assistant in Al-Fatah University in Libya and as a manager of IT department in the Libyan Environment general authority. His current research is in the field of optimization, genetic algorithms, and artificial intelligence.

ADRIAN HOPGOOD is professor of computing and head of the School of Computing and Technology at the Nottingham Trent University, UK. He is also a visiting professor at the Open University. His main research interests are in intelligent systems and their practical applications. He graduated with a BSc (Hons) in physics from the University of Bristol in 1981 and obtained a PhD from the University of Oxford in 1984. He is a member of the British Computer Society and a committee member for its specialist group on artificial intelligence.

LARS NOLLE graduated from the University of Applied Science and Arts in Hanover in 1995 with a degree in Computer Science and Electronics. After receiving his PhD in Applied Computational Intelligence from The Open University, he worked as a System Engineer for EDS. He returned to The Open University as a Research Fellow in 2000. He joined The Nottingham Trent University as a Senior Lecturer in Computing in February 2002. His research interests include: applied computational intelligence, distributed systems, expert systems, optimisation and control of technical processes.

ALAN BATTERSBY obtained an MSc in Computer Science from Hatfield Polytechnic in 1977. Prior to joining the Computing department at Nottingham Trent University, Alan was a Computing Development Officer for Bedfordshire Education Authority. His research interests include: Fuzzy Logic applied to Robotics, wavelets, compression and the Internet.

Hybrid Genetic Algorithms: A Review

Tarek A. El-Mihoub, Adrian A. Hopgood, Lars Nolle, Alan Battersby

Abstract—Hybrid genetic algorithms have received significant interest in recent years and are being increasingly used to solve real-world problems. A genetic algorithm is able to incorporate other techniques within its framework to produce a hybrid that reaps the best from the combination.

In this paper, different forms of integration between genetic algorithms and other search and optimization techniques are reviewed. This paper also aims to examine several issues that need to be taken into consideration when designing a hybrid genetic algorithm that uses another search method as a local search tool. These issues include the different approaches for employing local search information and various mechanisms for achieving a balance between a global genetic algorithm and a local search method.

***Index Terms*—Genetic algorithms, evolutionary computation, hybrid genetic algorithms, genetic-local hybrid algorithms, memetic algorithms, Lamarckian search, Baldwinian search.**

I. Introduction

A genetic algorithm is a population-based search and optimization method that mimics the process of natural evolution. The two main concepts of natural evolution, which are natural selection and genetic dynamics, inspired the development of this method. The basic principles of this technique were first laid down by Holland [1] and are well described, for example, in [2],[3].

The performance of a genetic algorithm, like any global optimization algorithm, depends on the mechanism for balancing the two conflicting objectives, which are exploiting the best solutions found so far and at the same time exploring the search space for promising solutions. The power of genetic algorithms comes from their ability to combine both exploration and exploitation in an optimal way [1]. However, although this optimal utilization may be theoretically true for a genetic algorithm, there are problems in practice. These arise because Holland assumed that the population size is infinite, that the fitness function accurately reflects the suitability of a solution, and that the interactions between genes are very small [4].

In practice, the population size is finite, which influences the sampling ability of a genetic algorithm and as a result affects its performance. Incorporating a local search method within a genetic algorithm can help to overcome most of the obstacles that arise as a result of finite population sizes.

Incorporating a local search method can introduce new genes which can help to combat the genetic drift problem [5], [6] caused by the accumulation of stochastic errors due to finite populations. It can also accelerate the search towards the global optimum [7] which in turn can guarantee that the convergence rate is large enough to obstruct any genetic drift.

The Parallel Recombinative Simulated Annealing (PRSA) algorithm [8] fights the genetic drift problem in another way by combining the concept of the cooling schedule of simulated annealing [9], Boltzmann tournament selection [10], and standard genetic operators.

Due to its limited population size, a genetic algorithm may also sample bad representatives of good search regions and good representatives of bad regions. A local search method can ensure fair representation of the different search areas by sampling their local optima [11] which in turn can reduce the possibility of premature convergence.

In addition, a finite population can cause a genetic algorithm to produce solutions of low quality compared with the quality of solution that can be produced using local search methods. The difficulty of finding the best solution in the best found region accounts for the genetic algorithm operator's inability to make small moves in the neighborhood of current solutions [12]. Utilizing a local search method within a genetic algorithm can improve the exploiting ability of the search algorithm without limiting its exploring ability [7]. If the right balance between global exploration and local exploitation capabilities can be achieved, the algorithm can easily produce solutions with high accuracy [13].

Although genetic algorithms can rapidly locate the region in which the global optimum exists, they take a relatively long time to locate the exact local optimum in the region of convergence [14], [15]. A combination of a genetic algorithm and a local search method can speed up the search to locate the exact global optimum. In such a hybrid, applying a local search to the solutions that are guided by a genetic algorithm to the most promising region can accelerate convergence to the global optimum. The time needed to reach the global optimum can be further reduced if local search methods and local knowledge are used to accelerate locating the most promising search region in addition to locating the global optimum starting within its basin of attraction.

The improper choice of control parameters is another source of the limitation of genetic algorithms in solving real-world problems [16] due to its detrimental influence on the trade-off between exploitation and exploration. Depending on these parameters the algorithm can either succeed in finding a near-

Manuscript received February 3, 2006.

T. A. El-Mihoub, A. A. Hopgood, Lars Nolle, and A. Battersby are with School of Computing & Informatics, Nottingham Trent University. Nottingham, NG11 8NS, UK (phone: +44 (0)870 127 8429; fax: +44 (0)115 848 8365; e-mail: tarek.elmihoub, adrian.hopgood, lars.nolle, and alan. Battersby @ntu.ac.uk).

optimum solution in an efficient way or fail. Choosing the correct parameter values is a time-consuming task. In addition, the use of rigid, constant control parameters is in contradiction to the evolutionary spirit of genetic algorithms [17]. For this reason, other search techniques can be utilized to set the values of these parameters whilst the search is progressing.

In this paper, hybrid genetic algorithms are reviewed through presenting the different ways in which the roles of a search method and a genetic algorithm can be integrated. The aim of this presentation is not to classify hybrid genetic algorithms, but to shed light on the possible ways of combining a search method within the framework of a genetic algorithm. However, the reader can refer to [18] for an architectural taxonomy of combinatorial memetic algorithms (MA) [19] and to [20], where meta-heuristics are classified based on the design space and implantation space aspects.

This paper also aims to gain an insight into some of the design issues of hybrid genetic algorithms through reviewing the different mechanisms of utilizing local search information within genetic search and the various techniques to achieve a balance between exploration and exploitation.

II. A Complementary View

Hybrid genetic algorithms, as any hybrid system, are based on the complementary view of search methods [21 p.223]. Genetic and other search methods can be seen as complementary tools that can be brought together to achieve an optimization goal. In these hybrids, a genetic algorithm incorporates one or more methods to improve the performance of the genetic search. There are several ways in which a search or optimization technique can complement the genetic search.

A. Capability Enhancement

A technique can be utilized within a genetic algorithm to enhance search capabilities. A genetic algorithm is normally viewed as a global search method that can capture the global view of a problem domain. Different techniques can be incorporated within a genetic algorithm to improve its performance in different ways. When a genetic algorithm as a global search method is combined with a problem-specific method as a local method, the overall search capability can be enhanced. The enhancement can be in terms of solution quality and/or efficiency. This performance can also be improved by ensuring production of feasible solutions in the case of highly constrained problems. This paper focuses on the global local complementary view of genetic hybrids which have been variously referred to as memetic algorithms (MA) [19], genetic-local search methods [22], Lamarckian genetic algorithms [23], Lamarckian search, and Baldwinian search [24].

Function approximation techniques can also be incorporated in a genetic search to speed up the search. It is also possible to utilize other techniques to replace one or more of the genetic operators in order to overcome some of the problems that face genetic search.

1) Improving Solution Quality

Local search methods and genetic algorithms are usually viewed as two complementary tools. A local search algorithm's ability to locate local optima with high accuracy complements the ability of genetic algorithms to capture a global view of the search space. Holland [1], cited in [25], suggested that the genetic algorithm should be used as a pre-processor for performing the initial search, before invoking a local search method to optimize the final population. Bilchev and Parmee [26], for example, used their ant colony optimization [27] model for continuous search spaces as local search method to improve the quality of the solutions produced by a genetic algorithm in order to solve a real-world, heavily constrained, engineering design problem.

Performing local search on a genetic algorithm's population can introduce diversity and help to resist the genetic drift. It enables fair representation of different search areas in order to fight premature convergence. Incorporating a local search algorithm also introduces an explicit refinement operator which can produce high quality solutions.

2) Improving Efficiency

The efficiency of a local search in reaching a local optimum integrates the efficiency of a genetic algorithm in isolating the most promising basins of the search space. Therefore, incorporating a local search into a genetic algorithm can result in an efficient algorithm. The efficiency of the search can be enhanced in terms of the time needed to reach the global solution, and/or the memory needed to process the population.

a) Convergence Speed

A major concern in genetic algorithm design is efficiency in terms of the time needed to reach a solution of desired quality. In real-world problems, function evaluations are the most time-consuming part of the algorithm. For example, the designers of today's complex engineering systems usually rely on expensive computer analysis and simulation programs, where the execution time for a single function evaluation can be of the order of hours or days [28]. Finite element analysis (FEA), computational fluid dynamics (CFD), heat transfer and vehicle dynamic simulations are examples of such programs. Hybridization in addition to parallelization [29], time utilization [30], and evaluation relaxation (function approximation) can be used to speed up a genetic search [31].

Genetic algorithms often show significant improvements in search speed when combined with local search methods utilizing domain-specific knowledge [20], [32]. There is an opportunity in hybrid optimization to capture the best of both schemes [13]. This is the reason why genetic hybrids are being increasingly used to solve real-world problems. Different search methods have been mixed with genetic algorithms in real-world applications [15], [22], [33-37].

b) Population Size

Population size is crucial in a genetic algorithm. It determines the memory size and the convergence speed in serial genetic algorithms and affects the speed of search in the case of parallel genetic algorithms. Efficient population sizing

is critical for getting the most out of a fixed budget of function evaluations. The gambler's ruin model [38] was used to estimate the population size of genetic algorithms. This model was used to show that population size depends on two parameters, which can be affected by incorporating local search. The two parameters represent the standard deviation of the population and the signal difference between the best and second best building blocks. If a local search method is incorporated in such a way as to reduce the standard deviation of the population and to increase the signal difference between the best and the second best chromosome, the resulting hybrid can be efficient even with small population sizes. Espinoza et al. [39] showed the effect of a local search method on reducing the population size, compared to a pure genetic algorithm. El-Mihoub et al. [40] demonstrated the combined effect of probability of local search and learning strategy on the population size requirements of a hybrid.

3) Guarantee Feasible Solutions

In highly constrained optimization problems, the crossover and mutation operators generally produce illegal or infeasible solutions and hence waste search time. This problem can be solved by incorporating problem-specific knowledge. Problem-specific knowledge can be used either to prevent the genetic operators from producing infeasible solutions or to repair them.

The partial matched crossover (PMX) [41] was proposed for use in order-based problems to avoid the generation of infeasible solutions. Grefenstette et al. [42] suggested a heuristic crossover operator that could perform a degree of local search for the traveling salesman problem (TSP). Davidor [43] designed "analogous crossover" where local information is used to decide which crossover sites can produce unfit solutions. Heuristic crossover operators were used to solve a timetabling problem in order to ensure that the most fundamental constraints are never violated [44]. Freisleben and Merz [45] proposed the distance preserving crossover (DPX) to produce feasible solutions to solve TSP without losing diversity. They used the non- sequential 4-change [46] as a mutation operator for the same reason. Cycle crossover (CX) [47], order crossover (OX) [47], matrix crossover (MX) [48], modified order crossover (MOX) [49], edge recombination crossover (ERX) [50], 2-opt operator [51], 3-opt operator [51] and or-opt operators [51] are examples of crossover and mutation operators which have been developed for TSP. A special edge recombination crossover [52] has been constructed for the three-matching problem (3MP). The crossover operator has been replaced with the gene-pooling operator to produce feasible solutions when optimizing the number and positions of fuzzy prototypes for efficient data clustering [53].

A problem-specific knowledge search method can be used to recover the feasibility of solutions generated by the standard genetic operators. Repairing such solutions can help the genetic search to avoid the danger of premature convergence, which occurs when all or most solutions are infeasible [54], [55]. The force feasible heuristic operator [56] was used to solve the problem of scheduling aircraft landing times. Konak and Smith [57] combined a genetic algorithm with a cut-saturation algorithm for the backbone design of communication networks. They use a uniform crossover operator with a K-node-connectivity repair algorithm to repair infeasible offspring. Areibi and Yang [58] used repair heuristics in their proposed approach to solve VLSI circuit layout. The approach combines a hierarchical design technique, genetic algorithms, constructive techniques, and advanced local search. They also used the OX operator to avoid infeasible solutions in solving VLSI design problems.

4) Fitness Function Estimation

If the fitness function is excessively slow or complex to evaluate, approximation function evaluation techniques can be utilized to accelerate the search without disrupting search effectiveness. This is because genetic algorithms are robust enough to achieve convergence in the face of noise produced by the approximation process. Fitness approximation schemes replace high-cost accurate fitness evaluation with a low-cost approximate fitness assignment procedure. This can be achieved either by evolutionary approximation, where the fitness of a chromosome is estimated from its parents' fitness, or function approximation, where the fitness function is replaced by an alternate simpler model. Jin [59] provides a comprehensive survey on fitness approximation techniques.

The selection of an appropriate approximation model to replace the real function is an important step in ensuring that the optimization problem is solved efficiently. Neural network [21 ch. 8] models have widely been used for function approximation [60]. Willmes et al. [61] compared neural networks and the Kriging method for constructing fitness approximation models in evolutionary algorithms. Jin and Sendhoff [62] combined the k-nearest-neighbor clustering method and a neural network ensemble to estimate a solutions' fitness. Burdsall and Giraud-Carrier [53] used an approximation of the network's execution to evaluate solutions fitness instead of constructing a radial basis function network (RBF) to optimize the topology of a neural network. The approximation is based on an extension of the nearest-neighbor classification algorithm to fuzzy prototypes. Ankenbrandt et al. [63] implemented a system of fuzzy fitness functions, to grade the quality of chromosomes, representing a semantic net. The system is used to assist in recognizing oceanic features from partially processed satellite images. Pearce and Cowley [64] presented a study of the use of fuzzy systems to characterize engineering judgment and its use with genetic algorithms. They demonstrated an industrial design application where a system of problem-specific engineering heuristics and hard requirements are combined to form a fitness function.

5) Operation Substitution

Genetic algorithms present a methodological framework that is easy to understand and handle. This framework is open to the incorporation of other techniques [65]. It is possible to utilize other techniques to perform one or more of the genetic algorithm operations. These incorporated techniques can be used to replace either the crossover operator, mutation operator or both.

In probabilistic model-building genetic algorithms (PMBGA) or estimation of distribution algorithms (EDA) [66], a probabilistic model is utilized to learn the structure of a problem on the fly. This model is used instead of the standard genetic operators to ensure a proper mixing and growth of building blocks. These algorithms replace the standard crossover and mutation operators of genetic algorithms, by building a probabilistic model that estimates the true distribution of promising solutions. New potential solutions are then generated by sampling this model. Population based incremental learning (PBIL) [67], univariate marginal distribution algorithm (UMDA), compact genetic algorithm (CGA), bivariate marginal distribution algorithms (BMDA), factorized distribution algorithms (FDA) and the Bayesian optimisation algorithm (BOA) [68] are all examples of PMBGA that are reported to have a better search ability, than that of the simple genetic algorithm, in solving a broad class of problems [66]. Tsutsui et al. [69] proposed the aggregation pheromone system (APS), which introduced the concept of pheromone trail of the ant colony optimization [27] into the PMBGAs, to solve real-valued optimization problems.

Leng [70] proposed the guided genetic algorithm (GGA) which is a hybrid genetic system that borrows the concept of feature and penalties from the guided local search (GLS) [71]. The GGA modifies the fitness function by means of penalties to escape local optima. Two specialized crossover and mutation operators, which are biased by the penalties to change genes that are involved in more penalties, are used in order to explore the search space.

When a problem-specific representation is used in a genetic algorithm, the standard genetic variation operators are usually replaced with problem-specific operators. Hedar and Fukushima [72] replaced the ordinary crossover with a simplex crossover that produces a simplex offspring from mating simplex parents (is the dimension of the problem to be solved). In this hybrid, a mutation operator, which is more suitable for simplex representation, was used. Quantum-inspired genetic algorithms [73]-[75] borrow the concepts of quantum-bits and -states superposition from quantum computing. In these algorithms, the individuals are represented as a string of quantum-bits. Quantum-gates are then used to modify these individuals instead of crossover and mutation operators. The power of these algorithms comes from the great diversity they provide by using quantum coding. Each single quantum individual in reality represents multiple classical individuals. The results reported from using this hybridization to solve combinatorial and continuous optimization problems are promising.

Tan et al. [76] replaced the standard mutation operator by simulated annealing [9] to solve system identification and linearization problems. The results showed a more accurate search and faster convergence when compared with a pure genetic algorithm. The multi-step crossover (MSX) [77] was proposed to solve combinatorial optimization problems. Riopka and Bock proposed a collective learning genetic algorithm [78], in which an intelligent recombination based on the exchange of knowledge between chromosomes, is used to effectively find high quality solutions to combinatorial optimization problems. Magyar et al. [52] introduce several heuristic crossover and local hill-climbing operators to solve the three-matching problem. Fundamental to the technique here is the adaptation of the selected operator. Two fuzzy connective-base (FCB) crossover operators types (dynamic and heuristic) have been proposed in [79] for real-coded genetic algorithms to fight premature convergence problems.

B. Optimizing the Control Parameters

The setting of genetic algorithm control parameters is a key factor in the determination of the exploitation versus exploitation trade-off. Other techniques can be used to monitor the behavior of a genetic algorithm in order to adapt its control parameters to improve the search performance. The ability of fuzzy logic to represent knowledge in imprecise and non-specific ways enables it to be used to reason on knowledge that is not clearly defined or completely understood. This ability makes fuzzy logic a suitable choice for adapting the control parameters of a genetic algorithm. Fuzzy logic has allowed a small group of researchers to devise ways of optimizing performance and solution quality of genetic algorithms [80]. It is used to incorporate the many heuristics and techniques of experienced genetic algorithm researchers into fuzzy logic systems in order to adapt the control parameters. The goal of such a system is generally to avoid undesirable behaviors such as premature convergence and to speed up the convergence of the genetic algorithm [81].

It is also possible to incorporate a genetic algorithm within another technique to optimize control parameters, since genetic algorithms are in practice very effective optimization techniques. A genetic algorithm can be applied to the optimization of a neural network in a variety of ways. It can be utilized to adjust the neural network weights [82]-[84] their topology [85]-[88] and learning rules [89], [90]. For a comprehensive review of evolving neural networks the reader can refer to [91]. Karr [92] described an application to the cart-pole balancing system and used a genetic algorithm to evolve the membership functions of a fuzzy controller. The resulting, optimized fuzzy logic controller performed better than the controller based on membership functions designed by a human expert. These promising results have been confirmed by an application of the method for online control of a laboratory pH system with drastically changing system characteristics [93]. Genetic algorithms can also be used to automate the learning of fuzzy control rules [94]. They have also been used to optimize the control parameters of ant colony optimization algorithms [95]-[97].

III. Hybrid Design Issues

Incorporating a search method within a genetic algorithm can improve the search performance on the condition that their roles cooperate to achieve the optimization goal. There is an opportunity in hybrid optimization to capture the best of both schemes [13]. This opportunity depends on the design details of the hybrid genetic algorithm. There are several issues that

need to be taken into consideration when designing a hybrid genetic algorithm. Some of the design choices faced by hybrid practitioners while solving real-world problems are discussed here.

Due to their major impact on hybrid genetic performance, the discussion is concentrated on the strategies of utilizing local search information within a hybrid, and mechanisms that can be used to achieve a balance between exploration and exploitation. First, the relation between local search and learning, and its different models, are presented. Then, different techniques that can be used to achieve the optimal division of labor between the global genetic algorithm and the local search method are reviewed.

A. Local Search and Learning

Local search methods use local knowledge to improve a solution's chances to propagate its characteristics into the next generations. Due to the similarities in the role of the local search within the genetic search and the role of learning within the evolution process, the local search is usually viewed as a learning process.

The way by which gained information through local search is utilized within a hybrid genetic algorithm has a great impact on the performance of the search process. Two basic approaches based on biological learning models have been adopted to utilize local information; the Lamarckian approach and the Baldwinian approach [98]. There is also a third model, which is a mixture of the basic models and its effectiveness has been proven in solving real-world problems [55], [99]-[101].

1) Lamarckian Learning

The Lamarckian approach is based on the inheritance of acquired characteristics obtained through learning. This approach forces the genetic structure to reflect the result of the local search. The genetic structure of an individual and its fitness are changed to match the solution found by a local search method. In the Lamarckian approach, the local search method is used as a refinement genetic operator that modifies the genetic structure of an individual and places it back in the genetic population.

Lamarckian evolution, in spite of being recognized as never occurring in biological systems due to the lack of a mechanism to accomplish it, can be simulated in a computer in order to shed light on issues of general evolvability. Lamarckian evolution can accelerate the search process of genetic algorithms [102]. On the other hand, by changing the genetic structure of individuals, it can disrupt schema processing which can badly affect the exploring abilities of genetic algorithms. This may lead to premature convergence [102]. When a Lamarckian approach is adopted, inverse mapping from phenotype to genotype is required. The inverse mapping may be computable in many simple applications. However, for real-world problem solving, the computation will typically be intractable [103]. Most of hybrid genetic algorithms that repair chromosomes to satisfy constrains are Lamarckian and the technique has been particularly effective in solving TSP [24].

2) Baldwinian Learning

The Baldwin learning allows an individual's fitness to be improved by applying a local search, whereas the genotype remains unchanged. In this way, it improves the solution's chances to propagate its structure to the next generations. Like natural evolution, learning does not change an individual's genetic structure, however it increases its chances of survival. The Baldwinian approach, in contrast to the Lamarckian one, does not allow parents to pass their learned or acquired characteristics to their offspring. Instead, only the fitness after learning is retained. A local search method in the Baldwinian approach is usually used as a part of the individual's evaluation process. The local search method uses local knowledge to produce a new fitness score that can be used by the global genetic algorithm to evaluate the individual's ability to be improved.

The Baldwin effect is somewhat Lamarckian in its results although it uses different mechanisms [103]. It explains interactions between learning and evolution by paying attention to balances between benefit and cost of learning. The Baldwin effect consists of the following two steps [104]. In the first step, learning gives individuals the chance to change their phenotypes to improve their fitness. Individuals, who found learning useful and help their fitness to improve, will spread in the next population. In the second step, if the environment is sufficiently stable, the cost associated with learning results in selection favoring individuals that have the traits, which are acquired by others through learning, already coded into their genotype. Through this mechanism, called genetic assimilation, learning can accelerate the genetic acquisition of learned traits indirectly. A critical precondition for genetic assimilation appears to be a strong correlation between genotype and phenotype space so that nearness in the phenotype space implies nearness in the genotype space [105]. Otherwise, the acquired traits have little chance of eventually becoming encoded in the genome via chance through genetic operations.

Hinton and Nolan [98] illustrated how the Baldwin effect can transform the fitness landscape of a difficult optimization problem into a less difficult one, and how the genetic search is attracted toward the solution found by learning. Gruau and Whitley [11] showed how local search can change the landscape of fitness function into flat landscapes around the basin of attraction. This change in fitness landscape is known as the smoothing effect. They demonstrated the impact of the smoothing effect on the search process. This learning strategy could be more effective but slower than the Lamarckian approach, since it does not disrupt schema processing of genetic algorithms [102]. Baldwinian search can also have the effect of obscuring genetic differences and, thus, hindering the evolution process [105]. This is known as the hindering effect. Essentially, this occurs as a result of different genotypes mapping to the same or similar phenotypes (as a result of the smoothing effect) with equivalent fitness scores being produced. The genotypes cannot be effectively discriminated according to their fitness values without considering the learning cost and the evolution of effective solutions is

hindered. The hindering effect can also obstruct the ability of the Baldwinian search to self-adapt the local-search-duration control parameter [106]. The Baldwinian effect can aggravate the problem of multiple genotype to phenotype mappings [24], [99]. This problem can also waste the resources of hybrids that use clustering techniques in the genotype domain to reduce unnecessary local search, in contrast to the Lamarckian approach which has been shown to help alleviate this problem [107].

Hart et al. [108] pointed to the importance of considering the cost of learning, which has been ignored by most researchers when studying the impact of the Baldwinian strategy on the hybrid search by analyzing its performance based on the number of generations of the genetic algorithm only. Learning can introduce a computational cost which overweighs its benefits in search.

3) Hybrid Lamarckian-Baldwinian Models

Hybrid Lamarckian-Baldwinian models are created with a view towards combing the advantages of both forms of learning models [55]. The combination of the Baldwinian and the Lamarckian approaches can be done at two different levels. Hybridization can be used at the individual-level, where some individuals evolve using the Lamarckian approach while the other individuals evolve using the Baldwinian approach [99], [100]. Houck et al. [99] found that this form of partial Lamarckian approach outperformed both the pure Lamarckian and the pure Baldwinian approaches on a selected set of test problems. The other level is the gene-level, where a number of genes evolve using the Lamarckian strategy and the remaining genes evolve using the Baldwinian approach [101]. This approach was used to solve the sorting network problem. It can reduce the problem search space and help to produce an efficient search [101].

The adoption of any form of learning in a hybrid genetic algorithm has a great impact on its performance. Several researchers have investigated how these different leaning strategies affect the performance of hybrid genetic algorithms by comparing them with pure genetic algorithms. Gruau and Whitley [11] compared Lamarckian, Baldwinian and pure genetic algorithms in evolving the architecture and the weights of neural networks that learn Boolean functions. They conclude that using either form of leaning is better than using a pure genetic algorithm. Orvosh and Davis [55] found that 5% partial Lamarckian is the optimal learning strategy to solve the survival network design problem and the graph coloring problem. Michalewicz and Nazhiyath [109] replaced 20% of the repaired solutions in their hybrid algorithm to solve numerical optimization problems with nonlinear constraints. Bala et al. [110] showed how the Baldwin effect can improve the performance of a genetic algorithm when integrated with a decision tree in order to evolve useful subsets of discriminatory features for recognizing complex visual concepts. However, Ku and Mak [111] found that only using Lamarckian evolution improved the performance of genetic algorithm in evolving recurrent neural networks. They also concluded that effective hybridization depends on the local search method used and the learning frequency. Houck et al. [99] used seven problems to compare the performance of different learning strategies. Their investigation concluded that neither the pure Lamarckian nor pure Baldwinian strategy was found to be consistently effective. It was discovered that the 20% and 40% partial Lamarckian search strategies yielded the best mixture of solution quality and computational efficiency based on a minmax criterion (i.e. minimizing the worst case performance across all test problems instance). Sasaki and Tokoro [112] found that adaptation by Lamarckian evolution was much faster for neural networks than Darwinian evolution in a static environment. However, when the environment changed from generation to generation, the Darwinian evolution was superior. Julstrom [24] reported that Baldwinian strategies perform poorly in solving the 4-cycle problem compared to a pure genetic algorithm and their effectiveness deteriorates with an increasing use of learning in contrast to Lamarckian strategies. He also found that applying Lamarckian leaning to all the individuals produced the most effective results. Joines et al. [100] found that using the pure Lamarckian approach (100% Lamarckian) produced the best convergence speed to the best known solution when solving the cell formation problem. Espinoza et al. [112] used 75% partial Lamarckian as the optimal leaning strategy in their hybrid to optimize two continuous functions. El-Mihoub et al. [40] investigated the combined effect of probability of local search and leaning strategy on the hybrid performance and found that combing a low probability of local search with the pure Lamarckian learning strategy can improve the convergence speed without disrupting the schema processing. Ishibushi et al. [114] found that the 5% partial Lamarckian worked well on the multi-objective 0/1 knapsack problem using a single population model, however, the 50% partial Lamarckian was the optimal choice using the island model.

The effectiveness of adopting the pure Lamarckian approach, the pure Baldwinian approach, or any mixture of them in a hybrid is affected by the fitness landscape, the representations, the percentage of population performs local search and local search method used [40], [99], [100], [103], [109], [114].

B. Balance between Global and Local Search

The hybrid algorithm should strike a balance between exploration and exploitation, in order to be able to solve global optimization problems. According to the hybrid theory [115], solving an optimization problem and reaching a solution of desired quality can be attained in one of two ways. Either the global search method alone reaches the solution or the global search method guides the search to the basin of attraction from where the local search method can continue to lead to the desired solution. In the genetic-local hybrid, the main role of the genetic algorithm is to explore the search space in order to either isolate the most promising regions of the search space, or, to hit the global optimum. However, the main role of the local search method is to exploit the information gathered by the global genetic algorithm. The division of the hybrid's time between the two methods influences the efficiency and the effectiveness of the search process. The optimal division of

the algorithm's time is an important issue that is faced the designers of hybrid genetic algorithms.

Although the aim of combining a global genetic algorithm and a local search method is to get the best out of the exploring ability of the former, and the efficiency of the latter in reaching local optima, the two methods can interact in a more complicated way than the one described above. Rosin et al. [116] argued that the mutation operator in a hybrid plays a different role than it does in a pure genetic algorithm. The local refinement requirement of the mutation operator becomes unnecessary in the existence of an explicit local operator allowing the mutation operator to take a more exploratory role. Land [117] suggested using larger mutations, at least large enough to move from one basin to another, in cases where each individual of the population is completely locally optimized. He went further, when he argued that local search obviates the need for crossover in solving the graph bisection problem, because local search is able to build the very same building blocks that the crossover would otherwise combine.

The exploring ability of the genetic algorithm can be further improved by utilizing local search to ensure fair representation of different regions of a search. This can improve the ability of the genetic algorithm to direct the search to the most promising regions of the search space. Once the algorithm has guided the search to the basin of attraction of the global optimum, utilizing local search can further improve the search to produce an effective optimization algorithm. The first goal of the hybridization, which is the effectiveness of search, can be satisfied if a genetic algorithm and a local search method cooperate in the manner mentioned above. However, there are other more destructive forms of interaction. For example, the mutation and crossover operators can disrupt good and complete local solutions which may waste algorithm resources and produce an inefficient search. The Lamarckian local search can disrupt the schema processing of the genetic algorithm which may lead to premature convergence and produce an ineffective search.

In addition to the role of genetic operators in systemically exploring the search space, they perform some form of local search with relative low cost compared to the more accurate local search methods. The improper use of the expensive local search in a hybrid can waste algorithm resources. The algorithm should be able to decide wisely on both methods, especially when both can achieve the desired task, taking into account the benefits and costs of their utilization. The condition of an appropriate use of both methods in addition to the condition of interacting in a cooperative way should be satisfied in order to produce an effective and efficient search algorithm.

Researchers have proposed different techniques to enable the hybrid to mix both methods wisely or at least to reduce the consequences of the improper use of the expensive local search. These techniques are based on modifying the different parameters of a local search method within a hybrid. Modifying the parameters of the local search, such as the frequency of local search, the duration of local search, and the probability of local search can help the hybrid to strike the balance between the two search methods.

1) Frequency of Local Search

The number of continuous uninterrupted generations that a genetic algorithm performs before applying local search is usually referred to as the frequency of local search. In the traditional hybrid genetic algorithm, the frequency of local search is 1, for example. The staged hybrid genetic algorithm [118], [119] was designed to separate the two search methods into two distinct stages by increasing the frequency of the local search in order to minimize the interference between the two search methods. Mathias and Whitely [118] used a local search frequency of 2 to solve the TSP. However, in a hybrid algorithm to solve the static correction problem [119], the genetic search algorithm was allowed to continue uninterrupted for ten generations before applying a single iteration of waveform steepest ascent iteration to each individual in the population. This hybrid algorithm produced solutions with improved quality of 5% and additional savings in time compared with the traditional hybrid genetic algorithm. Espinoza et al. [113] conducted a set of experiments to find the optimal local search frequency for two two-dimensional continuous test functions and they found that the optimal frequency of local search for these test functions was 3.

The optimal frequency of local search is function dependent and varies with time because the optimal time that should be spent on local and global search algorithms depends on the distribution of individuals in the population. Syrjakow and Szczerbicka [120] studied the optimal switch point between the genetic algorithm and local search to fine-tune the solution found by the pre-optimizer genetic algorithm. They studied three criteria: the number of function evaluations, the convergence speed of the genetic algorithm, and the regional accumulation of search points indicating the convergence toward a specific region in the search space so as to determine the optimal switch point. The convergence speed criterion produced the highest efficiency in their experiment. Lobo and Goldberg [13] addressed the problem of deciding between global search and local search in order to make the most out of either technique. They tried to answer the question, "when should the local search be used and when should the global genetic algorithm be used to achieve the maximum possible efficiency?" They viewed the problem as a two armed bandit problem where the payoff of each bandit is unknown and changes with time. They presented a model for efficient hybridizing based on the concept of probability matching. This model can be viewed as an adaptive technique that adjusts the frequency of local search depending on the efficiency of both genetic and local techniques as the search progresses. Tuson and Ross [121] used a similar model to adapt the operator probability in their cost based operator rate adaptation. They used their model to select the use of a mutation or crossover operation in a pure genetic algorithm. The same technique has been used to solve the three-matching problem [52], where an adaptive hybrid algorithm selects one operator from eight recombination and local search operators based on their

current and past benefit-cost ratio.

Espinoza et al. [113] used the change in coefficient of variation of the fitness function to determine whether the genetic algorithm is exploring new regions of the search space or exploiting the already visited regions. Based on that, the algorithm selects to perform either a genetic or a local iteration. The algorithm relies on the local search role to improve the sampling of the new regions that are being explored in the case of any increase in that coefficient. Once the search has branched to a local search, the fitness improvement-cost ratio of both the last genetic and the local iterations and the maximum number of local iterations are used to decide on continuing the local search or going to the global search. Their experiments showed that the algorithm is more efficient than a pure genetic algorithm and is stable against a greater range of parameter settings than the standard staged hybrid genetic algorithm.

Hacker et al. [28] proposed an approach that switches between global genetic and local search, based on the local topology of the search space. The basic idea of this approach ignores the role of local search in improving the sampling ability of the genetic algorithm. It concentrates on the efficiency of local search, i.e. finding the optimum once the global genetic algorithm has defined its basin of attraction. The utilization of the relative homogeneity of the population and regression analysis to determine whether the search is exploring a single basin or multiple basins was investigated. The coefficient of variance of both the fitness and phenotype is used to quantify the relative homogeneity of the population. A decrease in the values of the coefficient of variance indicates that the genetic algorithm has converged to a small area of the search space and the search process can therefore be made more efficient by switching to a local search. In contrast, an increase in its value indicates that a new region of the search space is being explored and hence there is less need to use a local search. Regression analysis has also been used to determine when to switch between global and local techniques. The value of the error of fitting the population of solutions to a second-order surface can indicate whether the genetic algorithm is exploring multiple basins or a single basin in the search space. Depending on the value of that error, the algorithm decides to switch to a local search or continue the global search. They concluded that utilizing local search could be helpful for small search spaces in the early stages of search due to their role in helping the genetic algorithm to define the most promising regions of the search space. However, for large and complicated search spaces, their role is limited to accelerating finding of the global optimum once the genetic algorithm isolates the most promising region and can be helpful in later stages of the search.

2) Duration of Local Search

Local search duration influences the balance between the global exploration of genetic algorithms and local refinement of the neighborhood search method in hybrid genetic algorithms [122], [123]. A hybrid with long local search duration will execute fewer generations of the genetic algorithm than a hybrid with shorter local duration, if both terminate after the same number of function evaluations.

On combinatorial domains, a local search can be performed until a solution converges to a local optimum. However, on continuous domains, the local search is typically truncated before reaching a local optimum when its step length becomes too small. Performing local search until a solution converges to a local optimum, which is referred to as complete local search, may lead to the loss of population diversity [102] depending on the learning strategy used. Hybrid genetic algorithms that adopt the pure Lamarckian approach are more prone to loss of diversity than others which utilize other learning techniques.

Applying a complete local search on costly function evaluations can also be expensive. However, there is a certain class of problems, decomposable fitness problems [124], where calculating the fitness of a solution given the fitness of its neighbor, is significantly less computationally expensive than computing its fitness from scratch. TSP is an example of this group of problems where computing the length of a tour that shares most of its edges with another tour, whose length is already known, is much cheaper than computing the length of a complete tour. Radcliffe and Surry [124] argued that hybrids are more suitable for problems exhibiting this property.

A few studies have been conducted which investigate the optimal duration of local search. Hart [7] found that using a short duration of local search produced the best results for the Griewank functions [125], whereas a long duration produced better results for the Rastrigin functions [126]. Rosin et al. [116] experimented with very short and very long local search durations in a hybrid to optimize the drug-docking configuration. Both durations were found to yield similar performance. Hart et al. [122] concluded that duration of local search is an important factor and hybrid genetic algorithms with long local searches will be most effective for nontrivial problems.

The high cost of a complete local search on expensive function evaluations makes any improper use of the local search difficult to recover from. However, the recovering from any misuse of partial local search is still possible. Partial local search is more suitable for hybrids that decide on a global or a local approach depending on the current state of the search and the previous performance of both methods. In this case, where there is a possibility of misjudgment in some circumstances, the use of partial local search gives the hybrid a higher chance to recover from such errors than using a complete local search.

3) Probability and Selection of Local Search

In any hybrid algorithm, a local search can be applied to either every individual in the population or only few individuals. In traditional hybrid genetic algorithms, a local search is applied to every individual in the population. However, applying a local search to every individual in the population on costly function evaluations can waste resources without providing any more useful information. In this case, the local search can be applied to individuals that fall in the same basin of attraction of the search space, whereby producing the same local optimum. Applying a local search to

a large fraction of the population can limit exploration of the search space by allowing the genetic algorithm to evolve for a small number of generations. The possibility of applying local search on more than one individual from the same basin can be reduced by performing local search on only a small fraction of the population. This also lowers the chances of applying an unnecessary local search on individuals that fall in non-promising regions of the search space. Deciding upon the optimal fraction of the population which should perform local search, and the basis on which these individuals are chosen, has a great impact on the performance of a hybrid.

Hart [7] investigated the impact of the fraction of the population that undergo local search on the performance of real-coded genetic algorithms. He found that a relation exists between this fraction, the population size and the performance of the hybrid. He also found that performing local search on small fractions could be more efficient when using larger populations and those large fractions can help to reflect the search space characteristics when using small populations. He concluded that a more selective use of local search could improve the efficiency of hybrids. Hart and Belew [127] studied the impact of the local search probability on the efficiency of hybrids. Their studies indicate that the probability of local search should be kept low in the initial stages and incremented in later generations. The population diversity in the initial stages of genetic algorithm enables good sampling of the search space. However, as the diversity diminishes in the later stages, the sampling ability of the genetic algorithm requires additional help from the local search.

Different techniques, such as tuning, distribution-based [7], fitness-based [7] techniques, and local search potential [117], have been proposed to decide on the optimal fraction of the population that should perform a local search. These techniques aim to reduce unnecessary local searches. However, they differ in the way they select individuals that perform the local search.

a) Tuning Technique

In the tuning technique, a primary experiment is conducted in order to find the optimal fraction of the population that should perform local search. This fraction is usually referred to as the probability of local search. This value is then used to run the real experiment and remains fixed during the run. Typically, the individuals that undergo local search are chosen uniformly at random. Rosin et al. [116] applied local search to 7% of the population in each generation in their hybrid to solve the docking problem. In Land et al. [128], only 5% of randomly selected individuals of the population perform a Marquardt-Levenberg local search in their hybrid to determine the basic parameters that describe the structure of a semiconductor wafer. Hart et al. [122] and Morris et al. [17] applied local search to 6% of the population. Espinoza et al. [113] found applying local search on 10% of the population produces the best efficiency for both their adaptive hybrid algorithm and the standard staged hybrid algorithm. In their adaptive hybrid genetic algorithm, this value is used as an initial value for the probability of local search, which is reduced by a specific value after applying local search. In a hybrid to solve TSP, Krasnogor and Smith [32] applied their adaptive local search method with a probability of 1.0 to each individual in the population, except the one with the best fitness.

b) Distribution-based Technique

Distribution-based techniques modify the probability of local search based on the distribution of individuals in the population. The motivation for these techniques is to ensure that only one individual from each basin of attraction in the search space can undergo local search. These techniques can improve the sampling ability of the hybrid by preventing bad representatives of good regions from misguiding the global genetic algorithm.

Hart [7] used the F statistic as a measure of distance over the space of genotypes to adapt the probability of local search. Joines and Kay [107] combined evolutionary algorithms with random linkage and borrowed the concept of short memory from tabu search [129] to avoid performing unnecessary local search on non-promising regions of the search space. The authors defined tabu hyperspheres around the offspring of the genetic algorithm to reduce the number of wasted function evaluations owing to the rediscovery of the same local optimum. The probability of local search of each offspring depends on the distance to the nearest tabu region. By decreasing the size of these tabu hyperpheres as the search progress, the algorithm can intensively search the most promising regions of the search space. This in turn can help to find the exact local optimum of the region which also represents the global optimum of the search space. The authors compared their hybrid using the Lamarckian learning approach with a pure genetic algorithm, and the standard hybrid genetic algorithm where each offspring perform local search using two different learning strategies. They reported that their hybrid outperformed other algorithms in terms of both solution quality and computation effort. Martinez-Estudillo et al. [130] selected individuals for local search using clustering techniques to optimize the structure and the weights of product-unit based neural networks. The results showed that the clustering approach was able to perform better than similar algorithms that do not use clustering analysis.

c) Fitness-based Technique

A fitness-based technique adaptively calculates the probability with which local search is applied. This technique uses the fitness information in the population to bias the local search towards individuals that have a better fitness. The local search probability of each individual is modified based on the relationship of its fitness to the fitness of other individuals. These methods assume that individuals with better fitnesses are more likely to be in the basins of attraction of the most promising regions. This assumption ignores the dynamic of genetic algorithms and the cumulative effect of applying local search on successive generations which can aggravate the sampling ability of the global genetic algorithm and can

misguide the search. For example, if a promising region of the search space is represented poorly by an individual with under-average fitness and, in the same population, a non-promising region is represented by individuals with over-average fitness, the representative of the non-promising region will have more chance to perform local search and improve its chances of survive.

Hart [7] found no statistical differences between the results obtained by applying fitness-based selection and the results of fixed probability of local search. Espinoza et al. [131] used a clustering technique that is tailored to the three different stages the authors have defined for constrained problems to adapt the probability of local search. In the first stage, where all the solutions are infeasible, and the last stage, where all the solutions are feasible, the authors experimented with clustering the individuals depending on their fitness. The selection was performed by means of Latin-hypercube sampling from clusters which had formed. In the second stage, where a few individuals are feasible, the probability of local search is proportional to the number of feasible solutions in the population. The results showed that the algorithm, which is based on a fitness clustering technique, is more reliably faster than the adaptive hybrid genetic algorithm with fixed starting local search probability. Lozano et al. [132] proposed a simple adaptive scheme which sets the probability of local search of each individual to either 1.0 or 0.0625 depending on the individuals fitness compared to the fitness of the current worst individual in the population. The authors concluded that this adaptation mechanism allows the balance between the global genetic search and the local search to be adjusted according to the particularities of the search space, thus allowing significant improvements in the performance for different classes of problems.

d) Local Search Potential Technique

The local search (LS potential) potential selection mechanism has been proposed by Land [117] to decide which individuals should perform the local search. Land suggested that biasing the local search towards individuals that can be most efficiently improved by local methods makes the most effective use of local search. The least easily improved solutions are likely to be those at or near to the local optimum and it is inappropriate to expend effort on fine refinement, as long as there are large differences in the population's fitness. In this way, the scheme biases the hybrid towards more exploration. As the population gets closer to the optima, this mechanism allows local search to progress to the next level of refinement. In his algorithm, he used the past local search effectiveness as a measure to estimate future effectiveness.

Different techniques have been used to control the different parameters of the local search in order to strike a balance with the global genetic methods. Most of the controlling techniques which are described by Eiben et al. [17] for controlling the parameters of evolutionary algorithms have been applied to the local search control parameters in a hybrid.

The self-adaptation techniques are reported to be successfully used to decide between different local search methods in solving the OneMax problem, NK-Landscapes, and TSP [134]. The self-adaptation technique has also been used to adapt the duration of local search in a hybrid through encoding the number of local iterations into chromosomes [106]. In this way, the global genetic algorithm decides on the individuals that should perform a local search and on its duration.

IV. Summary

In this paper, we have tried to shed some light on the effectiveness and efficiency of hybridizing genetic algorithms with various techniques through reviewing some of the wide variety of hybrid genetic approaches. These approaches show that hybridizing is one possible way to build a competent genetic algorithm [135] that solves hard problems quickly, reliably and accurately without the need for any forms of human intervention. Hybridization has been utilized to construct competent genetic algorithms that belong to two of the three main approaches for building competent genetic algorithms, i.e., perturbation, linkage adaptation, and probabilistic model-building [136]. The collective learning genetic algorithm is an example of a competent genetic algorithm that employs specifically designed representation and operators for adapting genetic linkage along with the evolutionary process. Other search and optimization methods can also be used to adapt genetic linkage. Probabilistic Model-Building Genetic Algorithms (PMBGA) are examples of probabilistic model builders which learn genetic linkage via building models based on the current population.

Hybridization is also one of the four main techniques for efficiency enhancement of genetic algorithms. Hybridization can also be used as a tool to achieve evaluation relaxation, which in turn is another main technique for efficiency enhancement.

The ability of a genetic-local hybrid to solve hard problems quickly depends on the way of utilizing local search information and the mechanism of balancing genetic and local search. By reviewing the different hybrid approaches, some of the important factors that affect the hybrid performance have been presented. This review shows that there is a trend towards adapting some of the hybrid design choices through adapting the control parameters associated with these choices while the search is progressing. Different adaptation techniques have been used to adapt the selection of a local search method, the selection of individuals for a local search, the duration of local search, the learning strategy, and other design aspects.

References

[1] J. Holland, *Adaptation in Natural and Artificial Systems*: The University of Michigan, 1975.

[2] K. De Jong, "An analysis of the behavior of a class of genetic adaptive systems," Doctoral Dissertation. Ann Arbor: The University of Michigan, 1975.

[3] D. E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning: Addison-Wesley, 1989.

[4] D. Beasley, D. R. Bull, R, and R. Martin, "An overview of genetic algorithms: part 1, fundamentals," University Computing, vol. 15, pp. 58-69, 1993.

[5] H. Asoh and H. Mühlenbein, "On the mean convergence time of evolutionary algorithms without selection and mutation," in Parallel Problem Solving from Nature, PPSN III, Y. Davidor, H.-P. Schwefel, and R. Manner, Eds. Berlin, Germany: Springer-Verlag, 1994, pp. 88–97.

[6] D. Thierens, D. Goldberg, and P. Guimaraes, "Domino convergence, drift, and the temporal-salience structure of problems," in 1998 IEEE International Conference on Evolutionary Computation Anchorage, USA: IEEE, 1998, pp. 535-540.

[7] W. E. Hart, "Adaptive global optimization with local search," Doctoral Dissertation. San Diego: University of California 1994.

[8] S. Mahfoud and D. Goldberg, "Parallel recombinative simulated annealing: a genetic algorithm," Parallel Computing, vol. 21, pp. 11-28, 1995.

[9] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," Science, vol. 220, pp. 671-680, 1983.

[10] S. W. Mahfoud, "Boltzmann selection," in Handbook of Evolutionary Computation, T. Back, D. B. Fogel, and Z. Michalewicz, Eds.: IOP Publising Ltd and Oxford University Press, 1997, pp. C2.5:1-4.

[11] F. Gruau and D. Whitley, "Adding learning to the cellular development of neural network: evolution and Baldwin effect," Evolutionary Computation, vol. 1, pp. 213-233, 1993.

[12] C. Reeves, "Genetic algorithms and neighbourhood search," in Evolutionary Computing, AISB Workshop, vol. 865 Lecture Notes in Computer Science, T. C. Fogarty, Ed. Leeds, UK: Springer-Verlag, 1994, pp. 115-130.

[13] F. G. Lobo and D. E. Goldberg, "Decision making in a hybrid genetic algorithm," in IEEE International Conference on evolutionary Computation. Piscataway, USA: IEEE Press, 1997, pp. 122-125.

[14] K. De Jong, "Genetic algorithms: a 30 year perspective," in Perspectives on Adaptation in Natural and Artificial Systems, L. Booker, S. Forrest, M. Mitchell, and R. Riolo, Eds.: Oxford University Press, 2005.

[15] P. Preux and E.-G. Talbi, "Towards hybrid evolutionary algorithms," International Transactions in Operational Research, vol. 6, pp. 557-570, 1999.

[16] K. Deb, "Limitations of evolutionary computation methods," in Handbook of Evolutionary Computation, T. Bäck, D. B. Fogel, and Z. Michalewicz, Eds.: IOP Publishing and Oxford University Press, 1997, pp. B2.9.

[17] A. E. Eiben, R. Hinterding, and Z. Michalewicz, "Parameter control in evolutionary algorithms," IEEE Transactions on Evolutionary Computation, vol. 3, pp. 124-141, 1999.

[18] N. Krasnogor and J. Smith, "A tutorial for competent memetic algorithms: model, taxonomy and design issues," IEEE Transactions on Evolutionary Computation, vol. 9, pp. 474-488, 2005.

[19] P. Moscato, "On evolution, search, optimization, genetic algorithms and martial arts: towards memetic algorithms," California Institute of Technology 1989.

[20] E. Talbi, "A Taxonomy of hybrid metaheuristics," Journal of Heuristics, vol. 8, pp. 541–564, 2002.

[21] A. A. Hopgood, Intelligent Systems for Engineers and Scientists, 2nd ed: CRC Press, 2001.

[22] T. Yamada and C. Reeves, "Solving the C_{sum} permutation flowshop scheduling problem by genetic local search," in International Confrenence on Evolutionary Computation. Anchorage, USA, 1998, pp. 230-234.

[23] G. M. Morris, D. S. Goodsell, R. S. Halliday, R. Huey, W. E. Hart, R. K. Belew, and A. J. Olson, "Automated docking using a Lamarckian genetic algorithm and an empirical binding free energy function," Journal of Computational Chemistry, vol. 19, pp. 1639-1662, 1998.

[24] B. Julstrom, "Comparing Darwinian, Baldwinian, and Lamarckian search in a genetic algorithm for the 4-cycle problem," in the 1999 Genetic and Evolutionary Computation Conference, Late Breaking Papers, S. Brave and A. S. Wu, Eds. Orlando, USA, 1999, pp. 134-138.

[25] Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs third ed: Springer-Verlag, 1996.

[26] G. Bilchev and I. C. Parmee, "The ant colony metaphor for searching continuous design spaces," in AISB Workshop on Evolutionary Computing, vol. 993, Lecture Notes In Computer Science, T. C. Fogarty, Ed. Sheffield, UK: Springer Verlag, 1995, pp. 25-39.

[27] M. Dorigo, V. Maniezzo, and A. Colorni, "Positive feedback as a search strategy," Politecnico di Milano, Milan 1991.

[28] K. A. Hacker, J. Eddy, and K. E. Lewis, "Efficient global optimization using hybrid genetic algorithms," presented at 9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Atlanta, USA, 2002.

[29] E. Cantú-Paz, "A survey of parallel genetic algorithms," Calculateurs Parallele, Reseaux et Systems Repartis, vol. 10, pp. 141-171, 1998.

[30] D. E. Goldberg, "Using time efficiently: genetic-evolutionary algorithms and the continuation problem," in the Genetic and Evolutionary Computation Conference. Orlando, USA, 1999, pp. 212-219.

[31] D. E. Goldberg, "Foreward," EURASIP Journal on Applied Signal Processing, vol. 8, pp. 731-732, 2003.

[32] N. Krasnogor and J. Smith, "A memetic algorithm with self-adaptive local search: TSP as a case study," in the Genetic and Evolutionary Computation Conference. Las Vegas, USA Morgan Kaufmann, 2000, pp. 987–994.

[33] S. Areibi and A. Vannelli, "Advanced search techniques for circuit partitioning," in Quadratic Assignment and Related Problems, vol. 16, DIMACS series in Discrtete Mathematics and Theoretical Computer Science, P. Pardalos and H. Wolkowicz, Eds., 1994, pp. 77-98.

[34] E. Besnard, N. Cordier-Lallouet, A. Schmitz, O. Kural, and H. P. Chen, "Design/optimization with advanced simulated annealing," American Insitute of Aeronautic and Astronautics 1999.

[35] K. Liang, X. Yao, and C. Newton, "Combining landscape approximation and local search in global optimization," in the Congress on Evolutionary Computation, vol. 2. Washington DC, USA: IEEE Press, 1999, pp. 1514-1520.

[36] J. Yen, J. C. Liao, B. Lee, and D. Randolph, "A Hybrid approach to modeling metabolic systems using genetic algorithms and simplex method," IEEE Transactions on Systems, Man, and Cybernetics, vol. 28, pp. 173-191, 1998.

[37] M. Chen and Q. Lu, "A hybrid model based on genetic algorithm and ant colony algorithm," Journal of Information & Computational Science, vol. 2, pp. 647-653, 2005.

[38] G. Harik, E. Cantu-Paz, D. E. Goldberg, and B. l. Miller, "The gambler's ruin problem, genetic algorithms, and the sizing of populations," Evolutionary Computation, vol. 7, pp. 231 - 253, 1999.

[39] F. B. Espinoza, B. Minsker, and D. Goldberg, "Performance evaluation and population size reduction for self adaptive hybrid genetic algorithm (SAHGA)," in the Genetic and Evolutionary Computation Conference, vol. 2723, Lecture Notes in Computer Science San Francisco, USA: Springer, 2003, pp. 922-933.

[40] T. El-Mihoub, A. Hopgood, L. Nolle, and A. Battersby, "Performance of hybrid genetic algorithms incorporating local search," in 18th European Simulation Multiconference (ESM2004), G. Horton, Ed. Magdeburg, Germany, 2004, pp. 154-160.

[41] D. E. Goldberg and R. Lingle, "Alleles, loci, and the traveling salesman problem," in the International Conference on Genetic Algorithms and their Applications. Hillsdale, USA: Lawrence Erlbaum, 1985, pp. 154-159.

[42] J. J. Grefenstette, R. Gopal, B. Rosmaita, and D. van Gucht, "Genetic algorithms for the traveling salesman problem," in the First International Conference on Genetic Algorithms and Their Applications, J. J. Grefenstette, Ed. Pittsburgh, USA: Lawrence Erlbaum, 1985, pp. 160-165.

[43] Y. Davidor, Genetic Algorithms and Robotics: A Heuristic Strategy for Optimization: World Scientific Publishing, 1991.

[44] E. K. Burke, D. G. Elliman, and R. F. Weare, "A hybrid genetic algorithm for highly constrained timetabling problems," in the sixth International Conference on Genetic Algorithms, L. J. Eshelman, Ed. Pittsburgh, USA Morgan Kaufmann 1995, pp. 605-610.

[45] B. Freisleben and P. Merz, "New genetic local search operators for the travaling salesman problem," in the Fourth Conference on Parallel Problem Solving from Nature vol. 1141, Lectures Notes in Computer Science, H.-M. Voigt, W. Ebeling, I. Rechenberg, and H.-P. Schwefel, Eds. Berlin, Germany: Springer-Verlag, 1996, pp. 890–899.

[46] S. Lin and B. Kernighan, "An effective heuristic algorithm for the traveling salesman problem," Operations Research, vol. 21, pp. 498-516, 1973.

[47] I. M. Oliver, D. J. Smith, and J. R. C. Holland, "A study of permutation crossover operators on the traveling salesman problem," in the Second International Conference on Genetic Algorithms on Genetic algorithms and their application. Hillsdale, USA, 1987, pp. 224 - 230.

[48] A. Homaifar, S. Guan, and G. E. Liepins, "Schema analysis of the traveling salesman problem using genetic algorithms," Complex Systems, vol. 6, pp. 533-552 1992.

[49] J. Wroblewski, "Theoretical foundations of order-based genetic algorithms," Fundamenta Informaticae, pp. 423–430, 1996.

[50] D. Whitley, T. Starkweather, and D. A. Fuquay, "Scheduling problems and traveling salesman: the genetic edge recombination operator," in the Third International Conference on Genetic Algorithms. Fairfax, USA, 1989, pp. 133 - 140.

[51] P. Jog, J. Y. Suh, and D. Van Gucht, "Parallel genetic algorithms applied to the traveling salesman problem," SIAM Journal of Optimization, vol. 1, pp. 515-529, 1991.

[52] G. Magyar, M. Johnsson, and O. Nevalainen, "An adaptive hybrid genetic algorithm for the three-matching problem," IEEE Transaction on Evolutionary Computation, vol. 4, pp. 135-146, 2000.

[53] B. Burdsall and C. Giraud-Carrier, "Evolving fuzzy prototypes for efficient data clustering," in Second International ICSC Symposium on Fuzzy Logic and Applications. Zurich, Switzerland, 1997, pp. 217-223.

[54] T. Ibaraki, "Combinations with other optimization methods," in Handbook of Evolutionary Computation, T. Back, D. B. Fogel, and Z. Michalewicz, Eds.: IOP Publishing and Oxford University Press, 1997, pp. D3:1.

[55] D. Orvosh and L. Davis, "Shall we repair? genetic algorithms, combinatorial optimization, and feasibility constraints," in the Fifth International Conference on Genetic Algorithms. Urbana-Champaign, USA: Morgan Kaufmann, 1993, pp. 650.

[56] J. Abela, D. Abramson, M. Krishnamoorthy, A. D. Selva, and G. Mills, "Computing optimal schedules for landing aircraft," in the 12th Conference of the Australian Society for Operations Research. Adelaide, 1993, pp. 71-90.

[57] A. Konak and A. E. Smith, "A hybrid genetic algorithm approach for backbone design of communication networks," in the 1999 Congress on Evolutionary Computation. Washington D.C, USA: IEEE, 1999, pp. 1817-1823.

[58] S. Areibi and Z. Yang, "Effective memetic algorithms for VLSI design = genetic algorithms + local search + multi-level clustering," Evolutionary Computation, vol. 12, pp. 327 -353 2004.

[59] Y. Jin, "A comprehensive survey of fitness approximation in evolutionary computation," Soft Computing, vol. 9, pp. 3-12, 2005.

[60] S. Lawrence, A. C. Tsoi, and A. D. Bäck, "Function approximation with neural networks and local methods: bias, variance and smoothness," in Australian Conference on Neural Networks. Canberra, 1996, pp. 16–21.

[61] L. Willmes, T. Bäck, Y. Jin, and B. Sendhoff., "Comparing neural networks and kriging for fitness approximation in evolutionary optimization," in IEEE Congress on Evolutionary Computation. Canberra, Australia, 2003, pp. 663-670.

[62] Y. Jin and B. Sendhoff, "Reducing fitness evaluations using clustering techniques and neural network ensembles," in Genetic and Evolutionary Computation Conference (GECCO 2004), vol. 3102 Lecture Notes in Computer Science. Seattle, USA: Springer, 2004, pp. 688-699.

[63] C. A. Ankenbrandt, B. Buckles, F. E. Petry, and M. Lybanon, "Ocean feature recognition using genetic algorithms with fuzzy fitness functions," in the Third Annual Workshop on Space Operations, Automation and Robotics. Houston, USA, 1989, pp. 679-685.

[64] R. Pearce and P. H. Cowley, "Use of fuzzy logic to describe constraints derived from engineering judgment in genetic algorithms," IEEE Transactions on Industrial Electronics, vol. 43, pp. 535-540, 1996.

[65] H.-P. Schwefel, "Advantages (and disadvantages) of evolutionary computation over other approaches," in Handbook of Evolutionary Computation, T. Bäck, D. B. Fogel, and Z. Michalewicz, Eds.: IOP Publishing and Oxford University Press, 1997, pp. A1.3.

[66] M. Pelikan, D. E. Goldberg, and F. Lobo, "A survey of optimization by building and using probabilistic models," IlliGAL 1999.

[67] S. Baluja, "Population-based incremental learning: a method for integrating genetic search based function optimization and competitive learning," Carnegie Mellon University 1994.

[68] M. Pelikan, D. E. Goldberg, and E. Cantu-Paz, "BOA: the Bayesian optimization algorithm," in the Genetic and Evolutionary Computation Conference. Orlando, USA: Morgan Kaufmann, 1999, pp. 525-532.

[69] S. Tsutsui, M. Pelikan, and A. Ghosh, "Performance of aggregation phermone system on unimodal and multimodal problems," in The 2005 IEEE Congress on Evolutionary Computation, vol. 1. Edinburgh, UK: IEEE, 2005, pp. 880-887.

[70] L. T. Leng, "Guided genetic algorithm," Doctoral Dissertation. University of Essex, 1999.

[71] E. P. Tsang and C. Voudouris, "Fast local search and guided local search and their application to British telecom's workforce scheduling problem," In Operations Research Letters, vol. 20, pp. 119-127, 1997.

[72] A. Hedar and M. Fukushima, "Simplex coding genetic algorithm for the global optimization of nonlinear functions," in Multi-Objective Programming and Goal Programming, Advances in Soft Computing, T. Tanino, T. Tanaka, and M. Inuiguchi, Eds.: Springer-Verlag, 2003, pp. 135-140.

[73] K.-H. Han and J.-H. Kim, "Quantum-inspired evolutionary algorithm for a class of combinatorial optimization," IEEE Transactions On Evolutionary Computation, vol. 6, pp. 580- 593, 2002.

[74] K.-H. Han and J.-H. Kim, "Quantum-inspired evolutionary algorithm with a new termination criterion, H_ϵ gate, and two-phase scheme," IEEE Transactions on Evolutionary Computation, vol. 8, pp. 156-169, 2004.

[75] H. Talbi, A. Draa, and M. Batouche, "A new quantum-inspired genetic algorithm for solving the travelling salesman problem," in 14th International Conference on Computer Theory and Applications. Alexandria, Egypt 2004.

[76] K. C. Tan, Y. Li, D. J. Murray-Smith, and K. C. Sharman, "System identification and linearisation using genetic algorithms with simulated annealing," in First IEE/IEEE Int. Conf. on GA in Eng. Syst.: Innovations and Appl. Sheffield, UK, 1995, pp. 164-69.

[77] T. Yamada and R. Nakano, "A genetic algorithm with multi-step crossover for job-shop scheduling problems," in First IEE/IEEE International Conference on Genetic ALgorithms in Engineering Systems Innovations and Applications (GALESIA '95): Sheffield, UK, 1995, pp. 146–151.

[78] T. P. Riopka and P. Bock, "Intelligent recombination using individual learning in a collective learning genetic algorithm," in the Genetic and Evolutionary Computation Conference (GECCO-2000), D. Whitley, D. Goldberg, E. Cantu-Paz, L. Spector, I. Parmee, and H.-G. Beyer, Eds. Las Vegas, USA: Morgan Kaufmann, 2000, pp. 104-111.

[79] F. Herrera and M. Lozano, "Heuristic crossovers for real-coded genetic algorithms based on fuzzy connectives," in the 4th International Conference on Parallel Problem Solving from Nature, vol. 1141, Lecture Notes In Computer Science. Berlin, Germany: Springer-Verlag 1996, pp. 336 - 345.

[80] J. N. Richter and D. Peak, "Fuzzy evolutionary cellular automata," in International Conference on Artificial Neural Networks in Engineering, vol. 12. Saint Louis, USA, 2002, pp. 185-191.

[81] F. Herrera and M. Lozano, "Adaptive genetic operators based on co-evolution with fuzzy behaviors," IEEE Transactions on Evolutionary Computation, vol. 5, pp. 149-165, 2001.

[82] R. K. Belew, J. McInerney, and N. N. Schraudolph, "Evolving networks: using the genetic algorithm with connectionist learning," in Artificial Life II. New York, USA: Addison-Wesley, 1991, pp. 511-547.

[83] H. Liang, Z. Lin, and R. W. McCallum, "Application of combined genetic algorithms with cascade correlation to diagnosis of delayed gastric emptying from electrogastrograms," Medical Engineering & Physics, vol. 22, pp. 229–234, 2000.

[84] D. J. Montana, "Neural network weight selection using genetic algorithms," in Intelligent Hybrid Systems: John Wiley & Sons, 1995, pp. 85-104.

[85] P. Arena, R. Caponetto, I. Fortuna, and M. G. Xibilia, "MLP optimal topology via genetic algorithms," in the International Conference on Artificial Neural Nets and Genetic Algorithms, A. Dobnikar, N. Steele, D. Pearson, and R. F. Albrecht, Eds. Portoroz, Slovenia: Springer-Verlag, 1993, pp. 670-674.

[86] N. Chaiyaratana and A. M. Zalzala, "Hybridisation of neural networks and a genetic algorithm for friction compensation," in The 2000 Congress on Evolutionary Computation, vol. 1. San Diego, USA, 2000, pp. 22-29.

[87] J. R. Koza and J. P. Rice, "Genetic generation of both the weights and architecture for a neural network," in Joint Conference on Neural Networks, vol. 2. Seattle, USA, 1991, pp. 397-404.

[88] G. F. Miller, P. M. Todd, and S. U. Hegde, "Designing neural networks using genetic algorithms," in the Third International Conference on Genetic Algorithms, J. D. Schaffer, Ed. San Mateo, USA: Morgan Kaufmann, 1989, pp. 379-384.

[89] D. Chalmers, "The evolution of learning: an experiment in genetic connectionism," in Connectionist Models,1990 Summer School, D. Touretzky, J. Elman, T. Sejnowski, and G. Hinton, Eds. San Diego, USA: Morgan Kaufmann, 1990, pp. 81–90.

[90] J. Fontanari and R. Meir, "Evolving a learning algorithm for the binary perceptron," Network, vol. 2, pp. 353–359, 1991.

[91] X. Yao, "Evolving artificial neural networks," Proceedings of the IEEE, vol. 87, pp. 1423-1447, 1999.

[92] C. L. Karr, "Design of an adaptive fuzzy logic controller using a genetic algorithm," in the Fourth International Conference on Genetic Algorithms. San Diego, USA: Morgan Kaufmann, 1991, pp. 450-457.

[93] C. L. Karr and E. J. Gentry, "Fuzzy control of pH using genetic algorithms," IEEE Transaction on Fuzzy Systems, vol. 1, pp. 46-53, 1993.
[94] M. Valenzuela-Rendon, "The fuzzy classifier system: motivations and first results," in the International Workshop Parallel Problem, vol. 496, Lecture Notes in Computer Science. Dortmund, Germany: Springer, 1991, pp. 338-342.
[95] T. White, B. Pagurek, and F. Oppacher, " ASGA: improving the ant system by integration with genetic algorithms," in the third Conference on Genetic Programming (GP/SGA'98). Madison, USA, 1998, pp. 610-617.
[96] H. M. Botee and E. Bonabeau, "Evolving ant colony optimization," Advanced Complex Systems, vol. 1, pp. 149-159, 1998.
[97] M. L. Pilat and T. White, "Using genetic algorithms to optimize ACS-TSP," in the Third International Workshop on Ant Algorithms, vol. Lecture Notes In Computer Science 2463. Berlin, Germany: Springer-Verlag, 2002, pp. 282 - 287.
[98] G. Hinton and S. J. Nowlan, "How learning can guide evolution," Complex Systems, vol. 1, pp. 495-502., 1987.
[99] C. Houck, J. Joines, M. Kay, and J. Wilson, "Empirical investigation of the benefits of partial Lamarckianism," Evolutionary Computation, vol. 5, pp. 31- 60, 1997.
[100]J. A. Joines, M. G. Kay, R. King, and C. Culbreth, "A hybrid genetic algorithm for manufacturing cell design," Journal of the Chinese Institute of Industrial Engineers, vol. 17, pp. 549-564, 2000.
[101]C. Sung-Soon and M. Byung-Ro, "A graph-based Lamarckian-Baldwinian hybrid for the sorting network problem" IEEE Transactions on Evolutionary Computation, vol. 9, pp. 105- 114, 2005.
[102]D. Whitley, S. Gordon, and K. Mathias, "Lamarckian Evolution, the Baldwin effect and function optimization," in Parallel Problem Solving from Nature - PPSN III vol. 866, Lecture Notes in Computer Science, Y. Davidor, H.-P. Schwefel, and R. Manner, Eds. Jerusalem: Springer-Verlag, 1994, pp. 6-15.
[103]P. Turney, "Myths and legends of the Baldwin effect," in Workshop on Evolutionary Computation and Machine Learning at the 13th International Conference on Machine Learning. Bari, Italy, 1996, pp. 135-142.
[104]P. Turney, D. Whitley, and R. Anderson, "Evolution, learning, and instinct: 100 years of the Baldwin effect," Evolutionary Computation, vol. 4, pp. iv-viii, 1996.
[105]G. Mayley, "Landscapes, learning costs and genetic assimilation," Evolutionary Computation, vol. 4, pp. 213 - 234, 1996.
[106]T. El-Mihoub, A. Hopgood, L. Nolle, and A. Battersby, "A self-adaptive Baldwinian search in hybrid genetic algorithms," in the 6th Fuzzy Days International Conference on Computational Intelligence. Dortmund, Germany: Springer, 2006, to be published.
[107]J. A. Joines and M. G. Kay, "Hybrid genetic algorithms and random linkage," in the 2002 Congress on Evolutionary Computation. Honolulu, USA: IEEE, 2002, pp. 1733-1738.
[108]W. E. Hart, T. E. Kammeyer, and R. K. Belew, "The role of development in genetic algorithms," in the Third Workshop on Foundations of Genetic Algorithms. San Fransico, USA, 1995, pp. 315-332.
[109]Z. Michalewicz and G. Nazhiyath, "Genocop III: a co-evolutionary algorithm for numerical optimization problems with nonlinear constraints," in 2nd IEEE International Conference on Evolutionary Computation, vol. 2. Perth, Australia IEEE, 1995, pp. 647-651.
[110]J. Bala, K. A. D. Jong, J. Huang, H. Vafaie, and H. Wechsler, "Using learning to facilitate the evolution of features for recognizing visual concepts," Evolutionary Computation, vol. 4, pp. 297–311, 1996.
[111]K. W. Ku and M. W. Mak, "Exploring the effects of Lamarckian and Baldwinian learning in evolving neural networks," in International Conference on Evolutionary Computation. Indianapolis, USA, 1997, pp. 617-622.
[112]T. Sasaki and M. Tokoro, "Adaptation toward changing environments: why Darwinian in nature?," in Fourth European Conference on Artificial Life, , Complex Adaptive Systems Series P. Husbands and I. Harvey, Eds. Brighton, UK: MIT press, 1997, pp. 145-153.
[113]F. B. Espinoza, B. Minsker, and D. Goldberg, "A self adaptive hybrid genetic algorithm," in the Genetic and Evolutionary Computation Conference (GECCO 2001). San Francisco, USA: Morgan Kaufmann Publishers, 2001, pp. 759.
[114]H. Ishibuchi, S. Kaige, and K. Narukawa, "Comparison between Lamarckian and Baldwinian repair on multiobjective 0/1 knapsack problems," in Evolutionary Multi-Criterion Optimization, Carlos A. Coello Coello, A. H. Aguirre, and E. Zitzler, Eds. Guanajuato, Mexico, 2005, pp. 370-385.
[115]D. E. Goldberg and S. Voessner, "Optimizing global-local search hybrids," in the Genetic and Evolutionary Computation Conference (GECCO 1999). Orlando, USA: Morgan Kaufmann, 1999, pp. 222-228.
[116]C. D. Rosin, R. S. Halliday, W. E. Hart, and R. K. Belew, "A comparison of global and local search methods in drug docking," in the Seventh International Conference on Genetic Algorithms, T. Bäck, Ed. Michigan, USA: Morgan Kaufmann, 1997, pp. 221-228.
[117]M. Land, "Evolutionary algorithms with local search for combinatorial optimization," Doctoral Dissertation. San Diego: University of California 1998.
[118]K. Mathias and D. Whitley, "Genetic operators, the fitness landscape and the traveling salesman problem," in Parallel Problem Solving from Nature-PPSN 2. Brussels, Belguim: North Holland-Elsevier, 1992, pp. 219-228.
[119]K. Mathias, L. Whitley, C. Stock, and T. Kusuma, "Staged hybrid genetic search for seismic data imaging," in International Conference on Evolutionary Computation. Orlando, USA, 1994, pp. 356-361.
[120]M. Syrjakow and H. Szczerbicka, "Combination of direct global and local optimization methods," in IEEE Conference on Evolutionary Computation. Perth, Western Australia: IEEE, 1995, pp. 326-333.
[121]A. L. Tuson and P. Ross, "Cost based operator rate adaptation: an investigation," in the Fourth International Conference on Parallel Problem Solving From Nature (PPSN IV), Lecture Notes in Computer Science. Berlin, Germany: Springer Verlag, 1996, pp. 461-469.
[122]W. E. Hart, C. R. Rosin, R. K. Belew, and G. M. Morris, "Improved evolutionary hybrids for flexible ligand docking in AutoDock," in Optimization in Computational Chemistry and Molecular Biology, C. A. Floudas and P. M. Pardalos, Eds.: Springer 2000, pp. 209-230.
[123]H. Ishibuchi, T. Yoshida, and T. Murata, "Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling," IEEE Transactions on Evolutionary Computation, , vol. 7, pp. 204- 223, 2003.
[124]N. J. Radcliffe and P. D. Surry, "Formal memetic algorithms," in Evolutionary Computing: AISB Workshop. Brighton, UK: Springer-Verlag, 1994, pp. 1-16.
[125]A. O. Griewank, "Generalized descent for global optimization," Journal of Optimization Theory and Applications, vol. 34, pp. 11-39, 1981.
[126]A. Törn and A. Zilinskas, "Global optimization," in Lecture Notes in Computer Science, vol. 350: Springer-Verlag, 1989.
[127]W. E. Hart and R. K. Belew, "Optimization with genetic algorithm hybrids that use local search," in Adaptive individuals in evolving populations: Models and algorithms, vol. 26, R. Belew and M. Mitchell, Eds.: Addison-Wesley, 1996, pp. 483-496.
[128]M. Land, J. J. SIDorowich, and R. K. Belew, "Using genetic algorithms with local search for thin film metrology," in the Seventh International Conference on Genetic Algorithms. East Lansing, USA: Morgan Kaufmann, 1997, pp. 537-544.
[129]F. Glover, "Tabu search- part I," ORSA Journal on Computing, vol. 1, pp. 190-260, 1989.
[130]A. Martinez-Estudillo, C. Hervas-Martnez, F. Martnez-Estudillo, and N. Garca-Pedrajas, "Hybrid method based on clustering for evolutionary algorithms with local search," IEEE Transactions on Systems, Man and Cybernetics, 2004.
[131]F. Espinoza, B. S. Minsker, and D. Goldberg, "Local search issues for the appliction of a self-adaptive hybrid genetic algorithm in groundwater remediation design," in American Society of Civil Engineers (ASCE) Environmental & Water Resources Institute (EWRI) World Water & Environmental Resources Congress 2003 & Related Symposia. Philadelphia, USA, 2003.
[132]M. Lozano, F. Herrera, N. Krasnogor, and D. Molina, "Real-coded memetic algorithms with crossover hill-climbing," Evolutionary computation, vol. 12, pp. 273 - 302 2004.
[133]F. T. Lin, C. Y. Kao, and C. C. Hsu., " Incorporating genetic algorithms into simulated annealing," in the Fourth International Symposium on Artificial Intelligence. Cancun, Mexico, 1991, pp. 290-297.
[134]N. Krasnogor and J. Simth, "Emergence of profitable search strategies based on a simple inheritance mechanism," in the Genetic and Evolutionary Computation Conference. San Francisco, USA: Morgan Kaufmann, 2001, pp. 432-439.
[135]D. Goldberg, "The race, the hurdle, and the sweet spot: Lessons from genetic algorithms for the automation of design innovation and creativity," in Evolutionary Design by Computers: Morgan Kaufmann, 1999, pp. 105–118.

[136]Y. Chen and D. Goldberg, "Convergence time for the linkage learning genetic algorithms," Evolutionary computation, vol. 13, pp. 279-302, 2005.

Tarek A. El-Mihoub graduated with a BSc in computer engineering from Al-Fateh Uinversity, Tripoli, Libya in 1993 and obtained his MSc in engineering multimedia from Nottingham Trent University in UK by the end of 2002.

He is currently a PhD Student at Nottingham Trent University. He worked as a Teaching Assistant at Al-Fatah University in Libya and as a Manager of computer department of the Libyan environment general authority. His current research is in the field of optimization, genetic algorithms, and artificial intelligence.

Adrian A. Hopgood graduated with a BSc (Hons) in physics from the University of Bristol in 1981 and obtained a PhD from the University of Oxford in 1984.

He is professor of Computing and Dean of the School of Computing & Informatics at Nottingham Trent University, UK. He is also a visiting professor at the Open University. His main research interests are in intelligent systems and their practical applications.

Prof. Hopgood is a fellow of the British Computer Society and a committee member for its specialist group on artificial intelligence.

Lars Nolle graduated from the University of Applied Science and Arts in Hanover in 1995 with a degree in Computer Science and Electronics. After receiving his PhD in Applied Computational Intelligence from The Open University, he worked as a System Engineer for EDS.

He returned to The Open University as a Research Fellow in 2000. He joined The Nottingham Trent University as a Senior Lecturer in Computing in February 2002. His research interests include: applied computational intelligence, distributed systems, expert systems, optimization and control of technical processes.

Alan Battersby obtained an MSc in Computer Science from Hatfield Polytechnic, UK in 1977.

Prior to joining the School of Computing and Informatics at Nottingham Trent University, UK, he was a Computing Development Officer for Bedfordshire Education Authority. His research interests include: Fuzzy Logic applied to Robotics, wavelets, compression and the Internet.

Self-adaptive Baldwinian Search in Hybrid Genetic Algorithms

T. El-Mihoub, A. A. Hopgood, L. Nolle and A. Battersby

School of Computing and Informatics, Nottingham Trent University
Clifton Lane, Nottingham, NG11 8NS, UK
tarek.elmihoub,adrian.hopgood,lars.Nolle,alan.battersby@ntu.ac.uk

Summary. The problem of proper utilization of the search time to adapt a hybrid to a given problem can be viewed as a problem of finding optimal control parameter settings. The algorithm's time utilization can be optimized through adapting the local search duration. Evolving this control parameter via genetic operations is one possible way to achieve this adaptation. However, the hindering effect can obstruct the self-adaptive ability of the Baldwinian search. Local search methods with narrow steps and the use of the local search duration to discriminate between solutions can help to alleviate this problem.

Key words: Self-adaptaion, Hybrid genetic algorithms, Baldwinian search, Hindering effect

1 Introduction

A genetic algorithm is usually combined with a domain-specific method to solve a real-world problem [8] . The success of such a hybrid algorithm in solving a given problem efficiently depends on its success in achieving a balance between exploration and exploitation [4, 8, 3]. Among the factors that affect this balance is the duration of local search [5], which is defined as the number of the consecutive local search iterations that is performed on a solution before terminating a local search procedure. This control parameter can be used to adapt the hybrid on-line to a specific problem.
The interactions between local search duration, learning strategy, fitness topology, and other genetic components have a great impact on search time utilization [4, 5]. The idea of evolutionary self-adaptation [6] can be applied to adapt the local search duration in order to optimise the performance of a hybrid on a particular problem without the need for external control.
The impact of the hindering effect [6] on obscuring genetic differences can obstruct the Baldwinian[7] search's self-adapting ability to a given problem. The genotypes cannot be effectively discriminated according to their fitness

without considering the learning cost and hence the evolution of effective solutions can be hindered.
The ability of genetic search to find favourable parameter settings for pure genetic algorithms has been proven [6]. However, its ability within a hybrid to self-adapt the control parameters, especially those related to incorporating a local method, may require further investigation. In this paper, we analyse the influence on the behaviour of the Baldwinian hybrid of simultaneously exploring both the problem search space and the control space of local search duration. This analysis can help to gain some insight into the factors that may affect the search performance in order to find ways to improve it.

2 Evolutionary Self-adaptation and Duration of Local Search

In evolutionary self-adaptive algorithms, the fitness of the individual associated with a specific control parameter value is used as feedback to assess the suitability of the control parameter values for solving a given problem. The link between the duration-of-local-search control parameter and the individual's fitness depends on the fitness function topology, the details of the local search method and the genetic algorithm's setup. By allowing the duration of the local search to evolve by means of genetic operations, the link between favourable duration of the local search and the fitness can be exploited. Genetic operations can adaptively control the duration of the local search method to optimise the individuals fitnesses. In this way, this link can be defined, which is essential for the adaptation of control parameters [11].
However, it may be difficult to define this link when the genetic algorithm is combined with Baldwinian search. The acquired fitness is the sum of the improvements introduced by applying a local search method for the encoded duration and the innate fitness. The hindering effect can direct the search towards individuals with long durations and a small innate fitness. The search process, in this case, is degraded from optimising the fitness function to optimising a single control parameter. The possibility of leading the search in this direction increases as the dimension of the fitness function increases, since it may be easier for the algorithm to optimise a single control parameter than to optimise a large number of variables. It can also waste its resources as it can direct the individuals towards performing useless local search iterations. The use of the acquired fitness as a metric to assess the quality of solutions in the Baldwinian search can produce an algorithm with poor performance.
The use of a local search method, which takes narrow steps in the search space while restricting the values of the duration of local search to very small numbers, can help to combat the hindering effect problem. In this way, the problems consequences on the ability of the algorithm to define a link between this control parameter and the fitness in the direction of optimising solutions quality can be alleviated.

However, the ultimate solution for the hindering effect problem is to rely on innate fitness to decide between solutions of equal acquired fitness values. Since the number of local iterations, which is a good indication of the cost of learning, is already encoded into the individual, it can be used together with the acquired fitness to direct the search towards solutions of high quality.
It may be beneficial to allow the local search method to cooperate with the global genetic algorithm to explore the search space in the early stages of the search by allowing wide local steps. However, as the Baldwinian search reaches the fitness-convergence-state, taking narrow local steps can be more helpful. By adapting the local step size according to the standard deviation of the population fitness, the search performance may be improved.

3 Experiments

A set of experiments was conducted to gain some insight into the evolutionary self-adaptive behaviour of the Baldwinian search using three different hybrids. Hybrid-A, which uses a local search method with a predefined maximum local step size and discriminates between solutions based on the acquired fitness only, was used to study the effect of local search step size on the performance. Hybrid-B, which is identical to Hybrid-A except that it uses local search iterations to discriminate between solutions of an equal acquired fitness, was used to investigate the effectiveness of using the local search duration to discriminate between solutions. The possibility of improving the hybrid performance by employing an adaptive local step was examined through Hybrid-C, which uses an adaptive local step size and utilizes the local search duration to discriminate between effective solutions.
In these hybrids, the number of local search iterations that should be performed by an individual was encoded into its chromosome. At each iteration, the local search method tries to find the smallest possible step in the allowed range of a randomly selected variable space that improves the fitness. Starting from the least significant bit of a randomly chosen variable and moving towards its most significant bit, the local search method keeps flipping the bits until an improvement in the fitness produced or a specified number of bits are flipped. In the case of no improvement in the fitness, the process is repeated for another randomly chosen variable until an improvement is produced. By controlling the maximum number of bits that can be scanned for fitness improvement of each variable before randomly selecting another variable, the algorithm controls the size of the local search step.
The generalized Ellipsoidal [2], Ackley [1], Schwefel [10], Rastrigin [10], and Griewank [10] functions were selected as a test suite. The hybrids used the simple elitist genetic algorithm with binary tournament selection, two-point crossover, and simple mutation. The values of the duration of local search parameter were restricted to very small values in the range 03. For all experiments, the rate of crossover and mutation were set to 0.7 and (popula-

tion size)$^{-1}$, respectively. The population sizes for the 2- and 10-dimensional functions were set to 50 and 100, respectively. Each variable was represented by a 10-bit string. The stopping criterion for all experiments was a maximum number of function evaluations. The number of bits that were exposed to modification was limited to a specific percentage of the length of the variables string. Each experiment was repeated 100 times.

4 Discussion

The results of the first two hybrids clearly show that as the size of the local search step decreases, the ability of the evolutionary self-adaptive Baldwinian hybrid to find a global optimum increases. This is depicted in Fig. 1 for the 10-dimensional Ellipsoidal, Ackley, and Schwefel functions. The algorithms were unable to find the global optimum for the 10-dimensional Griewank and Rastrigin functions. However, the curves of the best fitness of these functions show a similar trend. The curves of the percentage of experiments that found a global optimum of the 10-dimensional functions, as expected, have a steeper slope than the 2-dimensional functions. As shown in Fig. 1, Hybrid-B outperformed Hybrid-A in terms of the percentage that converged.

The experiments also show that using small local steps improves the speed of

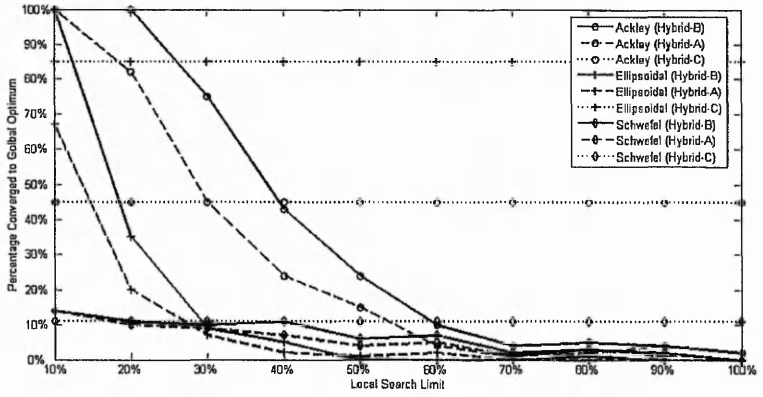

Fig. 1. The effect on convergence ability

the algorithms in finding the global optimum (Fig. 2). Fig. 2 also illustrates that Hybrid-B significantly outperforms Hybrid-A in terms of the search speed of the 10-dimensional Ackley and Schwefel functions.

Hybrid-C produced a near optimal performance in terms of the percentages that converged and an optimal performance in terms of convergence speed for the 10-dimensional Schwefel and Ellipsoidal functions, as illustrated by the dotted lines in Fig. 1 and Fig. 2. Hybrid-C also improved the best fitness and the search speed of the Rastrigin and Griewank functions. However, the algorithm produced a poor performance for the Ackley function.

5 Conclusions

The hindering effect can obstruct the ability of Baldwinian search to self-adapt the duration-of-local-search control parameter. The possibility of obstructing this ability increases as the dimension of the fitness function increases. The results presented in this paper also show that the use of a local search method with narrow steps in the search space can help to alleviate this problem and hence improve the performance of the Baldwinian search in terms of solution quality and convergence speed. The performance of the Baldwinian search can be further improved when the local search duration is used alongside the acquired fitness to discriminate between effective solutions. The use of an adaptive local search step can improve the performance of the Baldwinian search on some of the tested problems.

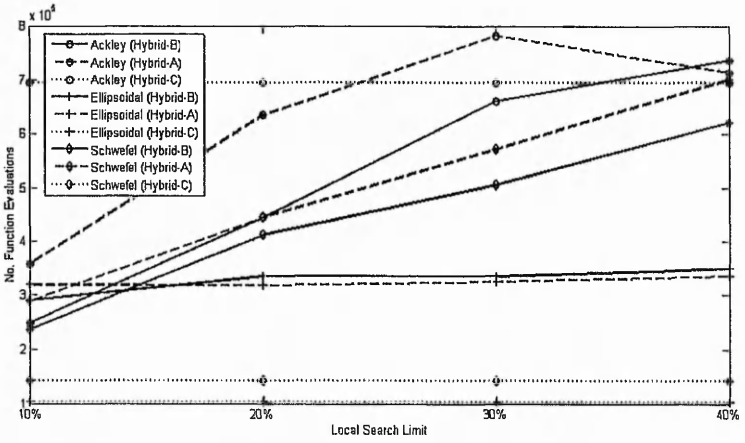

Fig. 2. The effect on convergence speed

References

1. Baeck T, Schwefel H-P (1993) An overview of evolutionary algorithms for parameter optimization. Evolutionary Computation. The MIT press. 1:1–23
2. Deb K, Anand A, Joshi D (2002) A computationally efficient evolutionary algorithm for real-parameter optimization. Evolutionary Computation. The MIT press. 371395
3. Goldberg D, Voessner S (1999) Optimizing global-local search hybrids. In: Banzhaf W, Daida J, Eiben A, Garzon M, Honavar V, Jakiela M, Smith R (Eds.) the genetic and evolutionary computation conference. Morgan Kaufmann. 222–228
4. Hart E (1994) Adaptive global optimization with local search. PhD Thesis, University of California, San Diego
5. Hart E, Rosin C, Belew R, Morris G (1997) Improved evolutionary hybrids for flexible ligand docking in AutoDock. In Intl conf of optimization in computational chemistry and molecular biology. 209–230
6. Hinterding R, Michalewicz Z, Eiben E (1997) Adaptation in evolutionary computation: A survey. In: the 4th IEEE international conference on evolutionary computation. IEEE. 65–69
7. Hinton G, Nowlan J (1989) How learning can guide evolution. Complex Systems. 1:495–502
8. Lobo F, Goldberg D (1997) Decision making in a hybrid genetic algorithm. in: IEEE international conference on evolutionary computation. IEEE Press. 121–125
9. Mayley G (1996) Landscapes, learning costs and genetic assimilation. in: Turney P, Whitley D, Anderson R (eds) Evolution, learning, and instinct: 100 years of the Baldwin effect. special issue of Evolutionary Computation. The MIT press.4: 213–234
10. Muehlenbein H, Schomisch M, Born J (1991) The parallel genetic algorithm as function optimizer. Parallel Computing. 17: 619–632
11. Tuson A, Ross P (1998) Adapting operator settings in genetic algorithms. Evolutionary Computation. The MIT press. 6:161–184

A Self-Adaptive Hybrid Genetic Algorithm for Color Clustering

Tarek El-Mihoub, Lars Nolle, Gerald Schaefer, Tomoharu Nakashima and Adrian Hopgood

Abstract — **Color palettes are inherent to color quantized images and represent the range of possible colors in such images. When converting full true color images to palletized counterparts, the color palette should be chosen so as to minimize the resulting distortion compared to the original. In this paper, we show that in contrast to previous approaches on color quantization, which rely on either heuristics or clustering techniques, a generic optimization algorithm such as a self-adaptive hybrid genetic algorithm can be employed to generate a palette of high quality. Experiments on a set of standard test images using a novel self-adaptive hybrid genetic algorithm show that this approach is capable of outperforming several conventional color quantization algorithms and provide superior image quality.**

I. Introduction

True color images typically use 24 bits per pixel which results in an overall gamut of 2^{24} i.e. more than 16.8 million different colors. While nowadays most images are captured and stored in that format, in certain applications (for example display of images on limited hardware such as mobile devices and for compression and retrieval of images [1]) it is advantageous to limit the range of possible colors to fewer entries whose ensemble are known as a color palette. Color quantization is the process of generating a suitable palette (usually of size between 8 and 256) where suitable is often defined as introducing as little distortion as possible, or equivalently, as maintaining the best possible image quality.

In this paper we apply a self-adaptive hybrid genetic algorithm (SAHGA) as a standard black-box optimization approach to the color quantization problem. The main advantage of black-box optimization algorithms is that they do not require any domain specific knowledge yet are able to provide a near optimal solution. We evaluate the effectiveness of our approach by comparing its performance to the results obtained by several purpose built color quantization algorithms [2-4]. The results obtained show that even without any domain specific knowledge our SAHGA based algorithm is able to outperform standard quantization algorithms and hence to provide palletized images with superior image quality.

The rest of the paper is organized as follows. The next section provides a formal definition of the color quantization problem. Section III provides the background for optimization based on self-adaptive hybrid genetic algorithm. Section IV explains our application of SAHGA, a modified HGA algorithm, to the color quantization problem. Section V presents experimental results based on a set of standard test images while Section VI concludes the paper.

Tarek El-Mihoub, Lars Nolle, Gerald Schaefer, and Adrian Hopgood are with the School of Computing and Informatics, Nottingham Trent University, Nottingham, UK.

Tomoharu Nakashima is with the College of Engineering, Osaka Prefecture University, Osaka, Japan.

II. Color Quantization

Color quantization produces a color palette that contains only a small number of colors (usually between 8 and 256); pixel data are then stored as indices to this palette. Clearly, the choice of colors that make up the palette has a crucial influence on the image quality of the quantized image. Formally, given an image quality metric which assigns $d(I_1(x,y),I_2(x,y))$ as the distance (or difference) between two pixels at location (x,y) in images I_1 and I_2, an n x m original image $O = \{o_i=\{R_i,G_i,B_i\},\ i = 1\ldots n \times m\}$, a palette of size N, $P = \{p_j = \{R_j,G_j,B_j\}, j = 1\ldots N\}$, P is optimal iff

$$\begin{aligned} &\neg\exists\, \overline{P} = \{\overline{p}_k = \{R_k,G_k,B_k\}, k = 1\ldots N\} \\ &\text{so that } D(O,q(O,\overline{P})) < E(O,q(O,P)) \end{aligned} \tag{1}$$

with E (the error between two images) defined as

$$E(I_1,I_2) = \sum_{x=1}^{n}\sum_{y=1}^{m} d(I_1(x,y),I_2(x,y)) \tag{2}$$

and q (the result of the quantization process)

$$q(O,P) = \{q_r = p_s, r = 1\ldots n\cdot m / d(o_r,p_s) < d(o_r,p_t)\ \forall\, t \neq s\} \tag{3}$$

However, the selection of the optimal color palette is known to be an *np*-hard problem [4]. In the image processing literature many different algorithms have been introduced that aim to find a palette that allows for good image quality of the quantized image. In general these can be divided into heuristic techniques such as the popularity algorithm [4] and clustering-based algorithms such as the median cut approach [4].

III. Hybrid Genetic Algorithm

Genetic algorithms [5] and other search methods can be seen as complementary tools that can be brought together to achieve an optimization goal. In these hybrids, a genetic algorithm incorporates one or more methods to improve the performance of the genetic search. There are several ways in which a search or optimization technique can complement the genetic search [6-10].

If a genetic algorithm is combined with a fast converging local search methods [11] the resulting hybrid can often outperform the algorithms [12]. Hybridizing a local search method provides the global genetic search algorithm with

some local knowledge that can guide and may accelerate the search to the global optimum [13]. Figure 1 shows a flowchart of the basic hybrid genetic algorithm. As it can be seen from the figure, after the genetic operators are applied in order to generate a new generation, each of the individuals in the new generation undergo optimization using a local search method.

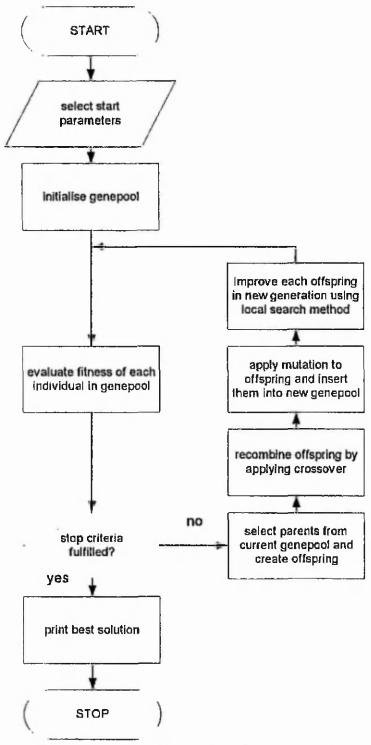

Figure 1 – Flowchart of hybrid Genetic Algorithm.

In this research, a new self-adaptive hybrid genetic algorithm (SAHGA) has been used, which employs a novel local search method, which is described in the following section.

A. Local Search Method used

The new local search algorithm used in this research is a probabilistic method that works on the genotype space by using a sub-group of the current population of solutions to optimize the structures of each solution present in the genepool. In this way, it aims to make use of some of the valuable genetic search information. It also aims to avoid disrupting the genetic schema processing by improving the solution in accordance with the global genetic search.

The modification of the initial solution based on a group of solutions of the genetic population can provide the local search method with a partial global view of the problem at hand. Based on this view, the search method can produce a solution in the context of global view captured by the genetic algorithm. This form of search can minimize any conflict with the global genetic search. The partial global aspect of the search method can be controlled by the group size and the mechanism of selecting the group members. This method is also characterized by its low costs. Its costs are equal to the costs of evolving a solution for a single iteration of the genetic search (i.e. one function evaluation per solution). This can help to minimize the loss of the hybrid's time in the case of any undesirable interference between the two search methods. Figure 2 provides pseudo code for the algorithm.

```
Procedure LocalSearch
Begin
  For each individual s0 in genepool
  Begin
    Select randomly group of individuals
    s1,s2,s3,s4;
    For each gene g in s0
    Begin
      If g = 1
        Set probability p(g) to 1.0;
      Else
        Set probability p(g) to 0.0;
    End
    For each individual s in group
    Begin
      Calculate probability value v as the absolute
      fitness difference between s0 and s,
      normalized by dividing it by the sum
      of differences between each group member
      and s0;
      For each allele a in s
      Begin
        If a equal to corresponding allele in s0
          Set corresponding probability in
          probability vector for s to 0.0;
        Else if a less than allele in s0
          Set probability vector for s to +v;
        Else
          Set probability vector for s to -v;
      End
    End
    Add all probability vectors for s0,s1,s2,s3,s4;
    Generate random vector r;
    For each element in random vector
    Begin
      If probability vector > random element
        Set corresponding gene in new solution to
        1;
      Else
        Set corresponding gene in new solution to
        0;
    End
    Evaluate new solution;
    If fitness of new solution > fitness of s0
      Replace s0 with new solution;
  End LocalSearch
```

Figure 2 – Pseudo code of local search method.

The algorithm assumes that each gene contributes uniformly to the fitness of the solution. Based on this assumption, the search method compares the genetic structure and the fitness of the solution to be improved with the structures and the fitness of a group of solutions selected from the current genetic population. Depending on the differences in both the structure and the fitness between this solution and the group members, the solution structure is modified in the direction of improving its fitness score. The new solution is evaluated and then inserted back into the population if it shows an improvement in its fitness.

B. *The Self-Adaptive Hybrid Genetic Algorithm (SAHGA)*

The success of such a hybrid algorithm in solving a given problem efficiently depends on its success in achieving a balance between exploration and exploitation [12, 13]. Among the factors that affect this balance is the duration of local search [14], which is defined as the number of the consecutive local search iterations that is performed on a solution before terminating a local search procedure. This control parameter can be used to adapt the hybrid on-line to a specific problem.

In the proposed hybrid algorithm, the number of local search iterations is incorporated into the representation of an individual. Through this parameter, the duration of a local search is controlled. It defines the number of local iterations that should be performed by the associated individual. The global genetic algorithm evolves the number of local search iterations parameter while the hybrid is using that control parameter to optimize the fitness function variables. Through adopting the evolutionary self-adaptation metaphor, the algorithm allows the global genetic algorithm to dynamically decide on the individuals that should perform a local search. It also decides on the duration of the local search method through modifying the number of local iterations as it co-operates with the local search to solve a given problem. This can facilitate the adaptation of number of local search iterations control parameter without exogenous control.

In general, the control parameters in the evolutionary self-adaptive algorithm can be adapted either at the individual level (i.e. local level) or at the population level (i.e. global level). In the local adaptation, the control parameter is applied to the associated solution only. In contrast, the control parameter in the global adaptation is tied to the population as a whole and not to a particular solution. The number of local iterations of an individual is computed by taking the average of the number of local iterations of the individuals of the whole population. Local adaptation is used in the proposed algorithm because it is reasonable to assume that different individuals are following different paths through the search space. It is also proven that local adaptation outperforms global adaptation [15].

In the proposed self-adaptive hybrid algorithm, after performing a genetic iteration, the number of local iterations associated with each solution is extracted from the chromosome's structure. Depending on the value of that parameter, a number of local search iterations are performed on that solution. If the value of that parameter is zero, no local search iteration will be performed. Otherwise the specified number of local iterations will be performed consecutively. Using the learning strategy specified by the algorithm, the resulting solution is mapped back to the mating pool.

The maximum value of the number of local iterations was set to three. The reason for selecting this value is the expected benefits of using small durations of local search to fight the hindering effect problem. The algorithm also makes use of the number of local iteration control parameter, which already exists within the chromosome, to discriminate between innate and acquired fitness. In a case of an equal fitness, the algorithm chooses the individual with the smaller value of local search iterations since its acquired fitness is closer to the innate one. This can help to alleviate the consequences of the hindering effect problem [11] associated with the Baldwinian approach.

IV. SAHGA FOR COLOR QUANTIZATION

In this paper we apply the SAHGA algorithm described above as a black box optimization algorithm to the color quantization problem. For color quantization the objective is to minimize the total error introduced through the application of a color palette. The color palette P for an image O, a codebook of N color vectors, should then be chosen so as to minimize the error function E

$$E(P,O) = \frac{1}{\sum_{j=1}^{N} l_j} \sum_{i=1}^{k} \sum_{j=1}^{l_j} \left\| P_i - O_j \right\| + c(P,O) \qquad (4)$$

with

$$c(P,O) = \sum_{i=1}^{k} \delta a_i, \quad a_i = \begin{cases} 1 & \text{if } l_i = 0 \\ 0 & \text{otherwise} \end{cases} \qquad (5)$$

where l_i is the number of pixels O_j represented by color P_i of the palette, $\|.\|$ is the Euclidean distance in RGB space, and δ is a constant (δ=10 in our experiments). The objective function $E(P,O)$ used is hence a combination of the mean Euclidean distance and a penalty function. The penalty function $c(P,O)$ is integrated in order to avoid unused palette colors by adding a constant penalty value to the error for each entry in the codebook that is not used in the resulting picture. As can be seen from Equation 4 the objective function is highly non-linear, i.e. it has a high degree of epistasis [16].

For our color quantization algorithm we employ the SAHGA algorithm with a population size of 100. For all experiments, binary tournament selection, single-point crossover, and simple mutation were used with a crossover probability of 0.6 and a mutation probability of 0.01. Each experiment was repeated 19 times and the codebook with the median error was used for comparison.

V. EXPERIMENTAL RESULTS

In order to evaluate our new method for color quantization, we have taken a set of three standard images commonly used in the color quantization literature, *Lenna*, *Pool*, and *Airplane*, and applied our optimization scheme to generate quantized images with a palette of 16 colors.

To put the results we obtain into context, we have also implemented four popular color quantization algorithms to generate corresponding quantized images with palette size 16. The algorithms we have tested were:

- Popularity algorithm [4]: Following a uniform quantisation to 5 bits per channel the n colours that are represented most often form the colour palette.

- Median cut quantisation [4]: This algorithm starts by computing the box that encompasses all colours present in the image. The box is then split (orthogonal to the colour axis) at the median value into two sub-cubes. The larger remaining sub- cube is then again divided at its median point and this process is repeated until n colour boxes have been found.
- Octree quantisation [3]: The colour space is represented as an octree where the root node corresponds to the whole colour space, the nodes at the next level the eight sub-cubes that are obtained by dividing each colour axis into two equal halves, and so on. In a first pass the sub-tree that represents the colours present in the image is built and in a second pass, starting at the bottom of the tree, nodes are successively merged until a tree of n colours is reached.
- Neuquant [2]: A one-dimensional self-organising Kohonen neural network is trained to generate the colour map. The Kohonen network defines a mapping from the colour values in the image to an index representing the palette entries. The weights of the network are updated based on the image data to ensure an optimal palette with good image quality.

For all algorithms, pixels in the quantised images were assigned to their nearest neighbours in the colour palette to provide the best possible image quality.

The results are listed in Tables 1 and 2, expressed in terms of mean-squared-error (MSE) and peak-signal-to-noise-ratio (PSNR) defined as

$$\mathrm{MSE}(I_1,I_2)=\frac{1}{3mn}[(R_1(i,j)-R_2(i,j))^2+(G_1(i,j)-G_2(i,j))^2+(B_1(i,j)-B_2(i,j))^2] \quad (6)$$

and

$$\mathrm{PSNR}(I_1,I_2)=10\log_{10}\frac{255^2}{\mathrm{MSE}(I_1,I_2)} \quad (7)$$

where $R(i,j)$, $G(i,j)$, and $B(i,j)$ are the red, green, and blue pixel values at location (i,j) and n and m are the dimensions of the images.

As MSE and PSNR are not necessarily the best quality indicators, the results are also provided in terms of S-CIELAB [17]. This is an image quality metric based on uniform colour spaces but it also takes into account the spatial interaction between neighbouring pixels based on a blurring effect derived from psychophysical experiments. S-CIELAB results, expressed in terms of ΔE differences between original and quantised images are provided in Table 3.

	Popalg	Medct	Octree	Neuqu.	SAHGA
Lenna	388.1	271.4	117.0	107.4	**93.4**
Pool	669.9	226.9	74.9	127.2	**60.3**
Airplane	1668.1	240.7	86.5	97.6	**72.8**
all	908.7	246.3	92.8	110.7	**75.5**

Table 1. Quantization results, given in terms of MSE.

	Popalg	Medct	Octree	Neuqu.	SAHGA
Lenna	22.24	23.79	27.45	27.82	**28.43**
Pool	15.91	24.32	28.77	28.24	**30.53**
Airplane	19.87	24.57	29.39	27.08	**29.51**
all	19.34	24.23	28.54	27.71	**29.45**

Table 2. Quantization results, given in terms of PSNR [dB].

	Popalg	Medct	Octree	Neuqu.	SAHGA
Lenna	11.92	21.81	20.54	41.03	**9.31**
Pool	10.34	8.89	10.66	10.10	**7.73**
Airplane	7.20	7.92	9.36	6.66	**4.85**
all	19.34	24.23	28.54	27.71	**29.45**

Table 3. Quantization results, given in terms of S-CIELAB ΔE.

From Tables 1 to 3 we can see our self-adaptive hybrid genetic algorithm approach to colour quantisation obtains clearly the best results for all three images. Overall a mean PSNR (MSE) of 29.45 dB (75.5) is achieved which is significantly better than the 28.54 dB (92.8) and 27.71 dB (110.7) obtained by Octree and Neuquant, the two next best algorithms.

In terms of S-CIELAB, the hybrid algorithm provides image quality that is more than 2 ΔE units lower than the next best approaches. Considering that a difference of 1 ΔE unit is perceptually visible this indeed indicates a significant improvement.

An example of the performance of the different algorithms is provided in Figure 3, which shows the *Pool* image together with the quantization results from all algorithms. Figure 4 provides error images of the quantised images from Figure 1 compared to the original.

It is clear that the popularity algorithm performs poorly on this image and assigns virtually all of the colors in the palette to green and achromatic colors. Median cut is better but still provides fairly poor color reproduction; most of the colors in the quantized image are fairly different from the original. The same holds true for the images produced by Neuquant. Here the most obvious artifact is the absence of an appropriate red color in the color palette. A far better result is achieved by the Octree algorithm, although here also the red is not very accurate and the color of the cue is greenish instead of brown.

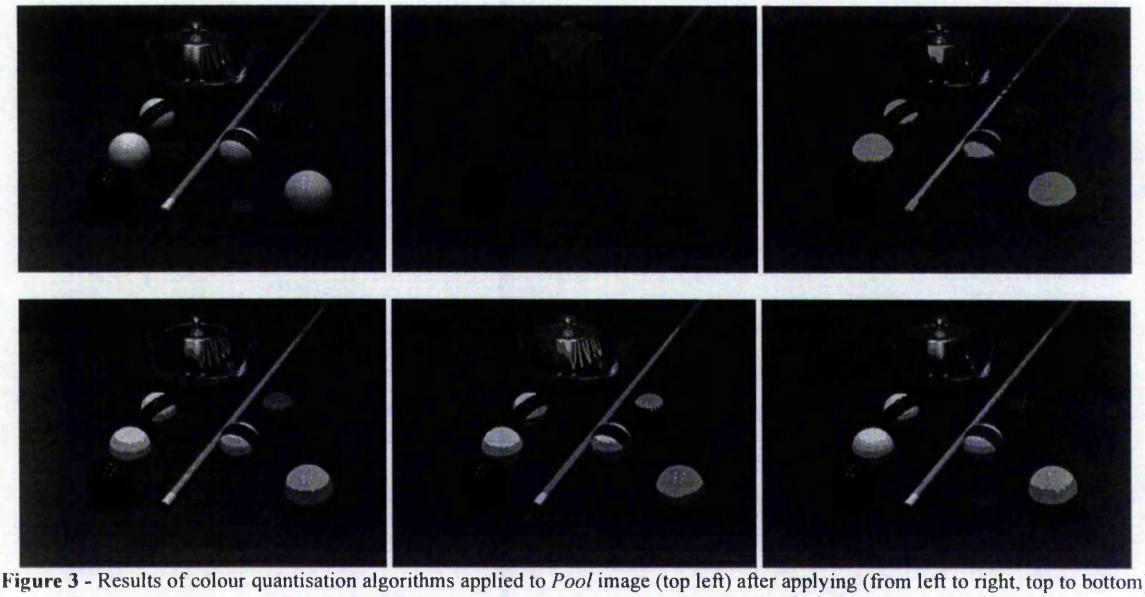

Figure 3 - Results of colour quantisation algorithms applied to *Pool* image (top left) after applying (from left to right, top to bottom) Popularity, Median cut, Octree, Neuquant, and SAHGA algorithms.

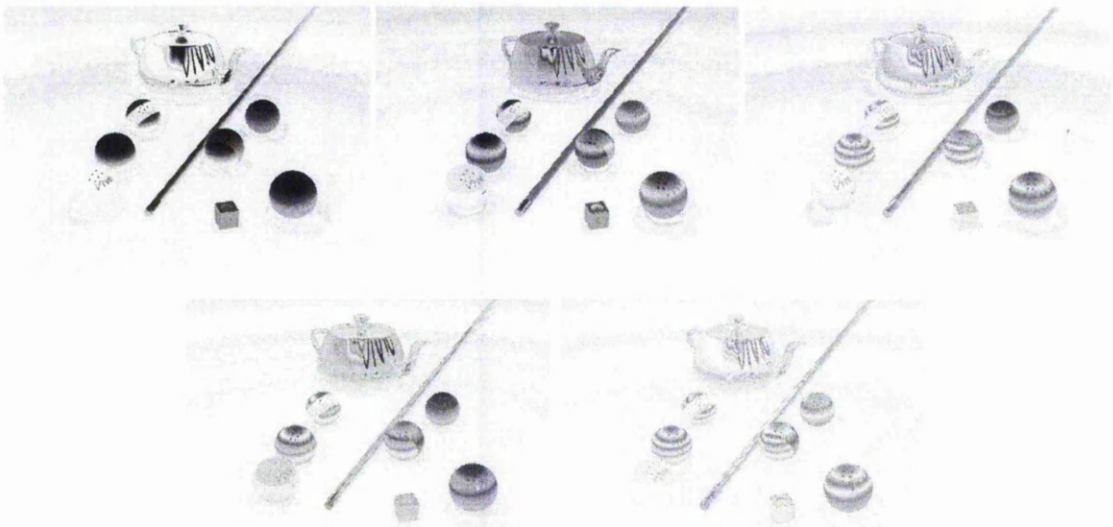

Figure 3 - Error images of the quantised images from Figure 1 for (from left to right, top to bottom) Popularity, Median cut, Octree, Neuquant, and SAHGA algorithms compared to the original.

Clearly the best image quality is maintained by applying our self-adaptive hybrid technique. Although the color palette has only 16 entries all colors of the original image are accurately presented including the red ball and the color of the billiard cue.

VI. Conclusion

In this work we have applied a novel self-adaptive hybrid genetic algorithm as a generic optimization algorithm to the color quantization problem. Experimental results obtained on a set of standard test images have demonstrated that this type of optimization techniques cannot only be effectively employed but is even able to outperform standard purpose built color quantization algorithms.

References

[1] G. Schaefer, G. Qiu, and G. Finlayson, "Retrieval of palettised colour images," in SPIE, vol. 3972 (Storage and Retrieval for Image and Video Databases VIII), 2000, pp. 483-493.

[2] A. H. Dekker, "Kohonen neural networks for optimal colour quantization," Network: Computation in Neural Systems, vol. 5, pp. 351-367, 1994.
[3] M. Gervautz and W. Purgathofer, "A simple Method for Color Quantization: Octree Quantization," in Graphics Gems, A. S. Glassner, Ed. San Diego, CA: Academic Press, 1998.
[4] P. S. Heckbert, "Color image quantization for frame buffer display," ACM Computer Graphics (ACM SIGGRAPH '82 Proceedings), vol. 16, pp. 297-307, 1982.
[5] J. Holland, Adaptation in Natural and Artificial Systems: The University of Michigan Press, 1975.
[6] D. E. Goldberg and R. Lingle, "Alleles, loci, and the traveling salesman problem," in the International Conference on Genetic Algorithms and their Applications. Hillsdale, USA: Lawrence Erlbaum, 1985, pp. 154-159.
[7] Y. Jin, "A comprehensive survey of fitness approximation in evolutionary computation," Soft Computing, vol. 9, pp. 3-12, 2005.
[8] M. Pelikan, D. E. Goldberg, and E. Cantu-Paz, "BOA: the Bayesian optimization algorithm," in the Genetic and Evolutionary Computation Conference. Orlando, USA: Morgan Kaufmann, 1999, pp. 525-532.
[9] X. Yao, "Evolving artificial neural networks," Proceedings of the IEEE, vol. 87, pp. 1423-1447, 1999.
[10] T. White, B. Pagurek, and F. Oppacher, " ASGA: improving the ant system by integration with genetic algorithms," in the third Conference on Genetic Programming (GP/SGA'98). Madison, USA, 1998, pp. 610-617.
[11] P. Turney, "Myths and legends of the Baldwin effect," in Workshop on Evolutionary Computation and Machine Learning at the 13th International Conference on Machine Learning. Bari, Italy, 1996, pp. 135-142.
[12] F. G. Lobo and D. E. Goldberg, "Decision making in a hybrid genetic algorithm," in IEEE International Conference on evolutionary Computation. Piscataway, USA: IEEE Press, 1997, pp. 122-125.
[13] W. E. Hart, "Adaptive global optimization with local search," Doctoral Dissertation. San Diego: University of California 1994.
[14] W. E. Hart, C. R. Rosin, R. K. Belew, and G. M. Morris, "Improved evolutionary hybrids for flexible ligand docking in AutoDock," in Optimization in Computational Chemistry and Molecular Biology, C. A. Floudas and P. M. Pardalos, Eds.: Springer 2000, pp. 209-230.
[15] W. M. Spears, "Adapting crossover in a genetic algorithm," in the Fifth Conference on Evolutionary Programming. San Diego, USA: MIT press, 1995, pp. 367-384.
[16] Y. Davidor, "Epistasis variance: suitability of a representation to genetic algorithms," Complex Systems, vol. 4, pp. 369-383, 1990.
[17] X. Zhang and B. A. Wandell, "A spatial extension of CIELAB for digital color image reproduction," in SID Symposium Technical Digest, vol. 27, 1996, pp. 731-735.