Heuristic Optimisation for the Minimum Distance Problem

Evelyn Yu-San Chan

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

August 2000

Acknowledgements

I should like to take this opportunity to thank my Director of Studies, Dr John Bland, for his confidence in me to carry out this research. I should also like to thank my Second Supervisor, Dr John Baylis, for his endurance in guiding me in Coding Theory. Most importantly, I should like to thank my family and friends for their tremendous support throughout these years.

Abstract

In this thesis two heuristic optimisation techniques are investigated, with the aim of obtaining minimum distance estimates of particular error-correcting codes.

Minimum distances are important in Coding Theory because the error-correcting capability of codes is dependent upon them. However, the exact minimum distances of many practically important codes are still unknown and so, for codes such as quadratic residue (QR) codes, the minimum distance problem (MDP) remains open. In this thesis the mathematical development of QR codes is presented and the MDP for particular QR codes is then investigated using heuristic optimisation techniques.

Heuristic techniques are necessary due to the combinatorial nature of the problem and the non-convex nature of the solution-space. In this thesis the particular heuristic optimisation techniques applied to the MDP are tabu search (TS) and ant colony optimisation (ACO). TS uses a neighbourhood search procedure in which use is made of a memory facility, called the tabu list, which enables the search to progress beyond local optima in the quest for a global optimum. Recently TS has been successfully applied to Bose-Chaudhuri-Hocquengham codes using a short-term memory approach. In this thesis longer-term strategies and a number of 'memory' lists are developed within a TS algorithm, to find minimum distance estimates for large QR codes.

Recently several discrete optimisation techniques have been developed by analogy with physical and biological processes (for example, simulated annealing and genetic algorithms). Because analogies with natural phenomena have been used to successfully derive non-deterministic heuristic methods which can be applied to NP-complete combinatorial optimisation problems, there is a growing interest in this approach to problem solving. ACO is a population-based method which was inspired by the behaviour of a colony of ants and their ability to 'optimise' their collective endeavours. In this thesis ACO is formulated in an error-correcting code context. A basic ACO algorithm is first presented and then developed to incorporate the co-operation amongst members of the colony. The developed ACO algorithm, together with TS as a local improvement phase, is then applied to large QR codes to obtain minimum distance estimates.

Contents

Chapter 1

Error-correcting Codes and Heuristic Optimisation

1.1 Introduction

Codes were invented to convert complicated human language to artificial language. Nowadays messages are often converted to sequences of binary digits $\{0,1\}$ and these digits are usually sent via a communication channel such as satellite communication links or cables. Typical examples are uses of the Internet and sending e-mail messages. Although communication channels are now more reliable than in past decades, in theory, no real channel is ideal because there could be disturbances and other interference that may corrupt the signal transmitted through the channel. These errors may be caused by lightning, earthquakes, thermal noise etc. It is important to note that for some applications a single error could lead to a disaster. Error-correcting codes are therefore used to correct errors when messages are distorted through the transmission in a communication channel [Roman 1997, Hill 1986].

Coding theory techniques are used to determine the correctability of errors that occur when information is transmitted from one source to another. Binary codes are widely used as errors in these codes can be easily corrected once the locations of the errors are known. The minimum distance, d, (to be defined later) of a code is the major parameter affecting its error-correcting performance. However, the exact minimum distance values are still unknown for many practically important codes, and therefore the estimation of d, herein called the minimum distance problem (MDP), for such codes remains unsolved.

In general, discrete optimisation methods are tools that may be used to solve combinatorial problems. The problem of finding the minimum distance d for any linear code is known to be an NP-complete problem, that is, both solvable in non-deterministic polynomial-time (i.e. in class NP) and can be translated into any other problem in class NP [Welsh 1988, Garey and Johnson 1979, Berlekamp et al. 1978]. One way to approach the MDP is to use approximation algorithms [Foulds 1984, Foulds 1981]. However, these algorithms cannot guarantee the final obtained solution to be optimal but the computation time associated with the approximate solution can be given. Therefore, the time that is required for finding a high quality approximation becomes important.

Approximation methods may be divided into two categories: general algorithms applicable to a wide variety of combinatorial problems and algorithms that are tailored to specific problems [Baykasoglu et al 1999, Carlton and Barnes 1996]. In this thesis, the heuristic optimisation techniques studied are tabu search (TS) and ant colony optimisation (ACO). These two algorithms are general techniques that have been successfully applied to different combinatorial problems such as scheduling [Logendran and Sonthinen 1997, Forsyth and Wren 1997], the travelling salesman problem [Carlton and Barnes 1996, Gambardella and Dorigo 1995] and the quadratic assignment problems [Gambardella et al 1999, Laguna et al. 1991]. In this thesis, the ideas behind these algorithms are then adapted to specifically tackle the minimum distance problem (MDP).

1.2 Importance of Minimum Distances

A code is a set of strings over a certain alphabet; for example, a code whose alphabet is Z_2 is a binary code. A block code is a set C of strings of symbols, of fixed length

n, with the symbols being chosen from a finite 'alphabet' A. If $|A| = q$, then q^n is the number of strings of length n from this alphabet. Generally C contains only a small fraction of the q^n 'words' and those in C are called the codewords. The codewords are the 'meaningful words of the language' and they correspond to the actual word transmitted. In general the transmission channel is 'noisy' so that the channel induces errors. The fact that $|C| << q^n$ is the key to the receiver being able to recover the original message with a high probability of success, provided that no pair of codewords are too 'similar' and that C has been well selected.

If A is a finite field F, the set of all words of length n over F, F^n, can be regarded as a vector space over F with dimension n. An (n,k) code over F is a k-dimensional subspace of F^n and an (n,k) binary code has 2^k vectors or codewords in it, where 'binary' now means that F is $GF(2)$ [Vermani 1996]. The weight $w(x)$ of a word is the number of non-zero digits in the word x. Let $x, y \in C$ and both words are of length n. The Hamming distance $d(x, y)$, is the "difference" between words x and y, meaning the number of places in which x and y have different alphabet symbols, so

$$d(x, y) = w(x - y) \tag{1.1}$$

Suppose C is a code with at least two codewords. The minimum distance $d(C)$ of C is the smallest distance between two distinct codewords. In symbols,

$$d(C) = \min\{d(c,d) \mid c,d \in C, c \neq d\} \tag{1.2}$$

Since $c \neq d$ implies that $d(c,d) \geq 1$, the minimum distance of a code must be at least 1. One way to determine d is by listing all pairs of the codewords, but for larger

linear codes, it would be more efficient to compute the weights of the individual words, which will be proved in Chapter 2; $d(C) = \min_{\underline{c} \in C, \underline{c} \neq 0} (w(\underline{c}))$ for linear codes.

The determination of the error-correcting capability of a code is dependent on the previously mentioned minimum distance of a code. Therefore, the exact minimum distance of the code must be known. Any code C is t error-correcting if and only if $d(x, y) \geq 2t + 1$ [Baylis 1998]. An important class of linear codes is the quadratic residue (QR) family. Very little is known about $d(C)$ for the larger members of this class. This thesis tackles the estimation of $d(C)$ computationally.

Over the years researchers have worked on determining the values of d and have found different upper bound results but exact values were not found for large QR codes. In many cases the minimum distances were found by computer searches over restricted subsets of codes. Coppersmith and Seroussi (1984) justified a method that minimum weight codewords could be found by restricted searches. The aim of this study is to obtain minimum distance estimates for some particular QR codes, using developed TS and ACO algorithms.

1.3 Optimisation Methods

One of the most commonly used optimisation algorithms is an iterative search method known as neighbourhood search [Glover and Laguna 1997]. This method aims to solve most combinatorial problems by generating solutions and testing their quality. With each move, a neighbourhood solution is generated in which a slight variation from the previous solution is achieved consisting of configurations that can be reached from one transition. With reference to Table 1.1, the major goal is to minimise the cost *(cost)*. The iterative search begins with an initial solution *(soln)* with cost *cost*. For each move *(move)*, a neighbourhood solution *nbhdsoln* is

generated. If *nbhdsoln* has a lower cost then both *cost* and *soln* are replaced by *nbhdcost* and *nbhdsoln*, respectively. Otherwise, another neighbour is generated and its cost compared to *cost*. This process continues until the termination criterion is reached (*move* = *maxmove)* or no possible improvement can be found. However there are some disadvantages of using iterative search methods; they often terminate at local minima which are sometimes dependent on the setting of the initial solution. In spite of this, iterative search methods are usually easy to formulate and a single run may be executed in a reasonable amount of computation time.

```
Begin
• Generate initial collection of solution, soln
• Evaluate cost, the cost of soln
      For move = 1 to maxmove
      mincost = cost
            For nbhd = 1 to maxnbhd
            • Generate nbhdsoln, a neighbour of soln
            • Evaluate nbhdcost, the cost of nbhdsoln
              /* If a solution is improved, update and move */
                  If nbhdcost < cost
                     soln = nbhdsoln
                     cost = nbhdcost
                     nbhd = maxnbhd
                  End(If)
            End (For)
            If nbhdcost ≥ mincost /* Trapped in local minimum - exit */
               move = maxmove
            End(If)
      End(For)
• Output soln and cost
End
```

Table 1.1 Pseudo-code for neighbourhood search.

Two classes of iterative search methods are local search and population-based search. The local search algorithm is an iterative search method and is based on the generation of neighbourhood solutions (see Table 1.1). One example of a local search algorithm is the hill-climbing technique in which the search generates solutions that have slight variations from the solutions constructed previously. This is done via a competitive strategy that favours better solutions. Tabu Search (TS) and simulated

annealings (SA) are types of local search algorithm. Population-based search uses a collection of data to form different solutions. This is done by first selecting members of the population to be 'parents' and then making changes to these parents to produce 'children' as new solutions. This type of search differs from local search since local search is based on using a single 'parent' to generate one or more 'children' whereas population-based search generates new solutions by re-combining aspects of two or more existing solutions (parents). Genetic algorithms (GA) and ant colony optimisation (ACO) are examples of population-based search algorithms.

1.4 Heuristic Approaches

This section reviews uses of different heuristic approaches for solving large combinatorial problems. In the last two decades, heuristic optimisation techniques have become popular for solving optimisation problems. Although heuristic techniques are still based on approximation, continued investigation has inspired growing interest in their ability to address realistic problems. For example, simulated annealing (SA) is based on physical processes in metallurgy and genetic algorithms (GA) seek to reproduce the biological behaviour of genes. More recently, a meta-heuristic (tabu search) was introduced by Glover (1989) as a master strategy that guides and modifies other heuristics to produce solutions beyond local optimality. Tabu Search (TS) uses memory of previously obtained solutions as an intelligent problem solving tool. Ant colony optimisation (ACO) [Bonabeau et al. 1999] simulates the ability of an ant colony to 'optimise' its collective activities. From all these recent heuristic findings, a fundamental aim is to unite artificial intelligence with optimisation to tackle NP-complete problems.

Simulated annealing (SA) was originally proposed by Metropolis et al (1953). It was an analogy of the physical process of heating a solid to melting point, followed

by cooling to a crystallise state with a perfect pattern where the cooling method is conducted under controlled conditions in order to avoid any imperfections. Almost thirty years later Kirkpatrick et al (1983) adapted this simulation process to search for feasible solutions in a combinatorial problem, with the aim of converging to an optimal solution. For this reason the algorithm became known as 'simulated annealing'.

SA is an iterative improvement algorithm that attempts to avoid entrapment in a local optimum by sometimes accepting a move that deteriorates the objective function value (i.e. cost). The algorithm begins with a feasible solution and proceeds to investigate the neighbourhood of this solution via a sequence of moves in search of optimal solutions. An improved solution (i.e. one with reduced cost) is always accepted but if a solution does not improve, then the solution may still be accepted according to some specified probability. The probability of acceptance decreases exponentially to zero as the number of moves increases. Without this facility, the search may become trapped in a local minimum, thus exploration would cease. Apart from its use in traditional application areas such as in scheduling [Cho and Kim 1997], the quadratic assignment problem [Wilhelm and Ward 1987] and the travelling salesman problem [Černy 1985], there are some successful applications of simulated annealing in Coding Theory. Aarts and Laarhoven (1992) briefly reviewed the application of local search to a special class of coding problems: covering and packing. Zhang and Ma (1994) designed an annealing-based algorithm to carefully tune some of the control parameters to find minimum distances for Bose-Chaudhuri-Hocquengham (BCH) codes. Although this research is not focussed on simulated annealing techniques, the results obtained by Zhang and Ma (1994) will be used for comparison purposes.

The genetic algorithm (GA) was first introduced as a highly robust search algorithm [Backhouse et al 1997, Felicity 1996] and in the last twenty years it has been used successfully as an optimisation device. This idea of GA is based on the evolutionary process of biological organisms in nature (i.e. a string of genes (chromosomes) of a particular member of a species). During the course of evolution, those individuals that adapt to the environment (i.e. fit individuals) will have a better chance of survival and reproducing offspring whereas those less fit individuals will be eliminated. This means that the fittest individuals will survive and the combination of two of the fittest individuals (parents or ancestors) is likely to produce fitter progenies (offspring). As a result, fitter offspring are evolved.

GA has been applied to a wide variety of problems such as the QAP [Lim et al 2000], the TSP [Larranaga et al 1999], scheduling [Sriskandarajah et al 1998] and maximum set problems [Hifi 1997]. The basic optimisation idea of GA is to work with populations of solutions and attempt to guide the search towards improvement by testing the fitness of survival of each solution. Each solution contributes towards the next generation in proportion to its fitness by using a probability function that is biased to favour those with superior quality.

Tabu Search (TS) is a meta-heuristic that was first introduced by Glover (1986) as an intelligent procedure that incorporates a hill-climbing technique and the use of adaptive memory [Glover 1990, Glover 1989]. The hill-climbing technique is a responsive exploration process that encourages the search towards the best solution (e.g. minimum distance) and the adaptive memory feature enables the search of solution-space to be effective and economical. In the neighbourhood search method (see Section 1.3), the search continues until no possible improvements can be found. However, many problems contain numerous local minima, which will trap the search.

To overcome this problem, TS uses a memory facility, called the 'tabu' list which forbids solutions to be re-visited for a period of time. Furthermore, TS uses a neighbourhood search procedure, in which the tabu list enables the search to progress beyond local optima in order to find a global optimum. Consequently TS has been used successfully in different areas such as the QAP [Taillard 1991, Skorin-Kapov 1990], flow-shop problems [Moccellin and Nagano 1998] and the layout problem [Song and Vanellli 1992, Bland and Dawson 1991]. In recent years, TS has been applied to Coding Theory to investigate lower bounds of some constant weight codes [Bland and Baylis 1997]. Following Zhang and Ma (1994), Bland and Baylis (1995) examined the effectiveness of estimating d for BCH codes [Hoffmsn et al. 1991] using tabu search. Their results showed that lower minimum distances may be obtained (compared to SA). In this thesis, longer-term strategies and a number of 'memory' lists are developed within a TS algorithm. The aim is, firstly, to improve the minimum distance values obtained by Bland and Baylis (1995); since the exact minimum distances for the BCH codes are known they can be used as target values for the developed TS algorithm. Then minimum distance estimates are sought for different QR codes, whose exact minimum distances are not known.

Recently several heuristic optimisation techniques have been developed by analogy with physical and biological processes (for example, simulated annealing [Kirkpatrick et al. 1983] and genetic algorithms [Backhouse et al. 1997]). Because analogies with natural phenomena have been used to successfully derive non-deterministic heuristic methods which can be applied to NP-complete problems, there is a growing interest in this approach to problem solving. Ant colony optimisation (ACO) [Dorigo and Gambardella 1997, Dorigo et al. 1991] is a population-based method which was inspired by the behaviour of a colony of ants and their ability to

'optimise' their collective endeavours [Beckers et al. 1992]. Dorigo et al. (1991) have drawn inspiration from the workings of natural ant colonies to derive an optimisation approach to difficult combinatorial problems like the travelling salesman problem [Dorigo et al. 1996], the quadratic assignment problem [Gambardella et al. 1999, Maniezzo et al. 1994] and other related fields [Dorigo and Caro 1999]. Essentially, ACO works by equating the notion of a candidate solution with the route taken by an ant between two places. Ants leave pheromone as they travel, and routes that correspond to good solutions will have a stronger pheromone trail than routes that lead to poor solutions. ACO incorporates a positive-feedback mechanism which reinforces the pheromone trails of good solutions. The ACO algorithm uses information obtained by a number of individual agents (computational ants) to form a (population) optimisation problem. Publications on ACO are relatively recent and have shown that it may yield high quality results [Bland 1999]. An aim of this study is to develop the ant colony optimisation algorithm and investigate the MDP for some BCH and QR codes.

1.5 Thesis Outline

In Chapter 2, the mathematical developments required for constructing QR codes are presented. Chen et al (1994) and Coppersmith and Seroussi (1984) developed generator matrices for some small QR codes but no general approach has been investigated. Also, to date, no research has been published that investigates minimum distance estimates for large QR codes using heuristic optimisation techniques.

In Chapter 3, a detailed explanation of the short-term memory TS algorithm is presented with minimum distance results for different QR codes. This is followed, in the same chapter by an introduction to the basic ACO algorithm in the context of the MDP. In Chapter 4, longer-term strategies (intensification/diversification approaches)

and a number of 'memory' lists (tabu list and influential candidate list) are developed within a TS algorithm, to find minimum distance estimates for large QR codes. Chapter 5 presents the development of a new strategy to incorporate co-operation amongst members of an ant colony. The developed ACO algorithm, together with TS as a local improvement phase, is then applied to tackle the MDP. A summary of this thesis and its achievements are given in Chapter 6, together with the possible improvements to the developed algorithms and scope for further research.

Chapter 2

Mathematical Structure of Codes and Generator Matrices

2.1 Introduction

Errors in binary codes can easily be corrected once the locations of the errors are known. As a result these codes are widely used. A code can correct up to t errors provided its minimum distance $d \geq 2t+1$ [Pretzel 1992, Hill 1986]. The exact minimum distances, however, are still unknown for many practically important codes, such as the quadratic residue codes. The estimation of d for such codes is therefore an important open problem.

Linear codes are the most widely studied types of codes because they are closed under addition and scalar multiplication. This feature makes the encoding and decoding of source messages much easier. In addition, the minimum distance of any linear code can be determined by examining the individual codewords, rather than calculating the distance between each pair of codewords. This makes the estimation of d more efficient and economical.

Cyclic codes are a class of linear codes [Baylis 1998] which have the additional structure of being closed under cyclic shifts. This in turn leads to efficient encoding and decoding techniques, which make such codes of practical importance. All linear codes, cyclic codes in particular, are completely determined by a generator matrix. For cyclic codes, there are generator matrices in which each row is the cyclic shift of the previous row.

The BCH family of codes [Lin 1970] is a family of linear cyclic codes whose design principles make considerable use of the theory of finite fields. This theory makes it possible to design a code with a minimum distance greater than or equal to

some pre-assigned value [Augot and Levy-dit-Vehel 1996]. The actual minimum distances in the BCH family are in many cases known exactly which makes BCH codes practically useful. This also makes them convenient 'test codes' for assessing the efficiency of the heuristic search algorithms used for estimating the minimum distance of linear codes defined by generator matrices.

Quadratic residue (QR) codes are cyclic codes of prime length p with the Galois Field $GF(s)$ as their alphabet, where s is another prime which is a quadratic residue modulo p. Several upper and lower bounds on the value of d are known for QR codes and in some cases the exact value of d is known [Coppersmith and Seroussi 1984]. For many of the larger ones, however, the theoretical bounds (a and b), which are consequences of the mathematical structure of these codes, is the best information we have. For such cases, where $a \leq d \leq b$ with $a < b$, heuristic search methods will be used to try to find a codeword of weight less than b and hence improve the upper bound.

2.2 Linear Codes

Linear codes (of which the BCH and QR codes are examples) are vector spaces whose elements are the codewords, each of which is a string of length n whose individual symbols are elements of some finite field. This finite field is also the set of scalars of the vector space which means a code C is linear if it is closed under addition and scalar multiplication. Furthermore, any linear code must contain the zero word because $0\underline{c} = \underline{0}$. The minimum distance of a linear code is the minimum weight of any non-zero codeword. To prove this consider a linear code C where $d(C) = d$ and let w be the minimum weight of any non-zero codeword. For some pair of non-zero distinct codewords, $\underline{c}_1, \underline{c}_2 \in C$ at distance d

$$d = d(\underline{c}_1, \underline{c}_2) = w(\underline{c}_1 - \underline{c}_2) \geq w \tag{2.1}$$

and if $\underline{c}$ is a codeword of weight w, then

$$w = w(\underline{c} - \underline{0}) = d(\underline{c}, \underline{0}) \geq d \tag{2.2}$$

Hence $w = d$. So if a code is known to be linear, its minimum distance is the minimum weight of the non-zero codewords. Hence, $d(C)$ may be obtained by scanning the weights of the individual codewords, rather than calculating $d(\underline{c}_1, \underline{c}_2)$ for every pair of codewords. This makes the computational time for finding d quicker since the time is now only linear in the size of C, rather than quadratic.

For binary codes the field is $GF(2)$. If C has dimension k and minimum distance $d(C) = d$, then C is called a $[n, k, d]$-code. A set $S = \{\underline{c}_1, \underline{c}_2, ..., \underline{c}_k\}$ is linearly independent if and only if $a_1, a_2, ..., a_k$ all equal to zero is the only solution of

$$a_1\underline{c}_1 + a_2\underline{c}_2 + ... + a_k\underline{c}_k = \underline{0} \tag{2.3}$$

The set of all linear combinations of the vectors in a given set $S = \{\underline{c}_1, \underline{c}_2, ..., \underline{c}_k\}$ is the linear span of S and is denoted by $\langle S \rangle$ and if a nonempty subset $S^* \subseteq S$ is linearly independent and $\langle S^* \rangle = \langle S \rangle$, then S^* forms a basis for $\langle S \rangle$. An advantage of linear codes is that C can be described by using a basis of C, rather than having a list of all the codewords. For this reason, linear codes are easier to generate. If C has length n and dimension k, then any matrix whose rows form a basis for C is a generator matrix for C.

A generator matrix G for a linear code C is a $k \times n$ matrix whose k rows are sets of codewords making up a basis of the vector space. Hence k is simply the dimension of this space. If G is a generator matrix for C and if $\underline{\alpha}$ is a word of

length k written as a row vector, then a codeword $\underline{c}$ is a unique linear combination of the words in the basis, so can be written as

$$\underline{c} = \underline{\alpha} G \tag{2.4}$$

for some $\underline{\alpha}$ in A^k where A is the alphabet. So for a linear code C given by a generator matrix

$$G = \left(\underline{c}_1, \underline{c}_2, ..., \underline{c}_k\right)^T \tag{2.5}$$

the problem of finding $d(C)$ reduces to minimising $w(\underline{\alpha} G)$. From now on the members of A^k will be referred to as alpha-vectors.

2.3 Polynomials and Cyclic Codes

A polynomial $f(x)$ can be expressed as $f(x) = a_0 + a_1 x + ... + a_{n-1} x^{n-1}$ where $a_0, ..., a_{n-1}$ are the coefficients, chosen from some field F. The set of all polynomials over F is denoted by $F[x]$, with the operations of addition, subtraction and multiplication of polynomials defined in the usual way. For each $f \in F[x]$, the degree of f is the largest power of x with non-zero coefficient. A code C of length n can be represented as a set of polynomials over F of degree at most $n-1$, in which $f(x)$ represents the codeword $a_0\ a_1\ a_2 .... a_{n-1}$.

Cyclic codes are linear codes over a field F which are closed under cyclic shifts. That is, if $\underline{c} = c_0\ c_1 \ ... \ c_{n-1}$ and $\underline{c} \in C$, then $\underline{c}^* \in C$ where $\underline{c}^* = c_{n-1}\ c_0\ c_1 ... c_{n-2}$. Now identify $\underline{c}$ with the polynomial $c_0 + c_1 x + ... + c_{n-1} x^{n-1}$ in the ring $F[x]$ modulo $x^n - 1$, then C becomes the principal ideal in this ring consisting of all polynomial multiples of $g(x)$, where $g(x)$ is an irreducible factor of $x^n - 1$ [Baylis 1998]. $g(x)$ is called a generator of C.

This connection between cyclic codes and polynomial algebra has important implications for encoding and decoding techniques, and for our purposes it provides a useful form of generator matrices for cyclic codes.

If $g(x)$ is $g_0 + g_1 x + ... + g_{n-k} x^{n-k}$, then the cyclic code having $g(x)$ as its generator polynomial has

$$G = \begin{bmatrix} g_0 & g_1 & .. & .. & g_{n-k} & 0 & .. & .. & 0 \\ 0 & g_0 & .. & .. & g_{n-k-1} & g_{n-k} & 0 & .. & 0 \\ .. & .. & : & : & : & : & : & : & 0 \\ : & : & : & : & : & : & : & : & : \\ 0 & .. & .. & g_0 & .. & .. & .. & .. & g_{n-k} \end{bmatrix}$$

as its generator matrix. Since the generator matrix is in an echelon form its rows are linearly independent and if row 1 corresponds to a codeword each row of the generator matrix is a codeword by the cyclic property. If g_0=0 or g_{n-k}=0, then the first or the last digit of all words in C are equal to 0 which contradicts the property that cyclic codes are closed under cyclic shifts. Hence, $g_0 \neq 0, g_{n-k} \neq 0$. Cyclic codes have generator matrices in which every row is a cyclic shift of the first row, although not all generator matrices of a given cyclic code are of this form. G clearly has k rows, so $dim(C) = k$.

2.4 Quadratic Residues Codes

The QR family of codes are believed to be good codes (in terms of their error-collecting capability) but less is known about their minimum distances. QR codes are linear and they share with BCH codes the property of being cyclic. Quadratic Residue (QR) codes are cyclic codes of prime length p with the Galois field $GF(s)$ as their alphabet, where s is another prime which is a quadratic residue modulo p. For the QR codes, the length of any codeword will be denoted as p.

- **Quadratic Residues**

Let p be an odd prime. If $1 \leq x \leq p$, then x is a quadratic residue modulo p if there is a natural number y for which

$$y^2 \equiv x (\text{mod } p) \tag{2.6}$$

If x has a square root modulo p, then x is a quadratic residue. Otherwise, x is a quadratic non-residue modulo p.

The following theorems (Theorem 2.1 to 2.3) [Rosen 1986, Macwilliams and Sloane 1977] provide the background for the mathematical development of the generator matrices for QR codes. These particular theoretical results form the mathematical basis for the generator matrices developed and used in this thesis.

Theorem 2.1

Suppose p is an odd prime, then there are exactly $\frac{p-1}{2}$ quadratic residues modulo p among the $p-1$ numbers $\{1,2,...,p-1\}$.

Proof

The $p-1$ numbers split into $\frac{p-1}{2}$ pairs with the squares of each member of a given pair being congruent to each other modulo p. Explicitly, we have

$$1^2 \equiv (p-1)^2 \bmod p$$

$$2^2 \equiv (p-2)^2 \bmod p$$

$$\left(\frac{p-1}{2}\right)^2 \equiv \left(p-\left(\frac{p-1}{2}\right)\right)^2 \bmod p$$

Furthermore, all of 1^2, 2^2, $3^2 \ldots \left(\frac{p-1}{2}\right)^2$ are different modulo p, because, if a^2, b^2 are two members of this list which are congruent modulo p, and $a^2 \equiv b^2 \bmod p$,

then $a^2 - b^2 \equiv 0 \mod p$, i.e. $(a-b)(a+b) \equiv 0 \mod p$, and because p is prime, this implies $a-b \equiv 0 \bmod p$ or $a+b \equiv 0 \bmod p$ (Euclid's lemma).

Hence $a \equiv b \mod p$ or $a \equiv -b \mod p$. Since a and b are chosen from the range $1,2,3,...,\frac{p-1}{2}$, the first possibility implies $a = b$, and the second is impossible. Hence in the set $\{1,2,...,p-1\}$ there are only $\frac{p-1}{2}$ distinct squares, namely $1^2, 2^2, 3^2, ..., \left(\frac{p-1}{2}\right)^2$. Hence there are $\frac{p-1}{2}$ quadratic residues in the set and the rest in the set are non-residues.

Theorem 2.2

2 is a quadratic residue modulo p whenever $p \equiv \pm 1 (\bmod 8)$.

Proof

Let p be an odd prime and a is an integer not divisible by p. The Legendre symbol $\left(\frac{a}{p}\right)$ is defined by

$$\left(\frac{a}{p}\right) = \begin{cases} 1 & \text{if } a \text{ is quadratic residue modulo } p \\ -1 & \text{if } a \text{ is a quadratic non-residue modulo } p \end{cases}$$

From Gauss lemma, $\left(\frac{a}{p}\right) \equiv (-1)^l \pmod{p}$ where l is the number of least positive residues modulo p of the integers $a, 2a, 3a..., \left(\frac{p-1}{2}\right) \cdot a$ that are greater than $\frac{p}{2}$ where $a = 2$. $l = \frac{p-1}{2} - \left[\frac{p}{4}\right]$ is the number of least positive residues that are greater than $\frac{p}{2}$. If $p \equiv 1 \pmod 8$, then $p = 8k+1$ for some integer k, then

$\frac{p-1}{2}-\left[\frac{p}{4}\right]=4k-\left[2k+\frac{1}{4}\right]=2k\equiv 0$ (modulo 2). If $p\equiv 7$ (modulo 8), $p=8k+7$ for some integer k, then $\frac{p-1}{2}-\left[\frac{p}{4}\right]=4k+3-\left[2k+\frac{7}{4}\right]=2k+2\equiv 0$ (modulo 2). Hence 2 is a quadratic residue if $p\equiv\pm 1$ (modulo 8).

2.5 Augmented and Expurgated Quadratic Residue Codes

For binary QR codes, $p\equiv\pm 1\,(\mathrm{mod}\,8)$ and 2 is a quadratic residue. Following from Theorem 2.1, the quadratic residues and the non-residues have degrees of $\frac{p-1}{2}$. The QR codes Q, N, $\overline{Q}$ and $\overline{N}$ are defined by their generator polynomials $q(x)$, $n(x)$, $(x-1)q(x)$ and $(x-1)n(x)$ respectively, where

$$q(x)=\prod_{i\in Q}\left(x-\alpha^{i}\right) \tag{2.7}$$

$$n(x)=\prod_{j\in N}\left(x-\alpha^{j}\right) \tag{2.8}$$

and α is a primitive p^{th} root of unity in some extension field of $GF(2)$. An element $\alpha\in GF\left(2^{r}\right)$ is primitive if $\alpha^{m}\neq 1$ for $1\le m\le 2^{r}-1$. It follows from Theorem 2.1 that Q, N have degree $\frac{p-1}{2}$. Since the generator polynomial of any cyclic code must be a factor of $x^{p}-1$, it follows that

$$x^{p}-1=(x-1)q(x)n(x) \tag{2.9}$$

The augmented QR codes Q and N generated by $q(x)$ and $n(x)$ are equivalent linear codes. By definition of equivalence of linear codes [Baylis 1998], this means that a generator matrix G^{*} for N can be obtained from a generator matrix G of Q, by column interchanges on G and row operations on G. Hence $q(x)$ can be obtained

by a permutation of the co-ordinate indices from $n(x)$ and vice versa. Consequently, the expurgated QR codes $\overline{Q}$ and $\overline{N}$ generated by $(x-1)q(x)$ and $(x-1)n(x)$ are also equivalent and

$$\overline{Q} \subseteq Q \text{ and } \overline{N} \subseteq N \tag{2.10}$$

The minimum distance for any augmented QR codes, Q and N is $d \geq \sqrt{p}$ and if $p = 8m-1$, then the minimum distance satisfies $d^2 - d + 1 \geq p$. These results are normally called the square root bounds [Macwilliams and Sloane 1977].

2.6 Generator Matrices

Theorem 2.3

The dimension of an augmented Quadratic Residue code is $\frac{p+1}{2}$.

Proof

From Theorem 2.1, it is known that there are exactly $\frac{p-1}{2}$ quadratic residues and quadratic residues modulo p, the dimension of the generator matrix is

$$k = p - \deg(g(x)) = p - \left(\frac{p-1}{2}\right) = \frac{p+1}{2} \tag{2.11}$$

Since QR codes are linear and cyclic, one way of getting a generator matrix is by cyclic shift the first row of codeword provided its linear combination will span the code. Hence, the weight of a generator matrix, G, is the number of non-zero digits of a word in G. For the expurgated QR codes, the dimension of a generator matrix is $\frac{p-1}{2}$ (see example on page 23).

- **Generator Matrices with weight = $\frac{p-1}{2}$**

Since there are $\frac{p-1}{2}$ quadratic residues modulo p between 1 and p, one form of generator matrix is by cyclic shift of the first row of a codeword in which the first row is formed by

$$G_{1x} = \begin{cases} 1 & x \in Q \\ 0 & x \notin Q \end{cases} \tag{2.12}$$

Q denotes the set of quadratic residues modulo p and each codeword corresponding to a row of G has weight $\frac{p-1}{2}$, so $d(C) \leq \frac{p-1}{2}$. For the larger QR codes the generator matrix described will have its rows with the relatively large weight of $\left(\frac{p-1}{2}\right)$. Therefore the search may not find low weight words in a feasible time. An alternative G matrix with significantly lower weight rows was found by using the algebra of a polynomial representation of the codewords, and in some cases this led to better results. For these, the best known results are of the form $a \leq d \leq b$, (a and b are positive integers).

- **Generator Polynomials**

A method of finding generator polynomial is based on factorisation through cyclotomic cosets. Consider

$$m_i(x) = \gcd\left(c_i(x), x^p - 1\right) \tag{2.13}$$

where $m_i(x)$ is the minimal polynomial of α^i obtained from each cyclotomic coset $C_i = \{i, 2i, 2^2 i, 2^3 i, \ldots\}$ by writing

$$c_i(x) = \prod_{j \in C_i} \left(x - \alpha^j\right) \tag{2.14}$$

Q and N are both disjoint unions of cyclotomic cosets. If r is the number of cyclotomic cosets in Q and l is the number in N, then

$$x^p - 1 = (x-1)\prod_{i=1}^{r} m_i(x) \prod_{j=1}^{l} m_j(x) \tag{2.15}$$

Since Q and N are equivalent, the generator polynomials for augmented QR code are

$$g(x) = n(x) = \prod_{i=1}^{r} m_i(x) \tag{2.16}$$

or

$$g(x) = q(x) = \prod_{j=1}^{l} m_j(x) \tag{2.17}$$

and the generator polynomials for expurgated codes are

$$g(x) = (x-1)q(x) \tag{2.18}$$

or

$$g(x) = (x-1)n(x) \tag{2.19}$$

The first row of a generator matrix derived from the generator polynomial is the binary string in which the i^{th} component (where $i = 1,2,\ldots, p$) is 1 if and only if i corresponds to a power of x with non-zero coefficient. Using the developed alternative G matrices, with rows of weight less than $\frac{p-1}{2}$, reduced search times and better (lower) estimates of d are obtained.

The dimension of augmented QR codes and expurgated QR codes are then fixed at $\frac{p+1}{2}$ and $\frac{p-1}{2}$, respectively. Using the generator polynomial as alternative G matrices, with rows of weight less than or equal to $\frac{p-1}{2}$, results in reduced search times and better (lower) estimates of d than the generator matrix described in equation (2.12).

- **Example**

In order to show the process of obtaining various generator matrices using a generator polynomial, the following worked example is given for small p. In this study the process will be programmed as a computational algorithm that can cope with large values of p.

Consider $p = 7$, from equation (2.6), the quadratic residues are

$$1^2 \equiv 1 \text{ (modulo 7)}$$

$$2^2 \equiv 4 \text{ (modulo 7)}$$

$$3^2 \equiv 2 \text{ (modulo 7)}$$

and the non-residues are 3, 5 and 6. From equations (2.13) and (2.14)

$$\begin{array}{r|l} & x^3+x+1 \\ \hline x^4+x^2+x+1 & x^7 \qquad\qquad +1 \\ & \underline{x^7+x^5+x^4+x^3} \\ & x^5+x^4+x^3 \qquad +1 \\ & \underline{x^5 \quad +x^3+x^2+x} \\ & x^4 \quad +x^2+x+1 \\ & \underline{x^4 \quad +x^2+x+1} \\ & 0 \end{array}$$

Hence, the generator polynomial for the augmented QR codes are $g(x)=n(x)=x^3+x+1$ and $g(x)=q(x)=x^3+x^2+1$. The generator polynomial for the expurgated QR codes are $g(x)=(x-1)n(x)=x^4+x^2+x+1$ and $g(x)=(x-1)q(x)=x^4+x^3+x^2+1$. Hence, the generator matrices for these codes are

$$g(x)=n(x)=x^3+x+1 \quad \Rightarrow \quad \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}$$

$$g(x)=q(x)=x^3+x^2+1 \quad \Rightarrow \quad \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

$$g(x)=(x-1)n(x)=x^4+x^2+x+1 \quad \Rightarrow \quad \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \end{bmatrix}$$

$$g(x)=(x-1)q(x)=x^4+x^3+x^2+1 \quad \Rightarrow \quad \begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}$$

Chapter 3

Heuristic Optimisation

3.1 Introduction

Optimisation is a process that aims to find the best (i.e. optimal) value of an objective function and the values of the associated variables (i.e. the optimum point). The meaning of best depends on the context of the optimisation problem. Also, some problems may be subject to constraints, which restrict the feasible solution space.

Classical discrete-variable optimisation problems include the quadratic assignment problem [Kelly et al. 1994], the travelling salesman problem [Foulds 1981] and transportation problems [Foulds 1984]. For example, an unconstrained problem such as the quadratic assignment problem requires the determination of the best (i.e. minimum) total interconnection distance of all possible layouts (i.e. object-location couplings), together with the associated costs. In this study optimisation is applied to the determination of minimum distance estimates of error-correcting codes.

Consider the minimum distance problem (MDP), (C, w), in which C is a linear block code comprising the set of codewords of length n that are linear combinations of k basis vectors $\underline{\beta}_i$, where $\underline{\beta}_i = \{\beta_{ij}\}$, $i = 1,2,...,k$, $j = 1,2,...,n$, and $\beta_{ij} \in \{0,1\}$. That is, codeword $\underline{c} \in C$ $(\underline{c} \neq 0)$ is such that

$$\underline{c} = \sum_{i=1}^{k} \alpha_i \underline{\beta}_i \quad \text{modulo } 2 \tag{3.1}$$

where $\alpha_i \in \{0,1\}$. Also w is an integer-valued objective function, $w: C \rightarrow Z$, such that $\underline{c} \in C$ has an associated weight, $w(\underline{c})$. For a particular codeword $\underline{c}_x \in C$ with $\underline{c}_x = \{c_{xj}\}$, $j = 1,2,...,n$ and $c_{xj} \in \{0,1\}$, then

$$w(\underline{c}_x) = \sum_{j=1}^{n} c_{xj} \tag{3.2}$$

The MDP is to find a non-zero codeword $\underline{c}^* \in C$ that minimises w over C, that is,

$$w(\underline{c}^*) = \min_{\underline{c} \in C - \{\underline{0}\}} w(\underline{c}) \tag{3.3}$$

Since C is a binary linear code it $w(\underline{c}^*) = \min_{\underline{c} \in C - \{\underline{0}\}} w(\underline{c})$ ce, d, is given by $w(\underline{c}^*)$. However, the cardinality of C is $|C| = 2^k$ so that when k is large the determination of d by complete enumeration may not be practical. In this study QR codes with values of k of the order of 10^2 will be investigated and so the MDP expressed by equation (3.3) is combinatorial.

The combinatorial aspect of the determination of the minimum distance of a linear code gives rise to certain difficulties in terms of the optimisation procedure. In the first instance many combinatorial optimisation problems are NP-complete; hence heuristic solution techniques become necessary. A second difficulty lies with the general lack of ability of heuristic methods to determine global optima of non-convex problems. Non-convex problems have numerous local optima and the first encountered usually traps or terminates the procedure.

Tabu search and ant colony optimisation are heuristic combinatorial optimisation techniques that have the ability to avoid entrapment by local optima and hence search for a global optimum. Details of basic

implementations of these algorithms are now presented, first in general terms, and then in an error-correcting code context for the MDP.

3.2 Tabu Search

Tabu search (TS) was introduced by Glover [Glover 1990, Glover 1989] as a heuristic approach to combinatorial optimisation; a computational problem solving tool that aims to find an optimum beyond local optimality.

In general TS is an iterative method in which the search progresses through a solution space via neighbourhoods of the current solution. Use is made of a memory facility called the tabu list [Laguna 1992, Glover 1990a] to avoid returning to previously visited solutions and cycling in solution space. The tabu list (whose length is usually specified) contains forbidden, or tabu, solutions and operates as a first-in first-out queue of the most recent solutions. Besides its use in the avoidance of cycling, the tabu list enables the search to progress beyond local optima in the search for a global optimum.

Tabu search, in basic terms, may be formulated as shown in the pseudo-code in Table 3.1. Besides problem dependent data, typical generic input data includes the size of the tabu list and the maximum number of moves through solution space (*maxmove*), which may be used as a stopping criterion (i.e. when *move* = *maxmove*).

With reference to Table 3.1, starting from an initial solution, *soln*, with initial cost (i.e. objective function value), *cost*, the tabu list is initialised with the inclusion of the start solution (or its characterisation). At the start of the algorithm the best (i.e. lowest) cost, *bestcost*, and the associated solution, *bestsoln*, will be *cost* and *soln*, respectively. Then within the *move* loop, a neighbourhood of the current solution is generated and, with reference to the

contents of the tabu list, the non-tabu neighbourhood solution with the lowest cost, *minsoln*, is identified (where *minsoln* has cost *mincost*). The tabu list is updated with the inclusion of *minsoln* (or its characterisation). If *mincost* is less than the current value of *bestcost* then both *bestcost* and *bestsoln* are updated. As a final step in the loop, the search formally moves to *minsoln* in solution space. On exiting the *move* loop the current value of *bestcost* and the associated *bestsoln* are output.

Begin
- Input data
- Select initial solution, *soln*
- Evaluate initial cost, *cost*
- Initialise tabu list
- Current best solution *bestsoln* = *soln* and *bestcost* = *cost*

For *move* = 1 to *maxmove*
- Generate neighbourhood of *soln*
- Identify the non-tabu neighbourhood solution, *minsoln*, with smallest cost, *mincost*
- Put *minsoln* in the tabu list
- **If** *mincost* < *bestcost*
  Bestcost = *mincost*
  Bestsoln = *minsoln*
  End (If)
- Update the search, *soln* = *minsoln*

End (For)
- Output *bestsoln* and *bestcost*

End

Table 3.1 Pseudo-code for the basic tabu search algorithm.

The above process of moving through solution space via the non-tabu neighbourhood solutions of least cost is the most basic tabu search algorithm. In recent years many refinements have been incorporated into the basic algorithm. These refinements (conceptual and computational) include the use of,

- mechanisms to diversify the search by 'jumping' to unexplored regions of the search space [Kelly et al. 1994]

- longer term memory facilities, in conjunction with the (short term) tabu list, to aid exploration of the current region of search space [Xu et al. 1996]
- variable length tabu lists to make efficient use of computing time [Skorin-Kapov 1994]
- efficient computational mechanisms to manage tabu list(s) [Hertz et al. 1997]

With various refinements, tabu search has been used by several researchers to successfully investigate classical NP-complete discrete optimisation problems, such as the quadratic assignment problem [Skorin-Kapov 1990], the travelling salesman problem [Carlton and Barnes 1996], scheduling [Glover et al. 1994] and transportation problems [Zachariasen and Dam 1996].

In the following section (basic) tabu search is formulated in an error-correcting context for the MDP. Based on this, a developed algorithm is presented in Chapter 4.

3.3 Tabu Search for the MDP

Tabu search for the MDP may be expressed formally as follows. Consider a codeword $\underline{c}_x \in C$, then a mapping function m^* may be defined on C such that $m^* : C \rightarrow C$ transforms one particular codeword to another (i.e. $\underline{c}_x$ to $\underline{c}_y$). Hence each $\underline{c} \in C$ has an associated set, $M_{\underline{c}}$, where $M_{\underline{c}} \subseteq M$ and comprises mappings $m^* \in M$ that are applied to $\underline{c}$,

$$M_{\underline{c}} = \left\{ m^* \mid m^* \in M, \underline{c} \in C \right\} \tag{3.4}$$

The above mappings lead to the generation of neighbourhoods. The neighbourhood $N(\underline{c}_x)$ of codeword $\underline{c}_x \in C$ is given by

$$N(\underline{c}_x) = \{\underline{c}_y \mid \underline{c}_y = m^*(\underline{c}_x), m^* \in M_{\underline{c}_x}\} \tag{3.5}$$

It is assumed that if $\underline{c}_x$ and $\underline{c}_y \in C$ then

$$\underline{c}_y \in N(\underline{c}_x) \Leftrightarrow \underline{c}_x \in N(\underline{c}_y) \tag{3.6}$$

The tabu search procedure is such that if $\underline{c}_x \in C$ is the current codeword then the search moves to the non-tabu codeword in $N(\underline{c}_x)$ with lowest weight, $\underline{c}_{\min}$, which is now stored in the tabu list T. Hence the search progresses via a sequence of moves from $\underline{c}_x \in C$ to $\underline{c}_{\min} \in C$ where

$$w(\underline{c}_{\min}) = \min_{\underline{c}_y \in N(\underline{c}_x) - T} w(\underline{c}_y) \tag{3.7}$$

For the MDP (see equation (3.3)), each codeword $\underline{c}$ is obtained from an alpha-vector $\underline{\alpha} = \{\alpha_i\}, i = 1,2,...,k$; $\underline{c} = \underline{\alpha} G$, where G is the generator matrix for code C. In this sense, codewords may be characterised by their alpha-vectors and, for the MDP, tabu search will operate in $\underline{\alpha}$-space. A merit of using alpha-vectors rather than codewords is that, while both are binary strings, an alpha-vector is shorter than its associated codeword and is therefore a useful characterisation for computational reasons.

In implementation terms, codewords $\underline{c} \in C$ will be characterised by $\underline{\alpha} = \{\alpha_i\}$, $i = 1,2,...,k$, where $\alpha_i \in \{0,1\}$. The search space is the set of all binary k-tuples, F_2^k, (excluding $\underline{\alpha} = \underline{0}$), and the mapping m is from one k-tuple to another $m^* : F_2^k \rightarrow F_2^k$ such that $\underline{\alpha}_x \in F_2^k$ becomes

$$\underline{\alpha}_y = m^*(\underline{\alpha}_x) \in F_2^k \tag{3.8}$$

Furthermore, m^* may be considered as k-tuples, $\underline{m}^*$, such that

$$\underline{\alpha}_y = \underline{\alpha}_x \oplus \underline{m}^* \tag{3.9}$$

where $\oplus$ means bitwise addition modulo 2. Hence, the set $M_{\underline{\alpha}}$ is such that $M_{\underline{\alpha}} \subseteq M$ and compromises those k-tuples $\underline{m}^* \in M$ that are applied to $\underline{\alpha}$,

$$M_{\underline{\alpha}} = \{\underline{m}^* \mid \underline{m}^* \in M, \underline{\alpha} \in F_2^k\} \tag{3.10}$$

The neighbourhood of $\underline{\alpha}_x \in F_2^k$ is given by

$$N(\underline{\alpha}_x) = \{\underline{\alpha}_y \mid \underline{\alpha}_y = \underline{\alpha}_x \oplus \underline{m}^*, \underline{m}^* \in M_{\underline{\alpha}_x}\} \tag{3.11}$$

Because of the one-to-one correspondence between the elements of C and F_2^k equations (3.5), (3.6) and (3.7) remain valid and equation (3.7) is implemented as

$$w(\underline{c}_{\min}) = \min_{\underline{\alpha}_y \in N(\underline{\alpha}_x) - T} w\left(\sum_{i=1}^{k} \alpha_{yi} \underline{\beta}_i\right) \tag{3.12}$$

where $\underline{\alpha}_y = \{\alpha_{yi}\}$, $i = 1,2,...,k$ and T contains codeword characterisations (alpha-vectors) and the basis vectors $\underline{\beta}_i$, $i = 1,2,...,k$, are those that comprise the generator matrix of QR codes (cf. Chapter 2). In the study of cyclic codes, the initial solution (without loss of generality) is taken to be $\alpha_{01} = 1, \alpha_{0i} = 0; i = 2,3,...,k$. Also, the moves applied to $\underline{\alpha}_x \in F_2^k$ comprise set $M_{\underline{\alpha}_x}$, where

$$M_{\underline{\alpha}_x} = \{m^* \mid m^* \in M, \underline{\alpha}_y = \underline{\alpha}_x \oplus \underline{m}^*,\ d_H(\underline{\alpha}_x, \underline{\alpha}_y) = 1\} \tag{3.13}$$

and $d_H(\underline{\alpha}_x, \underline{\alpha}_y)$ denotes the Hamming distance between $\underline{\alpha}_x$ and $\underline{\alpha}_y$. Hence Equation (3.11) simplifies to

$$N(\underline{\alpha}_x) = \{\underline{\alpha}_y \mid d_H(\underline{\alpha}_x, \underline{\alpha}_y) = 1\} \tag{3.14}$$

and the neighbourhood size is therefore

$$|N(\underline{\alpha}_a)| = k \tag{3.15}$$

where, to achieve equation (3.14), the elements of $\underline{m}^*$ are such that all except one are zero.

In computational terms, basic tabu search for the MDP may be formulated as shown in the flowchart in Figure 3.1. With reference to Figure 3.1, the input data includes code details such as the generator matrix, G, and the length of the alpha-vectors, k, where $k = \frac{p+1}{2}$ for augmented QR codes and $k = \frac{p-1}{2}$ for expurgated QR codes (see Chapter 2). Also, search details are required; the start alpha-vector $\underline{\alpha}_0$, the maximum number of moves (stopping condition), $m_{\max}$, and the (maximum) length of the tabu list, L. Other variables such as move number, m, alpha-vector, $\underline{\alpha}$, codeword, $\underline{c}$, and minimum distance, d, are then initialised as the current best values (using the subscript 'best'). The initial size of the tabu list is $|T| = 2$ and the contents of T are the zero vector (length k), $\underline{0}$, and $\underline{\alpha}_0$.

On entering the move loop a neighbourhood of the current point-of-search ($\underline{\alpha}_0$) is generated and denoted by $N(\underline{\alpha}_0)$. In the MDP, following Zhang and Ma (1994), $N(\underline{\alpha}_0)$ comprises alpha-vectors $\underline{\alpha}_i$, $i = 1,2,...,k$ where $\underline{\alpha}_i = \underline{\alpha}_0 \oplus \underline{m}_i$. Here $\oplus$ means bitwise addition modulo 2 and $\underline{m}_i$ is a vector of length k in which all elements are zero, except for element i, which is one. For each neighbourhood alpha-vector its associated codeword weight is calculated.

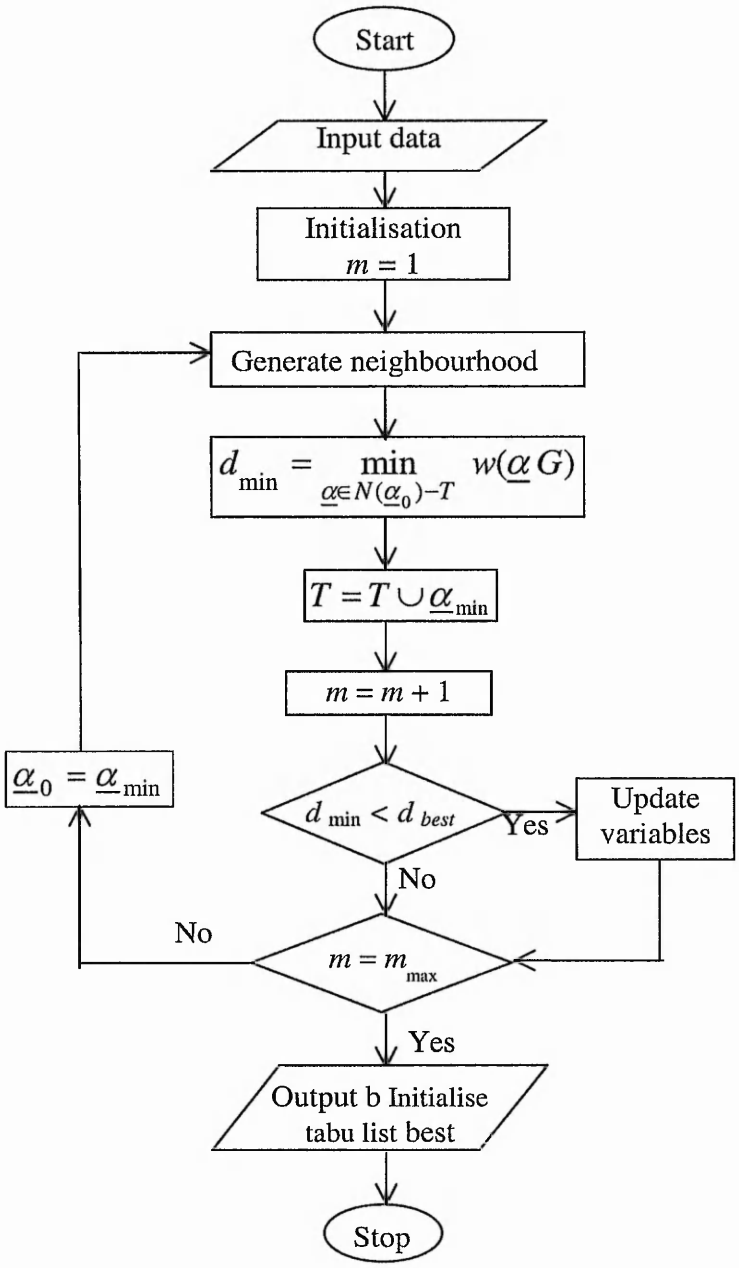


Figure 3.1 Flowchart of tabu search algorithm.

The non-tabu neighbourhood alpha-vector with the least codeword weight is then identified, i.e. $\underline{\alpha}_{\min}$ with (least) codeword weight $d_{\min}$. The tabu list is updated to include $\underline{\alpha}_{\min}$, (depending on the value of L, the tabu list, on a sweep of the move loop, may release its least recent alpha-vector so that it loses its

tabu status). If $d_{\min}$ is less than the current value of d_{best} then d_{best}, $\underline{\alpha}_{best}$, $\underline{c}_{best}$ and $\underline{m}_{best}$ are all updated. Whether or not updating has taken place, provided the maximum number of moves ($m_{\max}$) is not achieved, the point-of-search formally moves to $\underline{\alpha}_{\min}$ in alpha-space. Once the maximum number of moves has been performed the algorithm exits the move loop and outputs the current values for d_{best}, $\underline{\alpha}_{best}$, $\underline{c}_{best}$ and $\underline{m}_{best}$, prior to terminating.

In this study, the (basic) tabu search algorithm for the MDP has the following features.

- **Neighbourhood Structure and Search Strategy**

The neighbourhood structure described earlier in this section is the same as that used by Zhang and Ma (1994) in their investigation of minimum distances of BCH codes by simulated annealing. This neighbourhood structure is computationally easy to generate and its size is reasonable (k, the dimension of the code). This strikes a balance between being too large (which may be too time consuming to investigate) and too small (which would be quick to investigate but poor for search purposes).

The size of the neighbourhood is also a consideration when choosing a strategy for the movement of the search through successive neighbourhoods. In the outlined algorithm (see Figure 3.1) the strategy adopted to determine $\underline{\alpha}_{\min}$ was 'best found in neighbourhood' (BFIN). This strategy generally gives a relatively high level of solution quality (i.e. good $d_{\min}$ values) compared with, for example, a 'first found in neighbourhood' strategy (FFIN). With FFIN, neighbourhood alpha-vectors are investigated sequentially and the point-of-search moves to the (non-tabu) alpha-vector associated with the first

encountered weight value that is less than d_{best} (which may be greater than $d_{\min}$). If none are encountered then FFIN reverts to BFIN. FFIN may perform quicker than BFIN but at the possible loss of solution quality.

- **Tabu List Management**

The tabu list has specified length L and is managed so that it operates as a first-in first-out queue of alpha-vectors. The value of L has a direct bearing on the operation of the algorithm, both in terms of solution quality and execution time. If L is too small then, although it is possible to rapidly check for tabu alpha-vectors, there is a danger that the search will become trapped in a cycle within $\underline{\alpha}$-space. Alternatively, if L is too large then, while reducing the risk of cycling, the computation time required to check the tabu list may become prohibitive.

In the tabu search algorithm outlined in the flowchart in Figure 3.1 the length of the tabu list grows from $|T| = 2$ (where move number $m = 0$) up to $|T| = L$ (when move number $m = L - 2$) and is then maintained at a fixed length L. For each move m such that $(m+2) > L$ the least recent alpha-vector (starting with $\underline{\alpha}_0$) is released from the tabu list, although $\underline{0}$ is a permanent member throughout the search.

3.4 Minimum Distance Results using Tabu Search

The TS algorithm described in the previous section was applied to a number of BCH codes and different QR codes. In this study the maximum number of moves $m_{\max} = 100$ and the tabu list length $L = m_{\max} + 2$. The start solution

(alpha-vector) and neighbourhood structure are those described in the previous section.

All numerical experiments were performed using software written in Visual C++ and conducted on a Pentium II 266 MHz computer (similarly for all computational results reported in this thesis).

The BCH codes were the 10 codes investigated by Bland and Baylis (1995). Results are presented in Table 3.2. With reference to Table 3.2, n and k denote the length and dimension of the code, respectively, d_{BCH} is the known minimum distance, d_{best} is the obtained minimum distance using the basic TS algorithm, with associated best move m_{best} and execution time t_{best}.

Code	n	k	d_{BCH}	d_{best}	m_{best}	t_{best} (h:m:s)
1	127	64	21	26	1	0:00:00.44
2	127	57	23	24	1	0.00:00.28
3	127	50	27	31	0	0:00:00.00
4	255	115	43	60	9	0:00:13.28
5	255	107	45	65	2	0:00:03.63
6	255	99	47	68	11	0:00:12.41
7	255	91	51	71	6	0:00:06.48
8	255	87	53	70	3	0:00:02.58
9	255	79	55	78	1	0:00:00.99
10	255	71	59	77	4	0:00:02.86

Table 3.2 Minimum distances of BCH codes using the TS algorithm.

The results for d_{best} and m_{best} in Table 3.2 are identical to those of Bland and Baylis (1995). For code 3, m_{best} = 0 indicates that the value of d_{best} is associated with the start alpha-vector $\underline{\alpha}_0$ (i.e. $\underline{\alpha}_{best} = \underline{\alpha}_0$). As seen in Table 3.2, TS obtains the d_{best} values very fast and early in the search.

The following QR codes are investigated, those with generator matrices with weight $w(G) = \frac{p-1}{2}$, augmented QR codes ($n(x)$ and $q(x)$, see Chapter 2),

and expurgated QR codes (*(x-1)n(x)* and *(x-1)q(x)*, see Chapter 2). For each type of QR code, 10 codes (characterised by prime number p) were investigated using the basic TS algorithm.

The results for QR codes with $w(G) = \frac{p-1}{2}$ are presented in Table 3.3. In Table 3.3, d_{QR} denotes the known minimum distance value (Codes 1 to 4) or best known minimum distance bounds (Codes 5 to 10).

Code	p	d_{QR}	$w(G)$	d_{best}	m_{best}	$t_{best}(h:m:s)$
1	71	11	36	20	33	0:00:01.81
2	79	15	40	27	4	0:00:00.22
3	97	15	49	34	17	0:00:02.36
4	103	19	52	31	10	0:00:01.37
5	113	11-15	57	40	24	0:00:04.73
6	137	13-21	69	46	28	0:00:09.18
7	167	15-23	84	63	4	0:00:02.91
8	191	15-27	96	71	56	0:00:46.25
9	193	15-27	97	76	15	0:00:11.32
10	199	15-31	100	72	27	0:00:22.73

Table 3.3 Tabu search results using $w(G) = \frac{p-1}{2}$.

Results for augmented (*n(x)*) QR codes are presented in Table 3.4 for the same values of p as in Table 3.3.

Code	p	d_{QR}	$w(G)$	d_{best}	m_{best}	$t_{best}(h:m:s)$
1	71	11	15	11	24	0:00:01.05
2	79	15	23	15	16	0:00:00.99
3	97	15	27	16	95	0:00:11.26
4	103	19	27	20	13	0:00:01.98
5	113	11-15	31	16	1	0:00:02.22
6	137	13-21	47	29	90	0:00:24.06
7	167	15-23	43	35	4	0:00:02.91
8	191	15-27	47	36	25	0:00:19.72
9	193	15-27	57	38	1	0:00:01.31
10	199	15-31	51	40	9	0:00:08.40

Table 3.4 Tabu search results using *n(x)*.

A comparison of Tables 3.3 and 3.4 reveals that the QR codes with $w(G) = \frac{p-1}{2}$ produced relatively poor results. For these codes the minimum distance associated with the start alpha-vector, i.e. d_0, is approximately 100% higher than the equivalent augmented QR code in Table 3.4.

As seen in Table 3.4, for Code 1 (p = 71) TS was able to obtain the known minimum distance ($d_{best} = d_{QR} = 11$). Also the (relative superior) values of d_{best} were obtained very fast and early in the search (except for Codes 3 and 6).

The results for the other QR codes investigated; augmented ($q(x)$) and expurgated ($(x-1)n(x)$ and $(x-1)q(x)$), are given Tables A1 to A3, respectively, in Appendix A. The best results in terms of obtained minimum distances (d_{best}) are summarised in Table 3.5.

As seen in Table 3.5 the best results are, in general, obtained early in the basic 100 move search. This indicates that to obtain improvements (i.e. lower d_{best} values) the TS algorithm needs to be developed to include features such as diversification and long-term memory. A developed TS algorithm for the MDP is presented in Chapter 4.

To illustrate the operation of the basic tabu search algorithm, example convergence curves for the case of Code 6 in Table 3.3 and Table 3.4 are shown in Figure 3.2 and Figure 3.3, respectively.

Code	p	$g(x)$	d_{QR}	d_{best}	m_{best}	t_{best} $(h:m:s)$
1	71	$n(x)$	11	11	24	0:00:01.05
1	71	$(x-1)\,n(x)$	12	12	20	0:00:00.40
1	71	$(x-1)\,q(x)$	12	12	34	0:00:00.68
2	79	$n(x)$	15	15	16	0:00:00.99
2	79	$(x-1)\,n(x)$	16	16	15	0:00:00.41
2	79	$(x-1)\,q(x)$	16	16	14	0:00:00.34
3	97	$q(x)$	15	15	96	0:00:04.79
3	97	$(x-1)\,n(x)$	16	16	37	0:00:04.28
4	103	$(x-1)\,n(x)$	20	20	7	0:00:00.93
4	103	$(x-1)\,q(x)$	20	20	7	0:00:01.26
5	113	$(x-1)\,n(x)$	12-16	16	1	0:00:00.23
6	137	$(x-1)\,n(x)$	14-22	28	45	0:00:11.26
6	137	$(x-1)\,q(x)$	14-22	28	57	0:00:16.42
7	167	$n(x)$	15-23	35	4	0:00:02.91
7	167	$q(x)$	15-23	35	2	0:00:01.81
7	167	$(x-1)\,n(x)$	16-24	36	11	0:00:05.99
7	167	$(x-1)\,q(x)$	16-24	36	3	0:00:02.25
8	191	$(x-1)\,q(x)$	16-28	36	3	0:00:02.31
9	193	$(x-1)\,n(x)$	16-28	38	1	0:00:01.31
10	199	$n(x)$	15-31	40	9	0:00:08.40

Table 3.5 Table search results for QR codes.

With reference to Figure 3.2 it is seen that the weight of the generator matrix is 69 and the search encounters several local minima within the first 28 moves (56, 54 and 50) until it reaches its final value of 46 from move 28. In comparison, Figure 3.3 shows the effect of using augmented QR code $n(x)$. Although only one local minimum is encountered, the final weight of 29 is evidence of the superiority of using a generator polynomial with $w(G) \leq \frac{p-1}{2}$ instead of the quadratic residues (where $w(G) = \frac{p-1}{2}$) as the generator matrix (see Chapter 2).

Associated with the convergence curves shown in Figure 3.2 and Figure 3.3 are the search histories presented in Figure 3.4 and Figure 3.5, respectively. With reference to both Figures 3.4 and 3.5, the graph points indicate the lowest non-tabu weight values of each neighbourhood generated by the tabu search procedure and hence represent the weights associated with the 'path' the search takes in $\underline{\alpha}$-space.

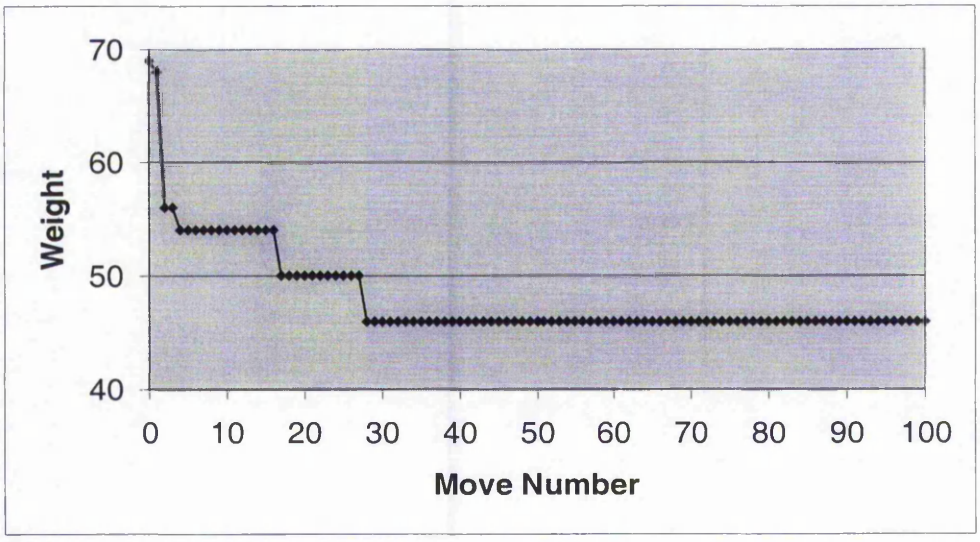


Figure 3.2 Convergence curve for p = 137 using $w(G) = \frac{p-1}{2}$.

In Figure 3.4 it is seen that the overall lowest non-tabu neighbourhood weight was found in the 28th neighbourhood investigated, thereafter only higher weight values are found. Figure 3.5 shows that the search process with a generator matrix that has lower weight than that of Figure 3.4, quickly reached several neighbourhood solutions with distance very close to the final value ($d_{best} = 29$). With reference to Figure 3.3, although d_{best} was found late in the search, the computational time is extremely quick and d_{best} is quite close to d_{QR} which indicates the merits of a TS approach to the MDP.

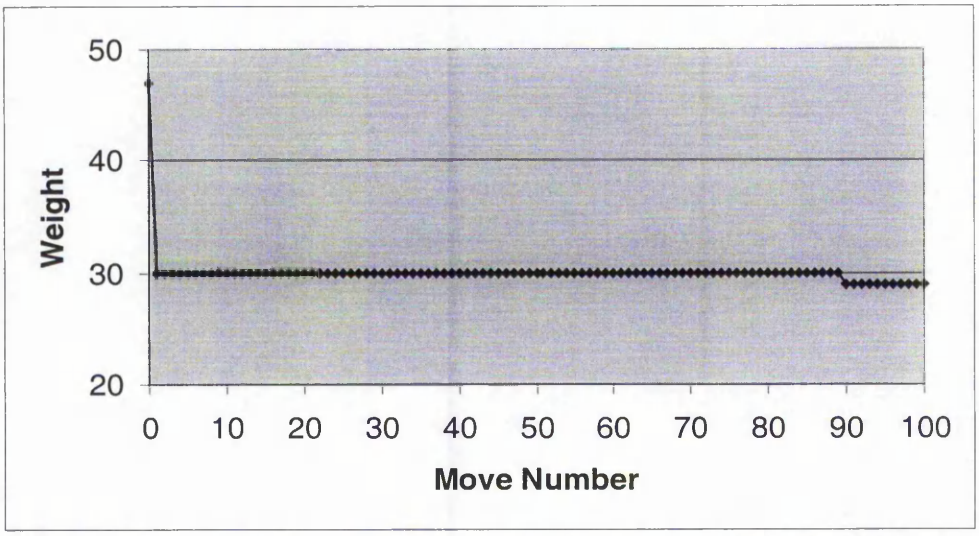


Figure 3.3 Convergence curve for p = 137 using augmented QR code $n(x)$.

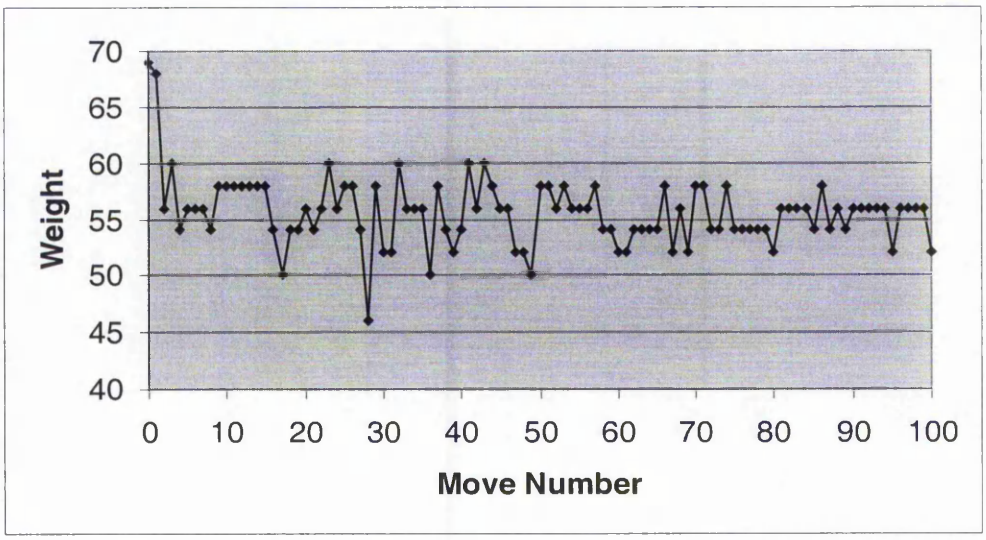


Figure 3.4 Search history for p = 137 using $w(G) = \frac{p-1}{2}$.

Although minimum distances found using this basic implementation of tabu search were of an acceptable quality, an aim is to achieve lower distances. Therefore some enhanced strategies will be presented in Chapter 4; these include diversification and intensification strategies, long-term influential candidate lists and intermediate-term memory tabu list.

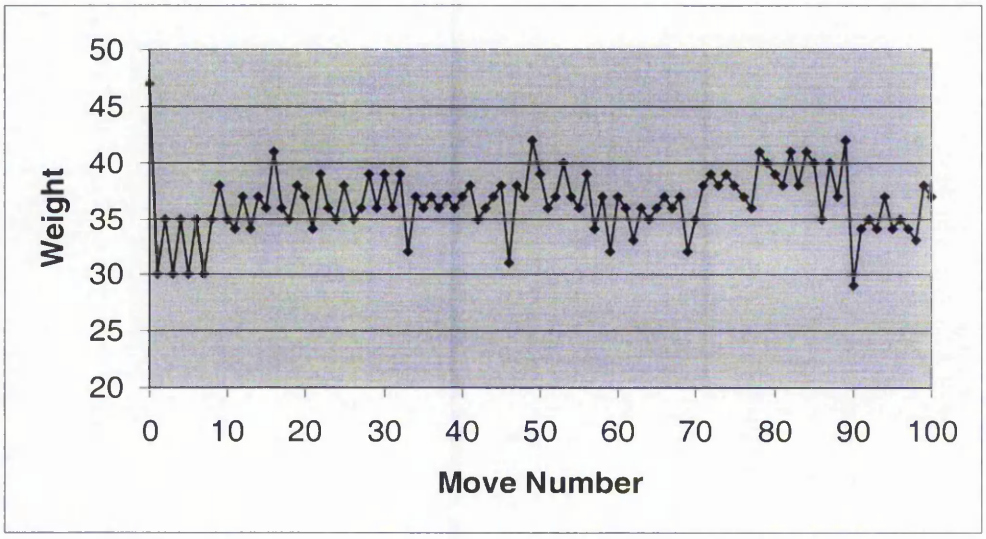


Figure 3.5 Search history for $p = 137$ using augmented QR code $n(x)$.

3.5 Ant Colony Optimisation

The natural phenomenon of how (real) ants find the shortest path between a food source and the nest [Beckers et al. 1992] has been an inspiration to solve many optimisation problems. Social insects like ants tend to make a collective decision and although they are almost blind they communicate between one another by depositing a substance called *pheromone*. At an instant in time, a moving ant leaves a trail of pheromone to inform other ants of the visited path. Any ant that encounters the trail marked by others is likely to detect and follow the same path. Furthermore, each ant that detects the trail will reinforce it with its own pheromone; thus the trail becomes stronger. However, ants that cannot detect any pheromone laid on the trail previously, or an isolated ant, essentially moves randomly.

The ability of real ants, in broad terms, to 'optimise' the route from their nest to a food source is illustrated in Figure 3.6. With reference to Figure 3.6(a), a number of ants leave the nest to seek food and arrive at a decision point in the

form of an obstacle blocking their path. The ants need to decide whether to turn left or right. At this point in time they have no information to aid their decision and so choose randomly; consider 50% of the ants turn in each direction, as illustrated in Figure 3.6(b). Assuming each ant travels at the same speed, those ants that turn right arrive at the food source quicker (because the path is shorter) than those that turn left. Consequently, these ants (i.e. the right-turners) are likely to choose the same path for their return journey (to the nest with food) because of the existing pheromone trail (see Figure 3.6(c)). Furthermore, on the return journey more pheromone is deposited, which reinforces that on the existing trail. As a result, subsequent ants leaving the nest are more likely to follow the path with the higher level of pheromone, that is, turn right when they meet the obstacle, as shown in Figure 3.6(d). In this manner, over a period of time, the ants will 'optimise' their route from the nest to the food source.

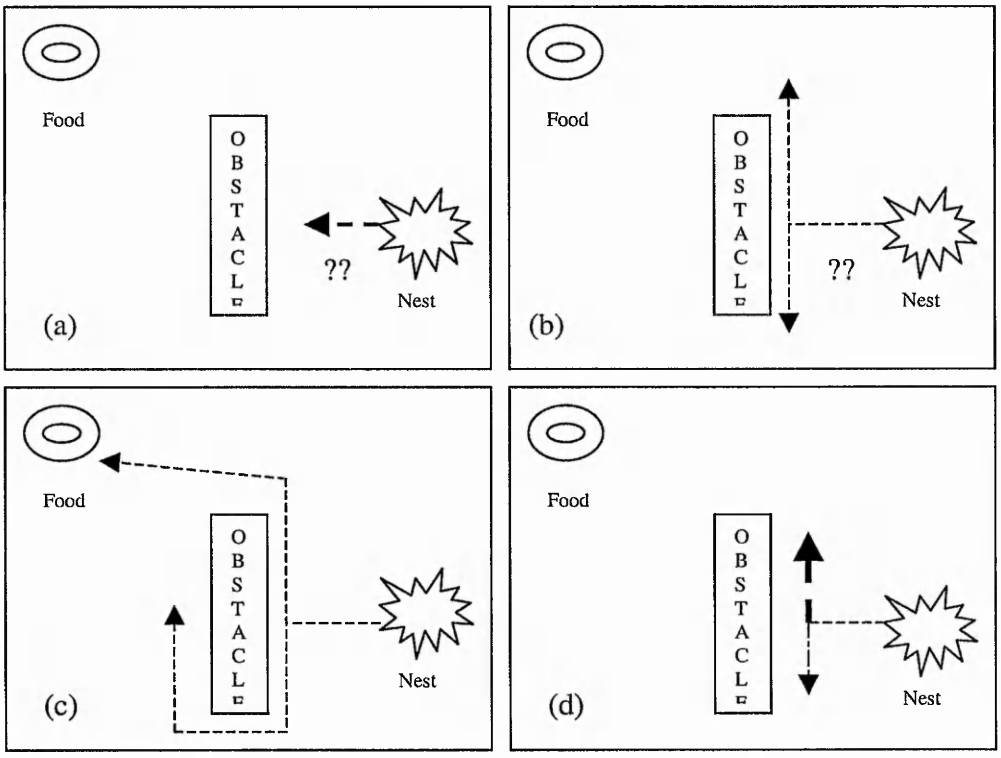


Figure 3.6 Behaviour of real ants.

The process described above, that is, decision-making aided by reinforcement of information (i.e. positive feedback) is an example in the natural world of autocatalytic behaviour. Over the past few years several heuristic optimisation techniques have been developed by analogy with physical and biological processes (for example, simulated annealing [Zhang and Ma 1994] and genetic algorithm [Backhouse et al. 1997]). Because analogies with natural phenomena have been used to successfully derive non-deterministic heuristic techniques, which can tackle NP-complete combinatorial optimisation problems, there is now great interest in this approach to problem solving.

Ant colony optimisation (ACO) is the general name given to the approach to problem solving by analogy with the collective performance of (real) ants. A basic ACO algorithm, called the ant system, which uses artificial (or computational) ants, is now described.

3.6 Ant System

The ant system was introduced by Dorigo [Dorigo et al. 1991] as (like tabu search) a heuristic approach to combinatorial optimisation. Also like tabu search, ant system is an iterative algorithm. However, compared with tabu search, there is a major conceptual difference in the optimisation procedure. Ant system does not search by means of a 'path' of successive solutions in an appropriate solution space, rather, it aims to progressively improve an aid to decision-making so that, at each iteration, better decisions may be made (i.e. ones that lead to better objective function values).

A feature of ant system (and ACO in general) is the trace intensity matrix. This matrix holds information for decision-making and its contents represents the level of pheromone deposited by a colony of computational ants. The trace

intensity matrix is updated each iteration with the information obtained during the iteration so that decisions that lead to improved objective function values may be made and, like tabu search, algorithm termination at non-global optima may be avoided.

Other (technical/computational) differences between ant system and tabu search are that ant system (and ACO in general) is a population-based algorithm and that it is stochastic in nature.

The (basic) ant system may be formulated as shown in the pseudo-code in Table 3.6. Besides problem dependent data, typical generic input data includes the maximum number of discrete time steps (*maxstep*), which may be used as a stopping condition (i.e. when *step* = *maxstep*). These 'time' steps do not measure the passage of time in a chronological sense but are used as counters in the development of the decision-making capability of the algorithm. Also the number of computational ants (*ant*) in the colony, i.e. *colsize*, is required.

With reference to Table 3.6, the elements of the colony trace intensity matrix are initialised to a small positive number (to get the algorithm started in computational terms. By strict analogy this number should be zero as pheromone has yet to be deposited).

On entering the *step* loop each ant 'constructs' a solution (in a probabilistic manner; explained later), *soln[ant]*, *ant* = 1,2,...,*colsize*, using data in the colony trace intensity matrix. Also, the cost (i.e. objective function value) associated with each solution is calculated, *cost[ant]*, *ant* = 1,2,...,*colsize*.

The next operation in the pseudo-code in Table 3.6, i.e. the local improvement phase, has no analogy with the endeavours of real ants; it is a purely computational device to (try to) improve all (or some) of the current

values of *cost[ant]*, *ant* = 1,2,...,*colsize*. If by the use of a local improvement algorithm, a value of *cost[ant]*, *ant* $\in$ {1,2,...,*colsize*}, is improved, then the variable is updated together with the associated value/expression of *soln[ant]*. Whether or not a local improvement phase is included in the ant system algorithm (its omission would be on the grounds of purism), the current solution with the lowest cost is identified, that is *minsoln* with cost *mincost*, where

$$mincost = \min \{cost[ant]\} \quad ant = 1,2,\ldots,colsize \tag{3.16}$$

On the first time step (*step* = 1) the current (overall) best solution (*bestsoln)* and associated cost (*bestcost*) will be *minsoln* and *mincost*, respectively. On subsequent time steps, *bestcost* and *bestsoln* are updated if *mincost*, given by equation (3.16), is less than the current value of *bestcost*.

Next, with reference to Table 3.6, the amount of pheromone deposited by each ant during the current time step is calculated and expressed as elements of a trace intensity matrix for each ant. Details of this matrix are given later, in Section 3.7, but here it is noted that there is an inverse relationship between the amount of pheromone deposited (i.e. magnitude of the matrix elements) by ant number *ant* and its cost, *cost[ant]*. In this manner, better solutions (i.e. those with lower costs) have greater pheromone deposits and hence will receive greater favour in the decision-making phase (solution construction) of the next time step. The 'step-deposit' trace intensity matrix for the colony of ants, representing the collective performance of the colony in the current time step, is obtained by combining all those of the individual ants.

As a final operation within the step loop the colony trace intensity matrix is updated with the inclusion of the above step-deposit matrix. The updated colony trace intensity matrix is used as the current trace intensity matrix in the

next time step (positive-feedback/autocatalysis). On exiting the step loop the current values/expressions of *bestcost* and *bestsoln* are output, prior to termination.

```
Begin
    • Input data
    • Set initial colony trace intensity data
    For step = 1 to maxstep
        For ant = 1 to colsize
            • Obtain a solution, soln[ant], probabilistically using colony
              trace intensity data.
            • Calculate the cost, cost[ant]
        End(For)
        • Perform local improvement
        • Determine the solution, minsoln, with the lowest cost, mincost
        If step = 1
        • Set current solution bestsoln and best cost, bestcost;
            bestsoln = minsoln
            bestcost = mincost
        Else
            If (mincost < bestcost )
                bestcost := mincost
                bestsoln := minsoln
            End(If)
        End(If)
        • Obtain trace intensity data for each ant
        • Update colony trace intensity data
    End(For)
    • Output bestcost and bestsoln
End
```

Table 3.6 Pseudo-code for the ant system algorithm.

The process, described above, of progressively improving an aid to decision-making that involves population-based information, is the most basic ACO algorithm, called ant system. Although the algorithm is simple, it has been used successfully to investigate classical NP-complete discrete optimisation problems. These problems include the quadratic assignment problem [Gambardella et al. 1999], the travelling salesman problem [Stützle and Hoss 1997, Gambardella and Dorigo 1995], graph colouring [Costa and Hertz 1997] and scheduling [Forsyth and Wren 1997].

ACO is a very recent heuristic optimisation approach to problem solving which has yet to be fully explored in terms of its development. Recent developments of ant system have concentrated on the elements of the colony trace intensity matrix and the updating process. These have lead to the following ant system variations.

- Max-min Ant System [Stützle and Hoss 1997], in which pheromone deposits (i.e. matrix elements) are restricted to lie between specified maximum and minimum values.
- Rank-based Ant System [Bullnheimer et al. 1997], in which only a specified number of ants within the colony (those with the highest-ranking cost values) deposit pheromone.

In the following section ant system is formulated in an error-correcting code context for the MDP. A developed algorithm is presented in Chapter 5.

3.7 Ant System for the MDP

The ant system described in general terms in the previous section is now formulated in an error-correcting code context for the MDP. This formulation is illustrated by the flowchart in Figure 3.3. The ant system for the MDP has several distinct phases which are now explained, with reference to Figure 3.7.

- **Input Data and Initialisation**

In this phase code details such as the generator matrix G and the length of the alpha-vectors, k (where $k = \frac{p+1}{2}$ for augmented QR codes and $k = \frac{p-1}{2}$ for expurgated QR codes) are read. Also input are ant system parameters (explained in subsequent sections) such as a and b (state probability parameters), Q (pheromone influence parameter), ρ (pheromone decay

parameter), the number of ants in the colony, x_{max}, and the maximum number of (time steps), s_{max}.

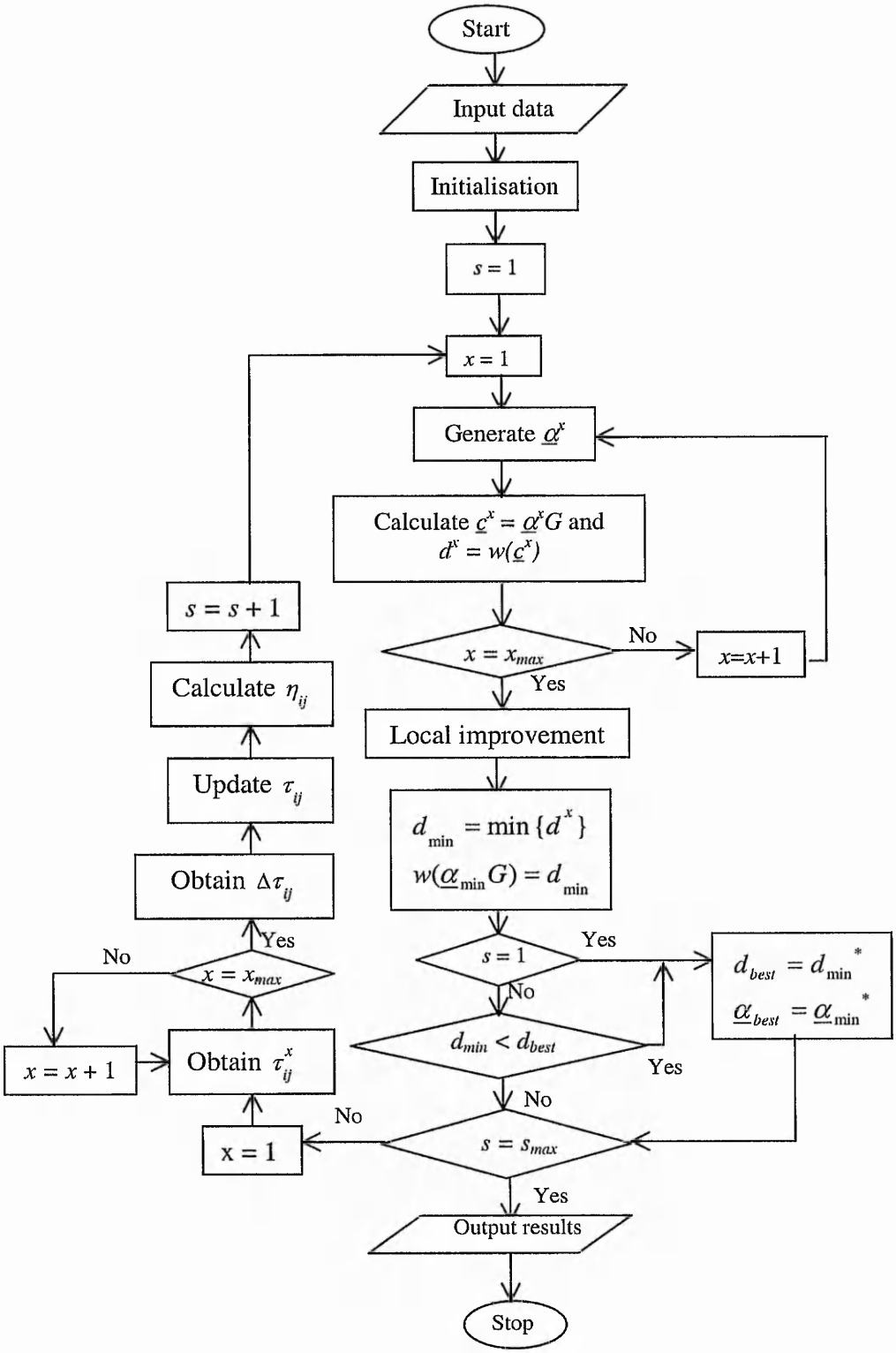


Figure 3.3 Flowchart of the ant system algorithm.

For the MDP, 'solution' and 'cost' mean alpha-vector, $\underline{\alpha}$, and its associated weight, d, respectively, where $d = w(\underline{c})$ with $\underline{c} = \underline{\alpha} G$ ($\underline{c}$ is the associated codeword).

The elements of the colony trace intensity matrix, τ_{ij}, and the trail desirability matrix, η_{ij}, are all initialised to small positive numbers. For both τ_{ij} and η_{ij} subscripts i and j denote the state and element number of an alpha-vector, $\underline{\alpha} = \{\alpha_j\}$, $j = 1,2,...,k$ (k elements), where $\alpha_j \in \{0,1\}$, so $i = 0,1$ (2 states).

- **State Probabilities and Solution Construction**

On entering the step loop, each ant, x, constructs an alpha-vector, $\underline{\alpha}^x$, $x = 1,2,...,x_{\max}$, in the following manner.

The probability of an ant selecting (constructing) a particular state i, for a particular element, j, of an alpha-vector is unknown but assumed to be directly proportional to $(\tau_{ij})^a (\eta_{ij})^b$, where a and b are user-specified input parameters. In other words, the probability of a particular (i, j) coupling depends on the amount of pheromone already on the coupling (τ_{ij}) and the 'inherent' goodness or desirability of the coupling (η_{ij}). This leads to the following equation for the probability of an ant choosing state 0 (rather than 1) in element j of an alpha-vector,

$$P_{0j} = \frac{[\tau_{0j}]^a . [\eta_{0j}]^b}{[\tau_{0j}]^a . [\eta_{0j}]^b + [\tau_{1j}]^a . [\eta_{1j}]^b} \qquad j = 1,2,...,k \tag{3.17}$$

where the denominator is a normalising term since

$$P_{0j} + P_{1j} = 1 \quad j = 1,2,...,k \tag{3.18}$$

The actual decision as to whether alpha-vector element α_j^x, $x = 1,2,...,x_{\max}$, $j = 1,2,...,k$ is 0 or 1 is made by comparing the value of P_{oj} (given by equation (3.17)) with random number $r \in [0,1]$ (a different random number is used for each element j, for each ant in each time step). For ant x and element j if $r < P_{0j}$ then $\alpha_j^x = 0$, otherwise $\alpha_j^x = 1$. In this manner each ant x in the colony, $x = 1,2,...,x_{\max}$, (probabilistically) constructs a set of alpha-vectors $\underline{\alpha}^x = \{\alpha_j^x\}$, $j = 1,2,..,k$. The associated codewords, $\underline{c}^x$, and weights, d^x are given by $\underline{c}^x = \underline{\alpha}^x G$ and $d^x = w(\underline{c}^x)$, respectively, $x = 1,2,...,x_{\max}$.

The codeword $\underline{c}^x = \underline{0}$ is forbidden so $\underline{\alpha}^x = \underline{0}$, $x \in \{1,2,...,x_{\max}\}$ may be used as a stopping condition.

- **Local Improvement**

As stated in Section 3.6, by strict analogy with real ants, this phase should have no place in the ant system. However it does have computational merit in the sense that it may improve the solution quality and convergence rate of the ant system (at the expense of execution time). The particular algorithm used for local improvement in this study is tabu search (see Section 3.3). For each ant, the current alpha-vector is used as the start alpha-vector for tabu search, with the aim of obtaining improved weights. If, for a particular ant, a weight lower than its current weight is obtained by tabu search, then both the weight and associated alpha-vector are updated.

- **Best Cost and Solution**

Whether or not local improvement is used, the lowest weight of the step, $d_{\min}$, is determined together with its associated alpha-vector, $\underline{\alpha}_{\min}$, where

$$d_{\min} = \min\{d^x\} \quad x = 1,2,...,x_{\max} \tag{3.19}$$

and $\underline{\alpha}_{\min}$ is such that $w(\underline{\alpha}_{\min} G) = d_{\min}$.

The (overall) best values for the weight, with its associated alpha-vector, codeword and step number, i.e. d_{best}, $\underline{\alpha}_{best}$, $\underline{c}_{best}$ and s_{best}, respectively are those associated with $d_{\min}$. Except in step 1 these terms may need updating if, during a particular time step, the determined value of $d_{\min}$ is lower than the current value of d_{best} (see Figure 3.3).

- **Pheromone Deposits of Individual Ants**

In this phase, the elements of the trace intensity matrices for each ant, $\delta\tau_{ij}^x$, are calculated, $x = 1,2,...,x_{\max}$. The values of the elements depend on whether or not alpha-vector element α_j^x is in state i (that is whether or not state i and element j are coupled). The values of the matrix elements may be calculated as follows,

$$\delta\tau_{ij}^x = \begin{cases} \dfrac{Q}{d^x} & \text{if } \alpha_j^x = i \\ 0 & \text{otherwise} \end{cases} \tag{3.20}$$

By equation (3.20), for ant x, some 'pheromone' is deposited only when, for particular values of i and j, $\alpha_j^x = i$ (i.e. there is an (i, j) coupling). Note that

the amount of 'pheromone' depends on the value of d^x in a monotonically decreasing manner. In other words, greater deposits are a consequence of lower values of d^x.

Hence for each ant x, its trace intensity matrix will have non-zero elements only in entries where state i and element j are coupled. Furthermore, the magnitude of these non-zero elements will be greater for those ants with lower values of d^x. Because lower values of d^x are preferred (in minimisation problems) the associated trace intensity matrix will have relatively more influence in the probabilistic construction of (good) alpha-vectors in the next time step. In equation (3.20) Q is a user-specified input parameter.

- **Autocatalysis and Trail Evaporation**

The analogy with nature in terms of autocatalysis and ant trail evaporation may be modelled computationally by the following equation (the update τ_{ij} box in Figure 3.3)

$$\tau_{ij} = \rho\tau_{ij} + \Delta\tau_{ij} \tag{3.21}$$

where the colony trace intensity matrix, $\Delta\tau_{ij}$, is the sum of the trace intensity matrices of the individual ants,

$$\Delta\tau_{ij} = \sum_{x=1}^{x_{max}} \delta\tau_{ij}^{x} \tag{3.22}$$

Equation (3.21) describes the positive-feedback (autocatalysis) on the colony trace intensity matrix; the colony trace intensity matrix for the next time step comprises (a proportion of) that of the current time step together with all the pheromone deposited in the current time step. The user-specified input

parameter ρ, where $0 < \rho < 1$, is used to model the evaporation of pheromone with the passage of time.

- **Trail Desirability**

In other studies using ACO [Dorigo and Caro 1999, Dorigo and Gambardella 1997] the trail desirability values, η_{ij}, have been fixed quantities that represent some supposed or inherent 'goodness' of particular (i, j) couplings (i.e. favoured/good couplings known, or established by an analysis of the particular problem, prior to the use of ant system).

Unfortunately, with the MDP, it is not possible to determine a priori which is the preferred state (0 or 1) for each element of an alpha-vector that will yield a low weight value.

Notwithstanding this, an attempt is made to obtain η_{ij} values (at each time step) by using the frequency of ones, σ_j, in the entry j, of the constructed alpha-vectors, $\underline{\alpha}^x = \{\alpha_j^x\}$, $x = 1,2,...,x_{\max}$,

$$\eta_{ij} = \begin{cases} \dfrac{\sigma_j + 1}{x_{\max} + 2} & \text{if i} = 1 \\ \dfrac{x_{\max} - \sigma_j + 1}{x_{\max} + 2} & \text{if i} = 0 \end{cases} \tag{3.23}$$

where

$$\sigma_j = \sum_{x=1}^{x_{\max}} \alpha_j^x \quad j = 1,2,...,k \tag{3.24}$$

In other words, η_{ij}, is a measure of the relative frequency of an (i, j) coupling in the colony and is used to represent the desirability of an (i, j) coupling. With reference to equation (3.23), the denominator is a normalising quantity. The one

in the numerator is included to cope with situations in which $\sigma_j = 0$, i.e. $\eta_{ij} = 0$ is avoided, which would adversely affect equation (3.17).

- **Output Data**

The updated values of τ_{ij} and calculated values of η_{ij} are used as the respective values in the next time step. On exiting the step loop (when $s = s_{\max}$, see Figure 3.3). The current values for d_{best}, $\underline{\alpha}_{best}$, $\underline{c}_{best}$ and s_{best} are output, prior to termination.

With reference to the ant system for the MDP outlined above and illustrated in the flowchart in Figure 3.3, the following comments are made.

- **Input Parameters**

Besides the number of ants in the colony, $x_{\max}$, and the number of time steps, $s_{\max}$, the user-specified input parameters are a, b, Q and ρ. In order to determine the values of these parameters that are most beneficial to the optimisation process for the MDP (they are problem dependent) numerical experiments are required to be performed. Furthermore, since ant system is a stochastic algorithm, once appropriate values of the input parameters have been established, further numerical experiments should be performed to investigate the repeatability of the output.

- **Stagnation and Termination**

Inappropriate values of input parameters may lead to an early occurrence of the following situations. If $\tau_{0j}^{x} >> \tau_{1j}^{x}$, $j = 1,2,...,k$ and $x \in \{1,2,...,x_{\max}\}$, then by equation (3.17), for a particular ant, $P_{0j} \sim 1$, $j = 1,2,...,k$. This may result in

the construction of $\underline{\alpha}^x = \underline{0}$, $x \in \{1,2,...,x_{\max}\}$, which will give the forbidden codeword $\underline{c} = \underline{0}$ and lead to termination.

Also, if the ratio $\frac{\tau_{1j}^x}{\tau_{0j}^x}$ takes extreme values (either very much less than 1 or very much greater than 1), $x = 1,2,...,x_{\max}$, then the values of P_{0j} and P_{1j} (see Equation (3.17)) will take extreme values (~ 0 or ~ 1) so that there may not be any variation between the constructed $\underline{\alpha}^x$, $x = 1,2,...,x_{\max}$ as time steps progress. Hence ant 'exploration' will stagnate and improved costs d and solutions $\underline{\alpha}$ may not be obtained. Under these conditions the optimisation process is in a state of 'stagnation'.

- **Time Considerations**

Besides the quality of the final minimum distance value, factors which affect the time to achieve a converged minimum distance include the number of ants in the colony and the number of ants used in the local improvement phase. Here a balance needs to be established between the number of ants needed for useful exploration (within-colony variation between the alpha-vectors) and the time the colony requires to obtain all its minimum distance values. Similar time-benefit arguments apply to the use of a local improvement phase (i.e. what proportion of the colony will be used for local improvement).

The primary stopping condition is when the specified maximum number of time steps, $s_{\max}$, is reached. The value of $s_{\max}$ should be set so that a sufficient amount of exploration is maintained while the best cost value, d_{best}, achieves convergence.

3.8 Minimum Distance Results using Ant System

The ant system algorithm described in Section 3.6 was used to determine the minimum distances of the same QR codes investigated in Section 3.4 using tabu search; those with generator matrices with weight $\frac{p-1}{2}$, augmented QR codes ($n(x)$ and $q(x)$, see Chapter 2) and expurgated QR codes ($(x-1)n(x)$ and $(x-1)q(x)$, see Chapter 2).

The initial trail and desirability matrices for the colony were set to $\tau_{ij}(0) = 10^{-6}$ and $\eta_{ij}(0) = 0.5$, respectively, $i = 1,2,...,k$ and $j = 0,1$. The maximum number of time steps was $s_{\max} = 500$ and the number of ants in the colony was linked to the size of the problem; $x_{\max} = k$.

First the values of the user-specified parameters (a, b, ρ, Q) were investigated to obtain the combination most beneficial to the optimisation process. The augmented QR code ($n(x)$) with $p = 137$ (code 6, see Tables 3.3 and 3.4) was used as a test code together with the following parameter values, a = {1, 2, 3}, b = {0, 1, 2}, ρ = {0.7, 0.8, 0.9} and Q = {1, 10}. For each combination of (a, b, ρ, Q) minimum distance results using the AS algorithm were obtained with 10 different seeds for the random number generator (random number $r \in [0,1]$ is compared with P_{0j} given by equation (3.17)).

Analysis of the minimum distance results gave the best combination (in terms of the obtained solution quality and reproducibility) as (1, 0, 0.7, 10). This combination of parameter values was then used by the ant system algorithm to obtain minimum distances for all other QR codes. As with the test

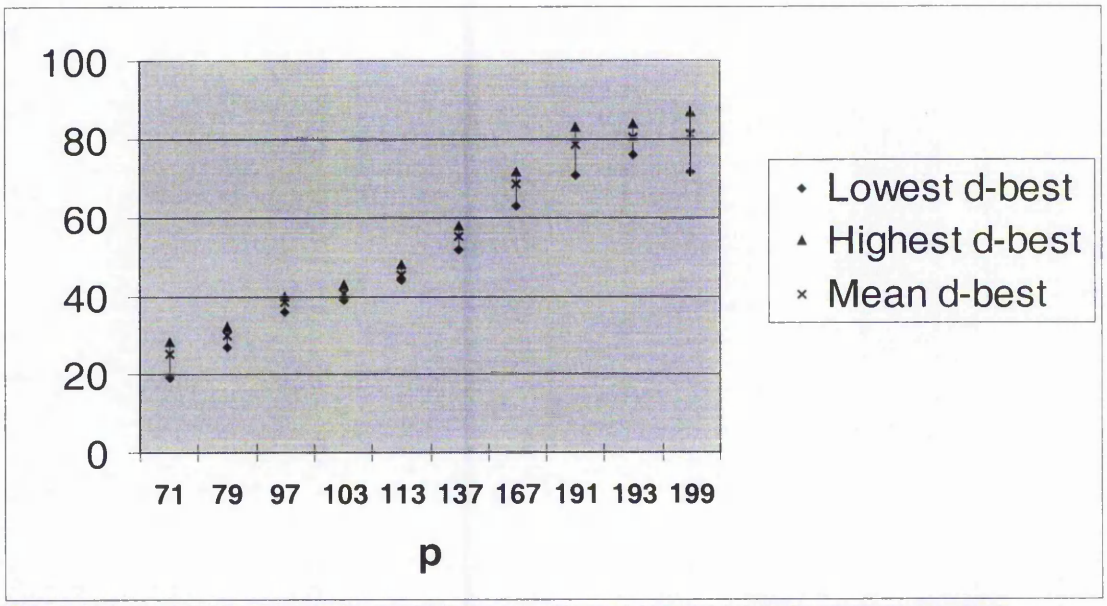
code, the ant system algorithm was used with 10 different random number seeds for each code (i.e. 10 runs per code).

The combination of user-specified parameters, i.e. (1, 0, 0.7, 10), were such that the optimisation process did not suffer from stagnation [Maniezzo et al 1994] and exploration (i.e. generation of a variety of alpha-vectors) continued for the duration of the runs. As a consequence the minimum distance values were, in general, obtained late in the optimisation process with relatively long execution times (compared to tabu search). The best values of the trail intensity and desirability indices were $a = 1$ and $b = 0$, respectively, which implies that the trail desirability, as formulated in this study, has no (beneficial) influence on the optimisation process. The pheromone evaporation parameter ρ was such that low values reduced the efficiency of the algorithm, i.e. longer execution times were required to obtain good solutions; $\rho = 0.7$ gave the best results in reasonable time. $Q = 10$ was the better value tested; a greater range of pheromone deposit values (see equation (3.20)) is possible, which enhances exploration, compared to the case in which $Q = 1$.

Results for QR codes with $w(G) = \frac{p-1}{2}$ and augmented ($n(x)$) QR codes are given in Figure 3.7 and Figure 3.8, respectively. Results for the other QR codes; augmented ($q(x)$) and expurgated ($(x-1)\,n(x)$ and $(x-1)\,q(x)$) are given in Figures A1, A2 and A3, respectively, in Appendix A.

With reference to Figures 3.7 and 3.8, for each code (characterised by prime number p) d_{QR} denotes the known minimum distance (or bounds), and $\bar{d}_{best}$,

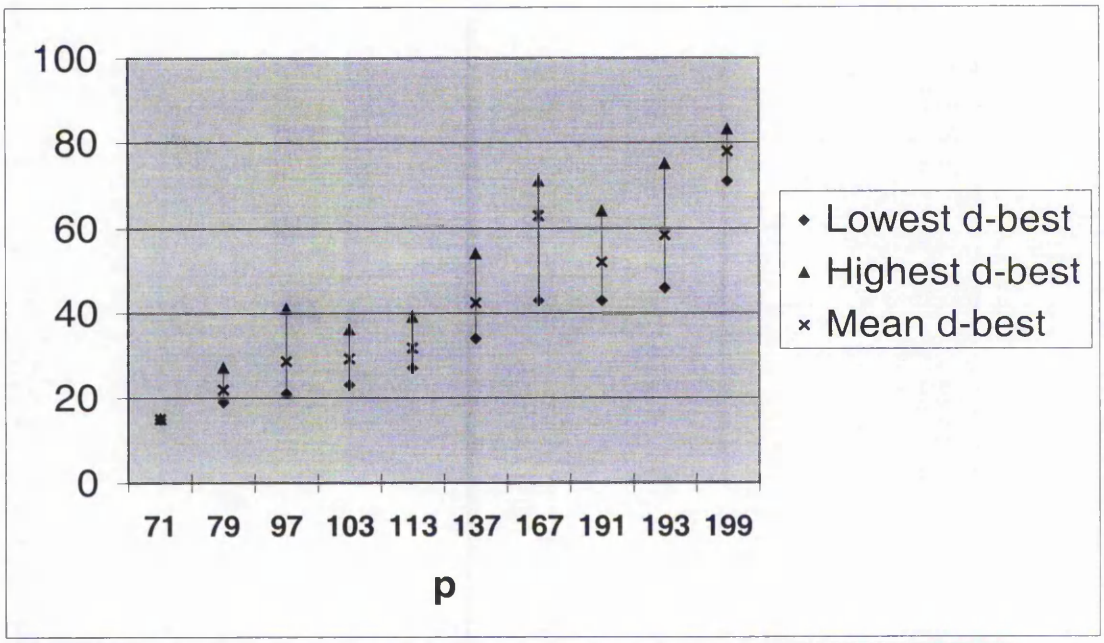
d^{+}_{best} and $\overline{d}_{best}$ denote the lowest, highest and mean value of the minimum distances obtained by the 10 runs for each code.



p	71	79	97	103	113	137	167	191	193	199
d_{QR}	11	15	15	19	11-15	13-21	15-23	15-27	15-27	15-31
d^{-}_{best}	19	27	36	39	44	52	63	71	78	72
d^{+}_{best}	28	32	40	43	48	58	72	83	84	87
$\overline{d}_{best}$	25.3	29.9	38.6	40.7	45.6	55.2	68.9	79	80.6	81.3

Figure 3.7 Minimum distances obtained by ant system using $w(G) = \frac{p-1}{2}$.

Inspection of the table in Figure 3.7 reveals that QR codes with $w(G) = \frac{p-1}{2}$ give poor values of d^{-}_{best}. In comparison the d^{-}_{best} values for the augmented QR codes ($n(x)$), shown in the table of Figure 3.8, are of reasonable quality for the smaller codes, although the quality degrades as the size of the code (characterised by the value of p) increases.



p	71	79	97	103	113	137	167	191	193	199
d_{QR}	11	15	15	19	11-15	13-21	15-23	15-27	15-27	15-31
d^{-}_{best}	15	19	21	23	27	34	43	43	46	71
d^{+}_{best}	15	27	41	36	39	54	71	64	75	83
$\bar{d}_{best}$	15	21.9	28.6	29.2	31.8	42.5	63	52.1	58.5	78

Figure 3.8 Minimum distances obtained by ant system using *n(x)*.

The ant system algorithm is stochastic and the graphs in Figures 3.7 and 3.8 illustrate, for each code, the variation in the minimum distances (d_{best}) obtained by the 10 runs.

A summary of all the obtained results for QR codes using the ant system algorithm is given in Table 3.9, which contains the best minimum distances (i.e. lowest d^{-}_{best} values) found for each value of p, together with the associated time step numbers and execution times, s^{-}_{best} and t^{-}_{best}, respectively.

Inspection of Table 3.5 (tabu search) and Table 3.9 (ant system) reveals, for each value of p that the basic tabu search algorithm produced better quality

results in shorter execution times, compared to those obtained using the basic ant system algorithm.

Code	p	$g(x)$	d_{QR}	$\bar{d}_{best}$	$\bar{s}_{best}$	$\bar{t}_{best}$ $(h:m:s)$
1	71	$n(x)$	11	15	147	0:00:16.20
1	71	$q(x)$	11	15	111	0:02:02.60
2	79	$(x-1)n(x)$	16	16	234	0:01:08.50
2	79	$(x-1)q(x)$	16	16	357	0:02:07.76
3	97	$q(x)$	15	18	465	0:05:02.91
4	103	$n(x)$	19	23	476	0:01:38.26
4	103	$q(x)$	19	23	500	0:04:42.54
5	113	$q(x)$	11-15	26	371	0:03:22.51
6	137	$q(x)$	13-21	30	461	0:05:03.19
6	137	$(x-1)q(x)$	14-22	30	464	0:07:28.80
7	167	$(x-1)q(x)$	16-24	36	446	0:09:35.67
8	191	$(x-1)n(x)$	16-28	36	450	0:06:15.47
9	193	$(x-1)n(x)$	16-28	40	441	0:11:46.84
10	199	$(x-1)n(x)$	16-32	52	500	0:14:54.63
10	199	$(x-1)q(x)$	16-32	52	483	0:07:22.37

Table 3.9 Minimum distances found by ant system using different generator polynomials.

In order to improve the quality of the ant system results (at the expense of execution time) a tabu search local improvement phase (see Section 3.7) was included in the basic ant system algorithm to give an algorithm denoted by AS(TS). The number of ants used in the local improvement phase was (the integer part of) $\frac{p+1}{10}$ (i.e. 20% of k) and the maximum number of tabu search moves was $m_{\max}$ = 100 per ant. In an attempt to show directly the influence of the inclusion of the local improvement phase, the AS(TS) algorithm was used with the same codes shown in Table 3.9. The minimum distances using AS(TS), $\bar{d}_{best}$, and associated time step numbers and execution time, $\bar{s}_{best}$ and $\bar{t}_{best}$, respectively, are given in Table 3.10. Comparison of the minimum

distance results in Tables 3.9 and 3.10 reveals, as anticipated, local search improves solution quality at the expense of execution time.

Code	p	$g(x)$	d_{QR}	$\bar{d}_{best}$	$\bar{s}_{best}$	$\bar{t}_{best}$ $(h{:}m{:}s)$
1	71	*n(x)*	11	11	1	0:00:55.20
1	71	*q(x)*	11	12	1	0:00:05.17
2	79	*(x-1)n(x)*	16	16	2	0:00:58.55
2	79	*(x-1)q(x)*	16	16	2	0:00:38.61
3	97	*q(x)*	15	16	7	0:02:08.49
4	103	*n(x)*	19	19	20	0:22:06.12
4	103	*q(x)*	19	20	9	0:02:15.07
5	113	*q(x)*	11-15	20	13	0:04:19.16
6	137	*q(x)*	13-21	30	13	0:07:47.17
6	137	*(x-1)q(x)*	14-22	26	18	0:35:48.63
7	167	*(x-1)q(x)*	16-24	36	17	2:21:36.10
8	191	*(x-1)n(x)*	16-28	44	9	2:09:21.28
9	193	*(x-1)n(x)*	16-28	40	17	3:24:02.80
10	199	*(x-1)n(x)*	16-32	52	43	6:47:03.80
10	199	*(x-1)q(x)*	16-32	48	15	2:08:30.66

Table 3.10 Minimum distances found by AS(TS) using different generator polynomials.

Chapter 4

Tabu Search for Minimum Distances

4.1 Introduction

In Chapter 3, although the minimum distances found using a basic implementation of tabu search were of acceptable quality, some additional features are now included in the algorithm with the aim of improving the solution quality. In Chapter 3, the elements of the tabu list are alpha-vectors which keep track of the search in binary space, but this type of tabu list is time-consuming to manage, especially when larger codes are examined. In this chapter a two-way conversion is introduced that converts alpha-vectors to integers with the aim of reducing the size of the list and improving the search efficiency.

In many optimisation problems, extra moves beyond local optima are often necessary (solution technique permitting) in order to improve the current best solution. The short-term memory (basic) tabu search method described in Chapter 3 allows moves beyond local optima but it may not be effective enough to guide the search for long periods. Hence the use of candidate list strategies [Glover and Laguna 1997], which enable the search to memorise certain important solutions for longer periods so that the optimisation process does not require the excessive memory that would be needed to store the complete search history. For finding minimum distances (i.e. the MDP), the influential candidate list is introduced in this chapter to identify 'influential' solutions, which may be used later on in the optimisation process. An interesting aspect of BCH and QR codes are the distance bounds [Pless 1989]. In this chapter the bounds are utilised as threshold values to measure the quality of the search.

Finally, the tabu search framework may also comprise intermediate and long-term memory, for diversification [Kelly et al. 1994, Skorin-Kapov 1994a, Glover 1990] and intensification [Glover 1990] strategies, in order to seek continually superior minimum distances. Intensification aims to examine 'promising' regions thoroughly whereas diversification drives the search into new regions. Use of the intensification and diversification strategies of this study is based on the distance bounds of BCH and QR codes and influential candidates are used as start solutions at different stages of the search. For intensification, the influential candidate list has a 'backtracking' effect in which the search returns to desirable regions and, for diversification, the influential candidate is treated as a 'penalty' to force the search into new regions. These search strategies enable the optimisation process to explore solution-space and exploit superior solutions.

4.2 Overview of the Developed Tabu Search Algorithm

The basic tabu search algorithm of Chapter 3 is developed to include a number of features (described in detail in Sections 4.3 to 4.8) designed to improve its capability of obtaining lower minimum distance values. The developed algorithm, in overview, is illustrated in the flowchart in Figure 4.1, which gives the strategic features.

With reference to the flowchart, typical input data includes code details such as the generator matrix G and (for QR codes) prime number p, together with search details such as the start solution $\underline{z}_0$ (where $\underline{z}_0$ is an integer characterisation of the start alpha-vector, $\underline{\alpha}_0$, explained in Section 4.3), the number of moves comprising a search phase, m_{phase} (the entire search comprises a number of phases) and the overall number of search moves, $m_{\max}$. The algorithm utilises a number of lists; a tabu list, an influential candidate list and lists to record the usage of particular 'solutions'.

Details of these lists and how they are managed are given in Sections 4.4 and 4.6, respectively.

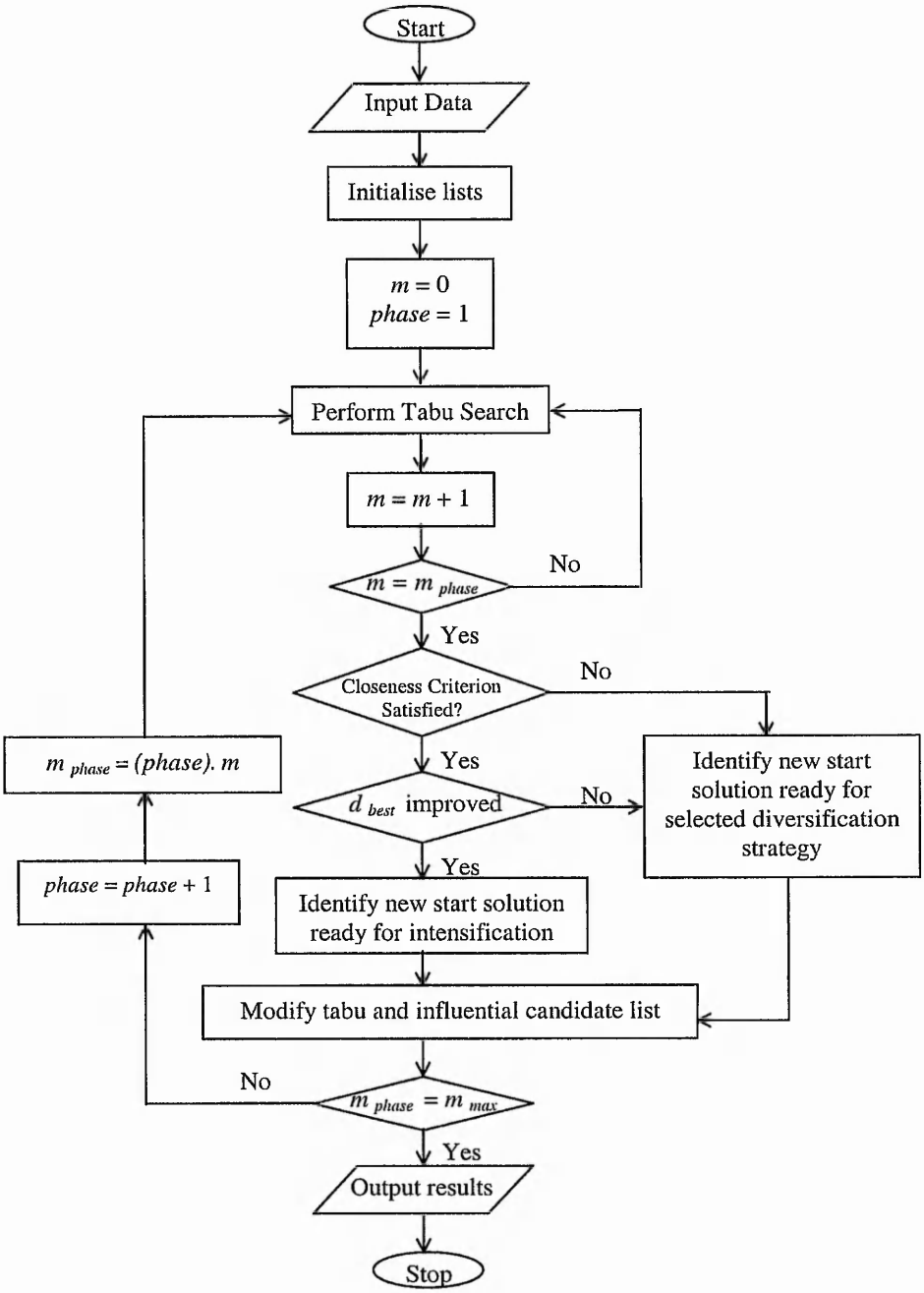


Figure 4.1 Flowchart for the developed tabu search algorithm.

All lists are initially empty except the tabu list which contains $\underline{0}$ and $\underline{z}_0$. At this stage the overall best (i.e. lowest) minimum distance is $d_{best} = w(\underline{c}_{best})$ where codeword $\underline{c}_{best} = \underline{\alpha}_{best} G$ and alpha-vector $\underline{\alpha}_{best} = \underline{\alpha}_0$. The best move is $m_{best} = 0$.

On entering the tabu search move loop, *phase* = 1 and $m = 0$. Tabu search is then performed until the current move number, m, equals the current value of m_{phase}. On performing a tabu search move a transition in solution-space is made from the current solution to $\underline{z}_{m+1}$ (the integer characterisation of $\underline{\alpha}_{m+1}$, see Section 4.3) where the current solution is either a phase start solution or $\underline{z}_m$ $(1 \le m \le m_{\max} - 1)$. If necessary the values/expressions of d_{best}, $\underline{c}_{best}$, $\underline{\alpha}_{best}$ and m_{best} are updated.

Next two decision points are encountered. If both the closeness criterion (explained in Section 4.5) is satisfied and the value of d_{best} has reduced during the latest phase of moves, then a new start solution is identified ready for the next phase of tabu search moves. Conceptually this will be an intensification phase (see Section 4.7). Alternatively, if the closeness criterion is not satisfied or d_{best} has not reduced then other start solutions are identified for the next search phase. This will be one of the diversification strategies described in Section 4.8. Once a new start solution has been established the algorithm lists (tabu and the influential candidate list) are updated. Then, if the current value of m_{phase} equals $m_{\max}$, the current values of d_{best}, $\underline{c}_{best}$, $\underline{\alpha}_{best}$ and m_{best} are output, prior to termination. If not, then the phase count is incremented and the value of m_{phase} is increased by a multiple of *phase*, prior to entering the move (m) loop and actually performing the next phase of tabu search moves (i.e. either intensification or diversification).

4.3 Two-way Conversion

In Chapter 3, elements stored in the tabu list are alpha-vectors (i.e. strings of binary numbers which may need large storage requirements). Although technology is available for large computer storage, it is highly desirable if the storage requirement of a search can be kept as low as reasonably possible. One method of achieving this is to use hash functions [Glover and Laguna 1997], which may be used to manage tabu lists in a computationally inexpensive way. The method of reducing storage requirements introduced in this chapter has the same broad objective as that of hash functions but is significantly different in its form and ability.

In this chapter a two-way conversion mechanism is used to achieve reduced storage requirements for tabu list elements. The conversion mechanism is such that alpha-vectors may be converted to (blocks of) integers, and vice versa. Compared to hash functions the main differences and similarities are as follows.

- Differences

1. A hash function approach has a random feature in the sense that a string of data (i.e. binary data) could 'hash' to the same integer, although the probability of such a collision is low. The two-way conversion approach (explained below) is one-to-one from alpha-vectors to integers and vice versa.
2. With the hash function approach, once 'hashing' to an integer has taken place the original data cannot be retrieved. This is possible with the two-way conversion approach.

- Similarities

1. The hash function and two-way conversion approaches are both easy to implement.
2. Both approaches lead to reduced storage requirements (compared to storing alpha-vectors).

In Chapter 3 the tabu list elements are alpha-vectors, $\underline{\alpha} = \{\alpha_i\}$, $i = 1,2,...,k$, where $\alpha_i \in \{0,1\}$. Computationally, each alpha-vector requires $2k$ bytes (2 bytes per bit). Since the smallest QR code investigated in this study is such that $k = 36$ ($k = \frac{p+1}{2}$ with $p = 71$) the minimum requirement is therefore 72 bytes. In view of this it is impossible to have a (simple) one-to-one correspondence from an alpha-vector to an integer because (for computers with a 16-bit data type) the maximum integer is $2^{31} - 1$, which is equivalent to 4 bytes of memory. Because of this upper limit on the number of bytes available to represent integers, the two-way conversion approach divides an alpha-vector into a series of blocks of binary strings and each block is individually converted into an integer (no greater than $2^{31} - 1$). The number of blocks, B, is given by,

$$B = \left\lfloor \frac{k}{31} \right\rfloor \tag{4.1}$$

where the square parentheses denote integer part only. The converted alpha-vector consists of a series of blocks (B blocks) of integers z_i, $i = 1,2,...,B$ which comprise the integer 'vector' $\underline{z}$;

$$\underline{z} = \{ z_1, z_2, ..., z_B \} \tag{4.2}$$

Where

$$z_i \in \{0,1,...,2^{31}-1\}\ ,\quad i=1,2,...,B-1 \tag{4.3}$$

$$z_B \in \{0,1,...,k \mod 31\} \tag{4.4}$$

The value of the integer z_i, $i=1,2,...,B$ is given by,

$$z_i = \sum_{j=1}^{g}(\alpha_j \cdot 2^{j-1})\ ,\quad g = \begin{cases} 31 & i=1,2,...,B\text{-}1 \\ k \mod 31 & i=B \end{cases} \tag{4.5}$$

As an example, consider prime number p = 199 (the highest value used in this study) so that $k=100$. Furthermore, consider the following alpha-vector, $\underline{\alpha}$, to correspond to a QR codeword (the number of blocks, by equation (4.1), is $B=4$),

$$\begin{aligned} \underline{\alpha} = \{&0010010 \ldots \text{all zeros} \ldots.0 && \text{block 1, 31 elements} \\ &1101100 \ldots \text{all zeros} \ldots.0 && \text{block 2, 31 elements} \\ &0110010 \ldots \text{all zeros} \ldots.0 && \text{block 3, 31 elements} \\ &1001100\} && \text{block 4, } k \bmod 31 \text{ elements} \end{aligned} \tag{4.6}$$

Then, using equation (4.5), $z_1 = 2^2 + 2^5 = 36$, similarly for z_2, z_3 and z_4 so that, by equation (4.2),

$$\underline{z} = \{36, 27, 37, 25\} \tag{4.7}$$

Note that the storage requirement for $\underline{\alpha}$ is 200 bytes ($2k$ bytes) whereas that for $\underline{z}$ is 16 bytes (4 bytes per block).

To convert each integer block $z_i \in \underline{z}$, $i=1,2,...,B$, back to form an assembled alpha-vector, $\underline{\alpha}$, where

$$\underline{\alpha} = \overset{B}{\underset{j=1}{C}}\{\alpha_i\}_j \ , \quad i = \begin{cases} 1,2,...,31 & \text{if } j < B \\ 1,2,...,k \bmod 31 & \text{if } j = B \end{cases} \tag{4.8}$$

(C denotes concatenation of the blocks of binary strings, $\{\alpha_i\}_j$, $j = 1,2,...,B$), requires the remainder on repeated (g times) division by 2 of each integer block; $g = 31$ if $j < B$ and $g = k \bmod 31$ if $j = B$.

The example with p = 199 has shown that an order of magnitude reduction in storage may be obtained by using the two-way conversion mechanism, which, together with its inverse ability, illustrates the merits of this approach.

4.4 The Influential Candidate List

Although in some applications [Bland and Baylis 1995, Bland and Dawson 1991], tabu search comprising short-term memory only (i.e. a tabu list comprising a relatively few, and transitory, elements) has produced relatively good results, the search becomes more powerful when longer term memory is included in the algorithm. Therefore, in this chapter, another form of tabu list is introduced, in which 'important' elements that may influence the search process are permanently (i.e. long-term) remembered. This (long-term memory) list, which will be called the influential candidate list (ICL), contains integer characterisations of 'important' (or 'influential') alpha-vectors for possible use at different stages of the optimisation process. Also, use of this list enables the (intermediate-term) tabu list to remain of reasonable size (see Section 4.6).

During each phase of the optimisation process the ICL stores the integer vectors $\underline{z}$ associated with minimum distance values d that are less than or equal to the current overall minimum distance, d_{best}, that is

$$\underline{z}_i \in \{ICL\} \iff d_i \leq d_{best} \quad , \quad i \in \{1,2,...,m_{\max}\} \tag{4.9}$$

The ICL is used as a means of remembering candidates that may be influential on the search in the longer term. In this study the phase start candidates (i.e. integer vectors, $\underline{z}$) for intensification and diversification (see Sections 4.7 and 4.8, respectively) are selected from this list. For intensification an influential candidate (IC) works in a similar manner to an 'elite' candidate [Glover 1997], that is, good solutions are re-used to thoroughly examine a particular 'promising' region of solution-space. In this study an IC may also be used for diversification purposes to drive the search away from the current region. In other words, the IC concept is a generalisation of that of an elite candidate.

To complement the use of the ICL, other lists are used to record the frequency count and usability-state of the members of the ICL. Each IC has an associated frequency count whose value (initially 1) is increased by 1 if the IC is used as a phase start solution or if a non-tabu neighbourhood best solution is in the ICL. Usually, a move would be made to this particular solution since it is no longer in the tabu list (hence not tabu by the intermediate-term memory). However, since it is an IC, this means that the search has previously visited this solution, which has, at some stage, left the tabu list. The implication is that the search has returned to a previously explored region of solution-space and there may be a danger of cycling; an IC with an associated high frequency count would indicate this danger.

The usability-state associated with an IC is a boolean quantity (initially 0) which becomes 1 permanently when the IC is used as a phase start solution. In general, an IC with usability-state 1 is forbidden to be re-used as a phase start solution (those with usability-state 0 are still eligible) unless all influential candidates have usability-state

(see random-start diversification in Section 4.8). The frequency count and usability-state associated with an IC are utilised in the diversification strategies (see Section 4.8).

The process of making a single tabu search move, from $\underline{z}$ (the current integer vector) to $\underline{z}_{\min}$ (the non-tabu neighbourhood best integer vector), is given in the pseudo-code in Figure 4.2.

Begin

- Obtain the neighbourhood, $N(\underline{z})$, of current integer vector $\underline{z}$ with associated current alpha-vector $\underline{\alpha}$,

$$N(\underline{\alpha}) \leftrightarrow N(\underline{z})$$

- Evaluate the weight of each neighbourhood codeword,

$$d_{nbhd} = w(\underline{\alpha}_{nbhd} G), \text{ where } \underline{\alpha}_{nbhd} \in N(\underline{\alpha})$$

do

- Identify the non-tabu neighbourhood integer vector, $\underline{z}_{\min}$, with the lowest weight, $d_{\min}$, where $d_{\min} \in \{d_{nbhd}\}$

If $\underline{z}_{\min} \notin T$

If $d_{\min} \leq d_{best}$

- $\underline{z}_{\min}$ enters both T and I
- f for $\underline{z}_{\min}$ initialised to 1, $f \in F$
- u for $\underline{z}_{\min}$ initialised to 0, $u \in U$

Else

- $\underline{z}_{\min}$ enters T only

End (If)

Else

- $\underline{z}_{\min}$ given tabu status but does not enter T
- f for $\underline{z}_{\min}$ increased by 1

End (If)

While ($\underline{z}_{\min} \notin T$).

- Search moves from $\underline{z}$ to $\underline{z}_{\min}$

End

Figure 4.2 Pseudo-code for a tabu search move.

In the pseudo-code f and u denote the frequency count and usability-state, respectively, associated with $\underline{z}_{\min}$ (when $\underline{z}_{\min}$ is an IC) and the sets T, I, F and U denote the tabu (intermediate-term) memory, influential candidate (long-term)

memory, frequency count and usability-state lists, respectively. Note that when $\underline{z}_{\min}$ $\notin T$ but $\underline{z}_{\min} \in I$, then this particular $\underline{z}_{\min}$ has been previously visited (since $\underline{z}_{\min} \in I$) and has 'passed through' T (since $\underline{z}_{\min} \notin T$). In this case, the frequency count associated with $\underline{z}_{\min}$ is increased by 1 but a move to this particular $\underline{z}_{\min}$ is not made (hence it does not enter T), rather, $\underline{z}_{\min}$ is updated to represent the non-tabu neighbourhood integer with the next lowest weight. A possible move from $\underline{z}$ to (the updated) $\underline{z}_{\min}$ is then investigated.

4.5 The Closeness Criterion

With QR codes minimum distance estimates may be obtained using square root bounds [MacWilliams 1977] as explained in Chapter 2. In this chapter use is made of these bounds to aid decisions concerning the nature of the optimisation process (i.e. intensification or diversification) during each phase of the search beyond the initial m_{phase} moves. The decision depends on whether or not the 'closeness criterion' is satisfied. This is a relationship in which the distance ratio $\frac{d_{bound}}{d_{best}}$ is matched against a specified threshold value, D, where $0 < D < 1$. That is, the closeness criterion is satisfied when

$$\frac{d_{bound}}{d_{best}} > D \tag{4.10}$$

Here d_{bound} is the (square root bound) minimum distance estimate for a particular QR code and d_{best} is the current overall lowest minimum distance value obtained during the optimisation process.

As seen in inequality (4.10) the closeness criterion is a measure of the closeness of d_{best} to d_{bound}. In this study $D = 0.6$ was found to provide a 'balanced' decision between intensification when (inequality (4.10) is true) and a diversification strategy (when false); if D is set too high then this may be too restrictive and solutions that belong to 'promising' regions may not be identified. Alternatively, if D is set too low then search effort may be wasted examining regions that may not contain 'elite' solutions.

4.6 Dynamic Tabu List Management

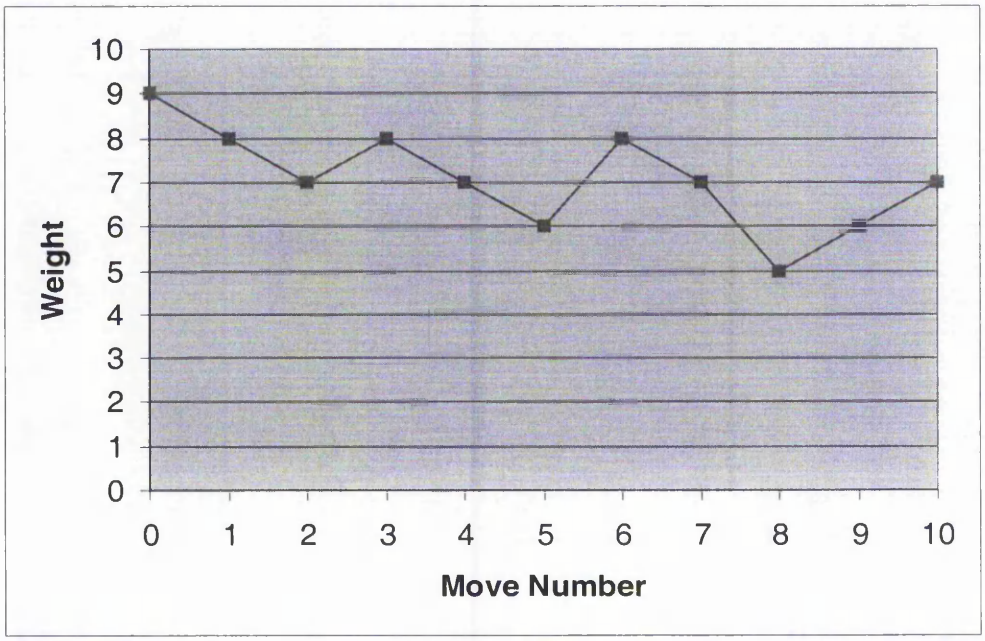
As stated in Chapter 3 the basic roles of a tabu list are to enable the search to escape from local optima and to help avoid cycling in solution-space. General ideas concerning the management of the tabu list have been proposed by Glover [Glover 1990] and adopted by other researchers [Chiang and Kouvelis 1996, Taillard 1991]. For example, Chiang and Kouvelis (1996) use a tabu list whose size varies (i.e. the list 'oscillates') between specified upper and lower bounds as the search progresses. Skorin-Kapov (1994) uses a tabu list in which certain members lose and then regain their tabu status (i.e. the list has 'gaps') as the search progresses.

The difficulty with the 'oscillating lists' approach is that the values of (pre-specified) upper and lower bounds are not obvious, in general. Also, the 'list with gaps' approach does not reduce storage requirements since the tabu-de-activated members are still stored, ready for re-activation at a later stage.

In this study the tabu list is dynamic in the sense that its size may vary as the search progresses. Furthermore no bounds are specified; the tabu list size is determined in an automatic manner by the optimisation process of each phase of the search. Also, members of the tabu list that loose their tabu status are cleared from the

tabu list (although some may be retained in the ICL for possible future use). This reduces storage requirements and increases search efficiency.

In this study the elements of the tabu list (and ICL) are integer vector characterisations (explained in Section 4.3) of alpha-vectors (as used in Chapter 3). To illustrate the management of the tabu list part of a hypothetical search (with phases comprising 10 moves) is presented in Figure 4.3, the graph points indicate the search history (i.e. the accepted neighbourhood lowest weight at each move, d_i, $i = 1,2,...,10$) for the initial phase of 10 moves. The tabu list contains the initial elements $\underline{0}$ and $\underline{z}_0$, where the start integer vector $\underline{z}_0$ has weight d_0, and, initially the current overall lowest weight, d_{best}, is such that $d_{best} = d_0 = 9$.



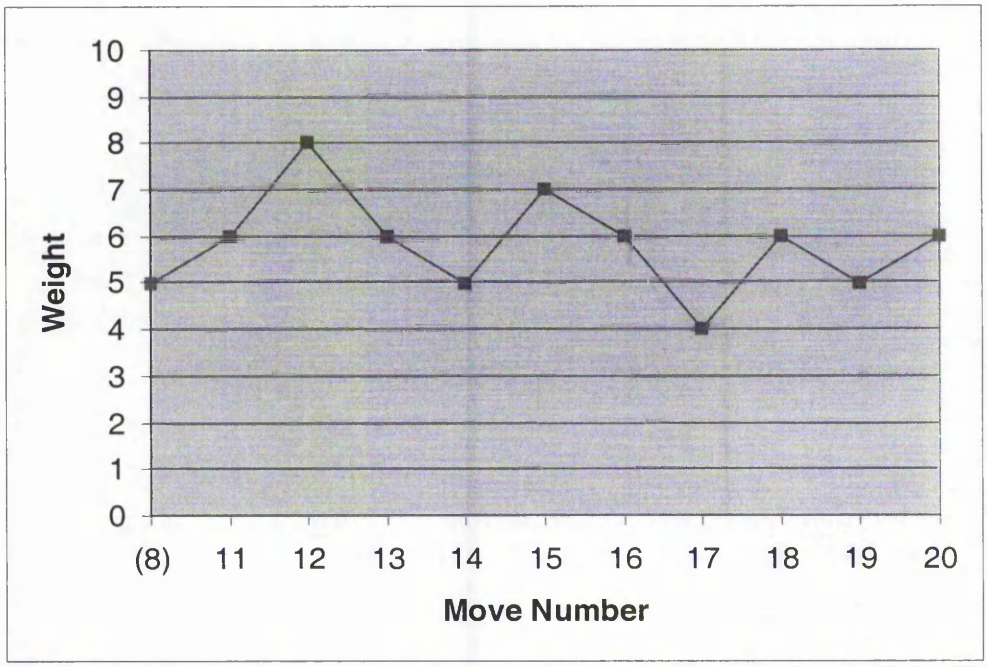
T
$\underline{0}$
$\underline{z}_0$
$\underline{z}_1$
:
:
:

I	F	U
$\underline{z}_1$	1	0
$\underline{z}_2$	1	0
$\underline{z}_4$	1	0
$\underline{z}_5$	1	0
$\underline{z}_8$	1	0

Figure 4.3 Search process after phase 1 (moves 1 to 10).

The integer vectors associated with the search history weights are $\underline{z}_i$, $i = 1,2,...,10$, which all enter the (intermediate-term memory) tabu list, T. With reference to the graph in Figure 4.3, notice that $d_i \leq d_{best}$, $i \in \{1,2,...,10\}$ at moves 1, 2, 4, 5 and 8; the associated integer vectors; $\underline{z}_1$, $\underline{z}_2$, $\underline{z}_4$, $\underline{z}_5$ and $\underline{z}_8$ are therefore influential candidates and enter the (long-term memory) ICL, set *I*. For each member of *I*, the frequency count and usability-state, $f_i \in F$ and $u_i \in U$, respectively, $i = 1, 2, 4, 5$ and 8, are initialised to 1 and 0, respectively.

The search history for the second phase of moves is shown in Figure 4.4. For illustration purposes the start integer vector is taken to be the most recent IC, $\underline{z}_8$. The tabu list is modified to contain the permanent member $\underline{0}$, the tabu list members of the preceding phase from $\underline{z}_{s-1}$, where $\underline{z}_s$ is the selected start integer vector (here $s = 8$), together with current phase tabu elements, $\underline{z}_i$, $i =$ 11, 12,...,20. The selection of the phase start integer vector, $\underline{z}_s$, depends on the nature of the phase optimisation process and is explained in Sections 4.7 and 4.8. It is noted that elements $\underline{z}_1$ to $\underline{z}_6$ have now left T and are no longer tabu. With reference to Figure 4.4, at the start of the second phase $d_{best} = d_8 = 5$, and $d_i \leq d_{best}$, $i \in \{11,12,...,20\}$ for i = 14 and 17. Hence $\underline{z}_{14}$ and $\underline{z}_{17}$ enter I. Also, to record that $\underline{z}_8$ has been used as a phase start solution, $f_8 = f_8 + 1 = 2$ and $u_8 = 1$. The retention of elements $\underline{z}_7$ to $\underline{z}_{10}$ in T preclude the search performing a back-track move from $\underline{z}_8$ and retracing a previous forward path from $\underline{z}_8$.

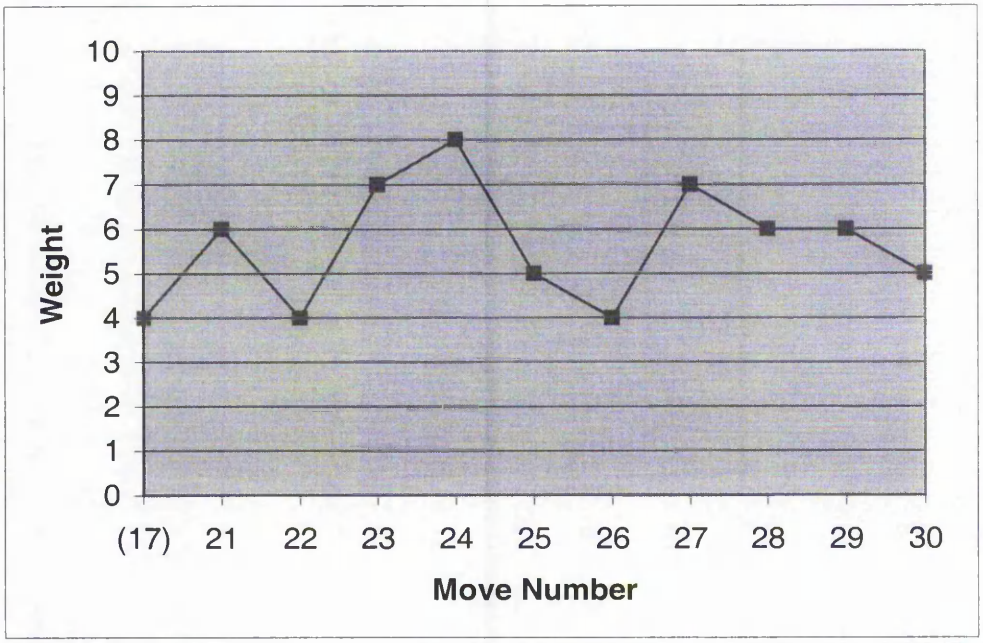


T
$\underline{0}$
$\underline{z}_7$
$\underline{z}_8$
:
$\underline{z}_{10}$
$\underline{z}_{11}$
:
$\underline{z}_{20}$

I	F	U
$\underline{z}_1$	1	0
$\underline{z}_2$	1	0
$\underline{z}_4$	1	0
$\underline{z}_5$	1	0
$\underline{z}_8$	2	1
$\underline{z}_{14}$	1	0
$\underline{z}_{17}$	1	0

Figure 4.4 Search process after phase 2 (moves 11 to 20).

As a final illustration phase, the various lists for moves 21 to 30 are shown in Figure 4.5, on the assumption that this phase started from $\underline{z}_{17}$ with integer vectors $\underline{z}_i$ such that $d_i \leq d_{best}$ for $i = 22$ and 26. Hence $f_{17} = f_{17} + 1 = 2$ and $u_{17} = 1$. Note that this example indicates that $\underline{z}_4$ (no longer in T) has been revisited but not accepted during this phase; $f_4 = f_4 + 1 = 2$.



T
$\underline{0}$
$\underline{z}_{16}$
$\underline{z}_{17}$
:
:
$\underline{z}_{20}$
$\underline{z}_{21}$
:
$\underline{z}_{30}$

I	F	U
$\underline{z}_1$	1	0
$\underline{z}_2$	1	0
$\underline{z}_4$	2	0
$\underline{z}_5$	1	0
$\underline{z}_8$	2	1
$\underline{z}_{14}$	1	0
$\underline{z}_{17}$	2	1
$\underline{z}_{22}$	1	0
$\underline{z}_{26}$	1	0

Figure 4.5 Search process after phase 3 (moves 21 to 30).

In general the tabu list operates as an intermediate-term memory, however, if the particular optimisation phase is 'random-start diversification' (explained in Section 4.8) then the phase start integer vector, denoted by $\underline{z}_r$, is obtained in a random manner. In this case the tabu list is cleared of all elements except $\underline{0}$ and $\underline{z}_r$, and then grows throughout this phase of the search. The tabu list is cleared because the search 'jumps' to a new region of solution-space and so previous tabu elements are no longer required. This process is repeated for all subsequent random-start diversification phases. In these cases the tabu list acts as a short-term memory.

4.7 Intensification

An intensification strategy is used within tabu search for the purpose of exploring 'attractive' regions of solution-space thoroughly. The measure of 'attractiveness' is problem dependent; low distance value is used in this study.

Intensification has been used with tabu search in the investigation of various problems [Ben-Daya and Al-Fawzan 1998, Hertz et al. 1997]. For example, some applications use intensification based on a measure of improvement in the solutions as the search progresses [Chiang and Kouvelis 1996] while others are based on the search returning to previous 'good' solutions for closer investigation [Glover 1990]. In this study an intensification phase is based on both solution improvement and previously found solutions. Also, use is made of the minimum distance bounds for QR codes, which may be considered as 'target' values.

With reference to the flowchart in Figure 4.1, intensification will be performed after the initial phase of tabu search moves (upto move $m = m_{phase}$, where *phase* is the input interval value), provided the minimum distance obtained in the last phase has satisfied both the closeness criterion (see inequality (4.10)) and d_{best} has improved (i.e. reduced).

As an example, consider the situation illustrated by the graph in Figure 4.3 (where $m_{phase} = 10$). Initially (move $m = 0$) $d_{best} = d_0 = 9$. Now if d_{bound} and D are taken to be, for example, 3 and 0.5, respectively, then, since $d_{best} = 5$ from moves 8 to 10, both the closeness criterion is satisfied and d_{best} has improved during the initial phase. Hence by the flowchart in Figure 4.1, the next 10 tabu search moves (upto $m_{phase} = 20$) will be an intensification phase.

The start solution used for an intensification phase is the most recent member that enters the ICL. This solution corresponds to the most recently found integer vector with minimum distance value equal to or better than the current value of d_{best}. It is therefore an 'elite' solution that may be used to commence a thorough examination of its region in the search-space.

With reference to the ICL (set I) displayed in Figure 4.3, the intensification phase start solution will be $\underline{z}_8$. Hence Figure 4.4 illustrates an intensification phase (moves 11 to 20). Moreover, based on the graph and data in Figure 4.4, the subsequent phase (moves 21 to 30) will also be an intensification phase, with $\underline{z}_{17}$ as the start integer vector.

4.8 Diversification Strategies

The purpose of a diversification strategy is to enable the search to explore new regions of solution space in the hope of obtaining improved solutions. A diversification strategy is usually employed when the search fails to yield an improved solution within the current region of solution-space and the search needs to explore elsewhere. Diversification is a very important component of a tabu search algorithm, without it the search process may become localised to a small region in solution-space, which would restrict the possibility of seeking a global optimum.

Researchers using tabu search have used a variety of techniques to diversify the search process; for example, moving gaps in the tabu list [Skorin-Kapov 1994], frequency-based memory [Chiang and Kouvelis 1996] and randomly chosen solutions [Ben-Daya and Al-Fawzan 1998].

In this study a number of diversification strategies are used. The actual strategy used for a particular phase of moves depends on the situation at the end of the phase,

that is, whether or not the closeness criterion is satisfied, and, whether or not the value of d_{best} has reduced. With reference to the flowchart in Figure 4.1, the diversification strategy box is expanded in Figure 4.6 to reveal the separate strategies, which are now explained.

- **First-Time Diversification**

The 'first-time diversification' strategy acts a 'first order' strategy in the sense that it is utilised the first time search diversification is required. With reference to the flowchart in Figure 4.6, first-time diversification is used when, after the initial phase of tabu search moves, either the closeness criterion is not satisfied or d_{best} has not improved. Alternatively, it is used after an intensification phase if d_{best} has not improved. In both scenarios an intensification phase is not justified, rather, the search is encouraged to explore a new region.

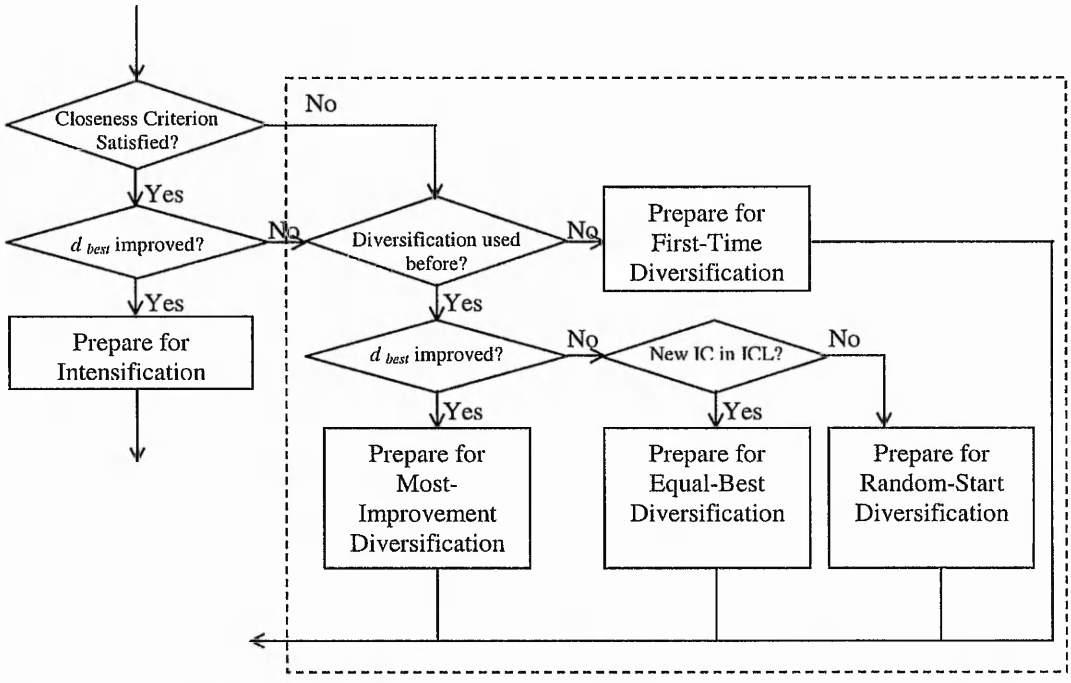


Figure 4.6 Flowchart for diversification strategies.

The start integer vector for the first-time diversification phase is the penultimate IC to enter the ICL. This particular (start) IC, whilst being a 'good' solution, is further back in the ICL than that used for intensification purposes, so that the search may investigate a region sufficiently distant from the current search 'position' (i.e. the end of phase integer vector).

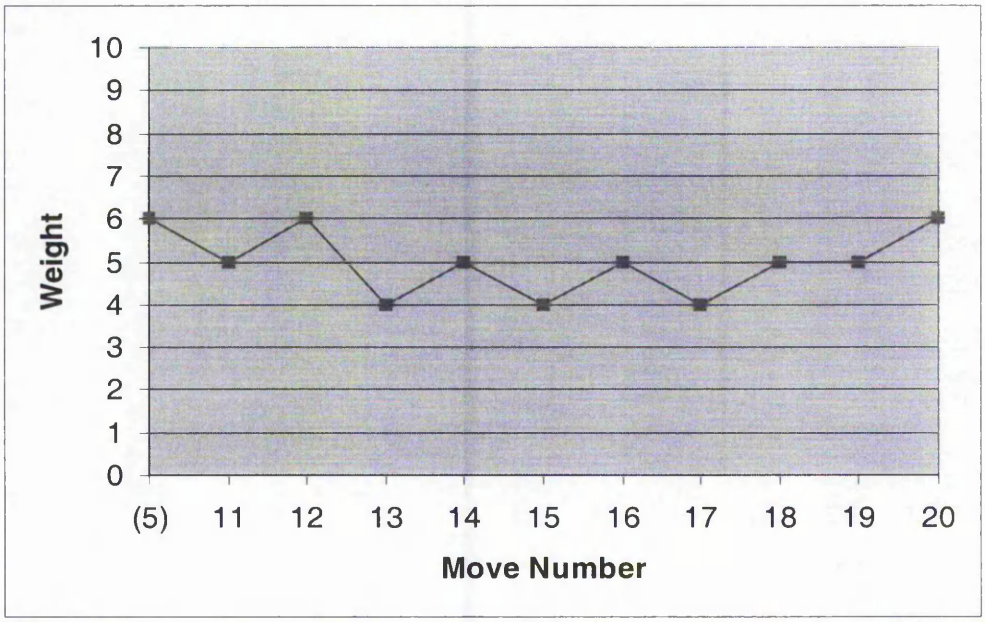
With reference to the illustrative graph shown in Figure 4.3, if $d_{bound} = 3$ and $D = 0.7$, then since $d_{best} = 5$, the closeness criterion (inequality (4.10)) is not satisfied. Hence moves 11 and 20 are a first-time diversification phase with start integer vector $\underline{z}_5$ (with $d_5 = 6$); the process explained in Section 4.6 is used to update lists T, I, F, and U for this phase. Notice that $\underline{z}_5$ is some way (hamming distance) back in the phase and so the search would have the opportunity to move away from the current search path and explore a new region.

- **Most-Improvement Diversification**

Diversification strategies used subsequent to first-time diversification may be considered as 'second-order' strategies and represent means of encouraging the search to explore new regions under increasingly adverse conditions (in terms of obtaining increasingly better solutions, i.e. lower weights).

With reference to the flowchart in Figure 4.6, the most-improvement diversification strategy will be used for a phase of moves if, during the previous phase of moves, the closeness criterion is not satisfied but there has been an improvement in d_{best}. To illustrate this situation, consider the graph in Figure 4.3 to represent the first phase of tabu search moves with first-time diversification (moves 11 to 20) starting from integer vector $\underline{z}_5$. The search path and lists for this first-time diversification phase are shown in Figure 4.7.

Assume that the values of d_{bound} and D are such that the convergence criterion is not satisfied during moves 1 to 20. However, it is seen in Figure 4.7 that d_{best} has improved during the current phase of moves. The subsequent phase of moves (21 to 30) will therefore use the most-improvement diversification strategy. The start solution for this particular diversification phase is the integer vector with the earliest found value of the end-of-phase d_{best}, that is, the first integer vector with the most improved weight. With reference to Figure 4.7, d_{best} = 4 from move 13 onwards so that the start integer vector for most-improvement diversification is $\underline{z}_{13}$.



T	I	F	U
$\underline{0}$	$\underline{z}_1$	1	0
$\underline{z}_4$	$\underline{z}_2$	1	0
$\underline{z}_5$	$\underline{z}_4$	1	0
:	$\underline{z}_5$	2	1
$\underline{z}_{10}$	$\underline{z}_8$	1	0
$\underline{z}_{11}$	$\underline{z}_{11}$	1	0
:	$\underline{z}_{13}$	1	0
:	$\underline{z}_{15}$	1	0
$\underline{z}_{20}$	$\underline{z}_{17}$	1	0

Figure 4.7 Search process after phase 2 (moves 11 to 20).

Compared to the first-time diversification strategy the start integer vector for the most-improvement diversification strategy may require a greater 'back-jump' from the search end-of-phase integer vector, and hence the possibility of greater diversification.

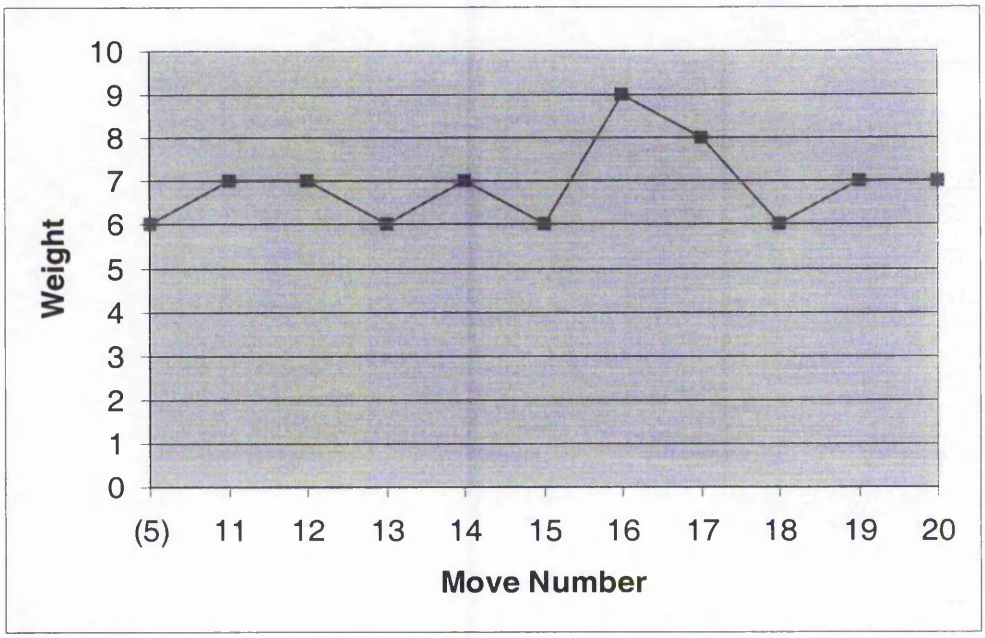
- **Equal-Best Diversification**

Like the most-improvement diversification strategy, the equal-best strategy is performed subsequent to first-time diversification. If during a phase of moves the value of d_{best} does not improve but integer vectors (at least one) with weights equal to d_{best} (i.e. 'equal best') have been found (and hence entered I), then equal-best diversification is used in the next phase of moves. The equal-best diversification strategy is not dependent on the closeness criterion.

To illustrate this situation, consider the graph and lists as shown in Figure 4.8, which gives a search history for a first-time diversification phase (moves 11 to 20). With reference to the graph in Figure 4.8, the weights are such that $d_i \geq d_{best}$, $i = 11,12,...,20$, with equality when $i = 13$, 15 and 18 (hence $\underline{z}_{13}$, $\underline{z}_{15}$ and $\underline{z}_{18}$ are influential candidates). Since d_{best} is not reduced during this phase, moves 21 to 30 will be an equal-best diversification phase.

The start integer vector for equal-best diversification is the earliest integer vector found during the current phase with weight equal to d_{best} (i.e. the least recent IC). With reference to the ICL in Figure 4.8, this will be $\underline{z}_{13}$.

Again, compared to the first-time diversification strategy, the start integer vector for equal-best diversification may require a greater 'back-jump' from the end-of-phase integer vector, and hence the possibility of greater diversification.



T
$\underline{0}$
$\underline{z}_4$
$\underline{z}_5$
:
$\underline{z}_{10}$
$\underline{z}_{11}$
:
$\underline{z}_{20}$

I	F	U
$\underline{z}_1$	1	0
$\underline{z}_2$	1	0
$\underline{z}_4$	1	0
$\underline{z}_5$	2	1
$\underline{z}_8$	1	0
$\underline{z}_{13}$	1	0
$\underline{z}_{15}$	1	0
$\underline{z}_{18}$	1	0

Figure 4.8 Search process after phase 2 (moves 11 to 20).

- **Random-Start Diversification**

For a particular phase of moves, the random-start diversification strategy is used under the same circumstances as the equal-best strategy, except that no influential candidate has entered the ICL during the previous phase of moves (see flowchart in Figure 4.6). With reference to the graph in Figure 4.8, if $\underline{z}_{13}$, $\underline{z}_{15}$ and $\underline{z}_{18}$ were such that $d_i > 6$, $i =$ 13, 15 and 18 (and hence were not members of the ICL) then $d_i > d_{best}$ for $i = 11, 12, \ldots, 20$. Hence the random-start diversification is used for the next phase of moves (21 to 30).

The start integer vector for random-start diversification is generated randomly from the least recent IC with usability-state 0 and the highest frequency count (if all

influential candidates have usability-state 1 then the least recent IC with the highest frequency count is selected). With reference to the lists in Figure 4.8 (omitting $\underline{z}_{13}$, $\underline{z}_{15}$ and $\underline{z}_{18}$) the generating IC would be $\underline{z}_1$.

The random-start diversification start integer vector $\underline{z}_r$, is obtained from the generating IC, $\underline{z}_g$, (where $\underline{z}_g$ is $\underline{z}_1$ in this example) by randomly selecting, for each integer element $z_{gi} \in \underline{z}_g$, $i = 1,2,...,B-1$, an integer $z_{ri} \in \underline{z}_r$, $i = 1,2,...,B-1$, that has a separation of at least $\frac{(2^{31}-1)}{4}$, i.e.

$$|z_{gi} - z_{ri}| > \frac{(2^{31}-1)}{4}, \quad i = 1,2,...,B-1. \tag{4.11}$$

For the last block (B) integer element $z_{gB} \in \{0,1,...,k \bmod 31\}$ and so z_{rB} is chosen randomly from this set.

Random-start diversification (here moves 21 to 30) starts with the I, F and U lists shown in Figure 4.8 (with $\underline{z}_{13}$, $\underline{z}_{15}$ and $\underline{z}_{18}$, and associated u and f values omitted) except that for the generating IC, here $\underline{z}_1$; $f_1 = f_1 + 1 = 2$ and $u_1 = 1$. The tabu list is cleared of all integers and contains $\underline{0}$ and $\underline{z}_r$ only. The lists are then maintained, in the manner previously described, for subsequent moves (21 to 30).

During a random-start diversification phase of moves the tabu list acts as a short-term memory; it is anticipated that the point-of-search will have (randomly) 'jumped' to a 'distance point' and hence would render the existing tabu list elements redundant.

The use of a least recent IC to generate the random-start integer vector makes use of the full range of the ICL (not just members of a recent phase) and so gives as many members as possible the chance to contribute to phase starts.

In practice, 5 candidate start integer vectors were randomly generated from a generating integer vector, $\underline{z}_g$, and the one with the lowest weight was selected to be $\underline{z}_r$. Also the Hamming distance between the associated alpha-vectors $\underline{\alpha}_g$ and $\underline{\alpha}_r$ was checked and found to be suitably large (approximately 35% difference in hamming distance).

4.9 Minimum Distance Results

In this study 10 codes (BCH and QR) were investigated by the developed tabu search algorithm using a number of phases, in which each phase comprised 100 moves and the maximum number of moves was set to $m_{\max} = 1000$. In all computational experiments the same neighbourhood structure and start solution as in Chapter 3 were used. Initially $m_{phase} = 100$ and the size of a tabu list ranges between $|T| = 2$ (when move number $m = 0$) and $|T| = 2(m_{phase} + 1)$, after which the search terminates.

- **BCH codes**

In order to assess the developed algorithm, a comparison of the obtained results is presented in Table 4.1 for various BCH codes (numbered as 1 to 10 in this study). Here d_{bound} denotes the known minimum distances for BCH codes and the values of d_{BB} given in Table 4.1 represent minimum distances obtained by Bland and Baylis (1995). Note that improvements were made using the developed tabu search algorithm since this algorithm has extended the search used by Bland and Baylis (1995). In Table 4.1, n and k denote the length and the dimension of a code. The minimum distance obtained is denoted as d_{best} with the associated move, m_{best}, and the executable time t_{best}.

In Table 4.1 it is seen that codes 2, 5, 8 and 10 produced the same minimum distance as the method used by Bland and Baylis (1995) and these minimum distances were found early which implies that the search did not utilise the developed tabu search algorithm. However, codes 1, 3, 4, 6, 7 and 9 have found minimum distances lower than Bland and Baylis (1995). In general, the execution time for each code was extremely quick and the minimum distances obtained are better than (or equal to) those obtained by Bland and Baylis (1995).

Code	n	k	d_{bound}	d_{BB}	d_{best}	m_{best}	$t_{best}(h{:}m{:}s)$
1	127	64	21	26	24	122	0:00:25.55
2	127	57	23	24	24	1	0.00:00.28
3	127	50	27	31	27	288	0:00:28.06
4	255	115	43	60	58	806	0:16:30.48
5	255	107	45	65	65	2	0:00:03.63
6	255	99	47	68	66	603	0:09:24.96
7	255	91	51	71	70	512	0:06:21.13
8	255	87	53	70	70	3	0:00:02.58
9	255	79	55	78	74	958	0:09:04.76
10	255	71	59	77	77	4	0:00:02.86

Table 4.1 Minimum distances of BCH codes using the developed TS algorithm.

n	k	d_{SA}	$t_{best}(h{:}m{:}s)$
127	64	27	0:08:15.00
255	91	75	0:42:17.00

Table 4.2 Minimum distances using simulated annealing.

As a (very limited) comparison, the results for codes 1 and 7, obtained by Zhang and Ma (1994) using simulated annealing, are given in Table 4.2. In Table 4.2 it is observed that tabu search (see d_{BB} and d_{best}) gives lower minimum distances compared to simulated annealing (d_{SA}). Also the execution times are considerably quicker.

- **QR codes**

The tabu search algorithm described in the previous sections was applied to various QR codes. These include codes in which the generator matrices have weight equal to $\frac{p-1}{2}$, the augmented QR codes ($n(x)$ and $q(x)$) and expurgated QR codes ($(x-1)n(x)$ and $(x-1)q(x)$). For each type of QR code, 10 specific codes (numbered 1 to 10 in this study), characterised by their value of prime number p, were investigated. Results for $n(x)$, using the lower and upper bounds of the minimum distances in the closeness criterion (see inequality (4.10)), are presented in Tables 4.3 and 4.4, respectively, and results for other QR codes are presented in Appendix B.

With reference to Tables 4.3 and 4.4, d_{bound} denotes the known minimum distance bounds [MacWilliams 1977] i.e. (where appropriate) lower bound and upper bound, respectively. For each code, the obtained minimum distance, d_{best}, the associated move, m_{best}, together with the executable time t_{best} are presented. Note that the developed algorithm has found the exact minimum distance in the case of code 4 (see Tables 4.3 and 4.4). Furthermore, the execution time is short. Also, it is observed that codes 6 and 7 have different values of d_{best} in Tables 4.3 and 4.4. The reason for this is that for these particular codes the closeness criterion has influenced the search route and led the search to take different paths in solution-space and hence obtain different values of d_{best}. Although there is still a need to improve d_{best} for larger codes (i.e. codes 6 to 10), the results obtained are generally quite close to d_{bound} (upper value) and are obtained in short execution time.

Code	p	d_{bound}	d_{best}	m_{best}	$t_{best}(h{:}m{:}s)$
1	71	11	11	24	0:00:01.05
2	79	15	15	16	0:00:00.99
3	97	15	16	95	0:00:11.26
4	103	19	19	277	0:00:48.01
5	113	11	16	1	0:00:02.22
6	137	13	28	307	0:01:23.87
7	167	15	35	4	0:00:02.91
8	191	15	36	25	0:00:19.72
9	193	15	38	1	0:00:01.31
10	199	15	40	9	0:00:08.40

Table 4.3 Minimum distances obtained using $n(x)$ and lower bound.

Code	p	d_{bound}	d_{best}	m_{best}	$t_{best}(h{:}m{:}s)$
1	71	11	11	24	0:00:01.05
2	79	15	15	16	0:00:00.99
3	97	15	16	95	0:00:11.26
4	103	19	19	277	0:00:48.01
5	113	15	16	1	0:00:02.22
6	137	21	29	90	0:00:26.47
7	167	23	32	633	0:04:41.00
8	191	27	36	25	0:00:19.72
9	193	27	38	1	0:00:01.26
10	199	31	40	9	0:00:08.40

Table 4.4 Minimum distances obtained using $n(x)$ and upper bound.

In order to visualise the optimisation process for code 6 in Table 4.3 (for example), the search route through the algorithm is given in Table 4.5 and the search history (upto move 400) is given in Figures 4.9 to 4.12. With reference to Table 4.3 the search obtained d_{best} at move 307 and the search was terminated after 1000 moves.

With reference to Figure 4.9, the search starts with minimum distance $d_0 = 47$ (the weight of the generator matrix) and the initial search (moves 1 to 100) found d_{best}

= 29 at move 90. Since d_{bound} = 13 (see Table 4.3) the closeness criterion is not satisfied and so moves 101 to 200 (i.e. phase 2) are first-time diversification.

Phase	Move	Search Route
1	1-100	Initial Search
2	101-200	First-Time Diversification
3	201-300	Equal-Best Diversification
4	301-400	Equal-Best Diversification
5	401-500	Most-Improvement Diversification
6 - 10	501-1000	Random-Start Diversification

Table 4.5 Search route for p = 137.

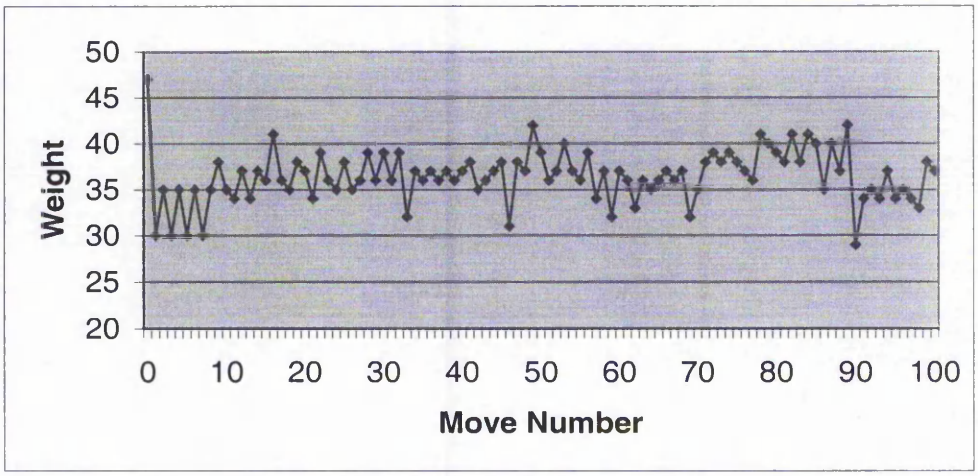


Figure 4.9 Search history (moves 1 to 100) for p = 137.

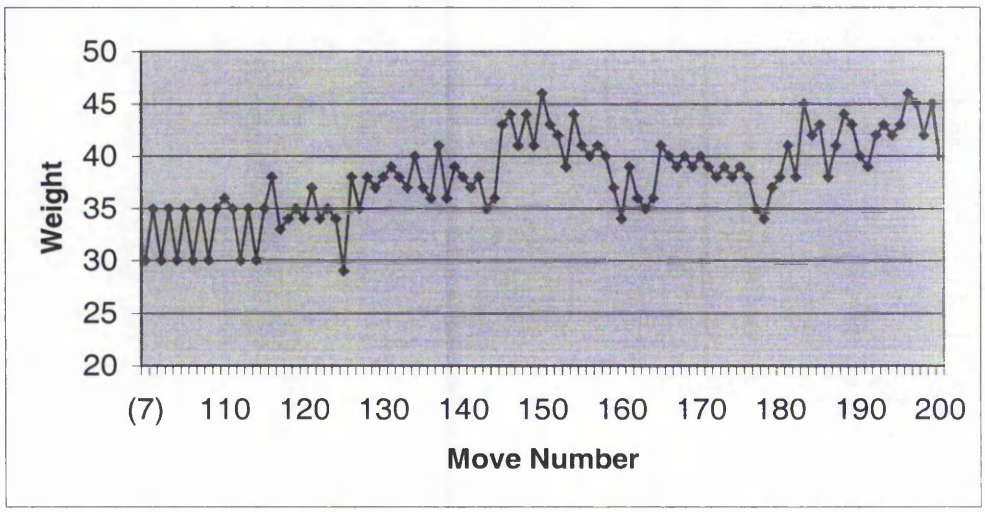


Figure 4.10 Search history (moves 101 to 200) for p = 137.

The first-time diversification phase (phase 2) is shown in Figure 4.10 in which the start solution is the penultimate IC of phase 1, which was found at move 7 (see Figure 4.9) and has minimum distance $d_7 = 30$. Although d_{best} has not improved (still 29) an IC was added to the ICL at move 125 ($d_{125} = d_{best} = 29$). Again the closeness criterion is not satisfied and so phase 3 is an equal-best diversification phase in which the start solution is the integer vector z_{125}.

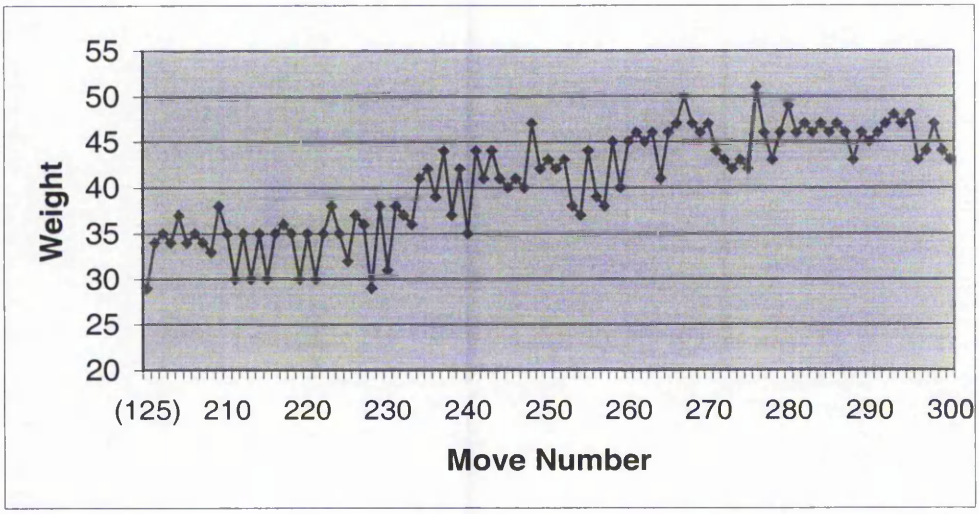


Figure 4.11 Search history (moves 201 to 300) for $p = 137$.

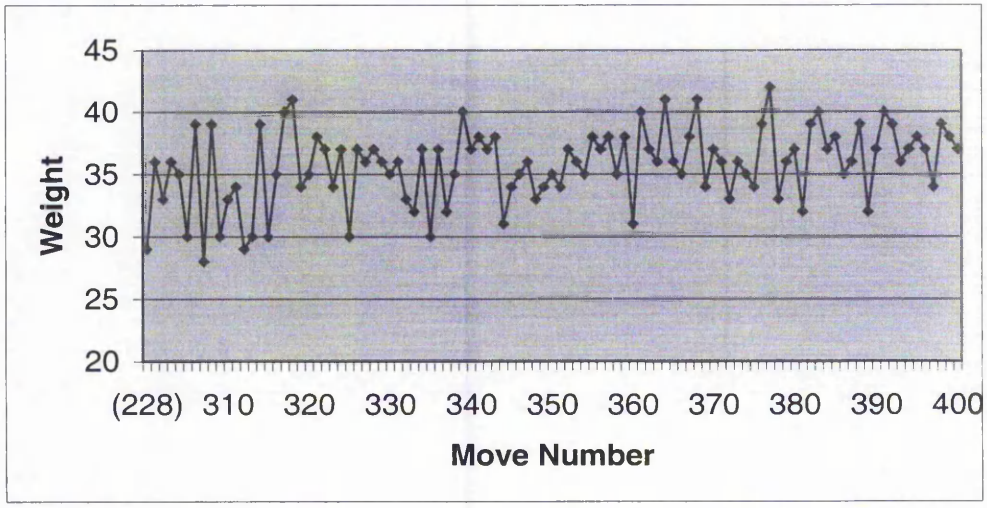


Figure 4.12 Search history (moves 301 to 400) for $p = 137$.

Phase 3 is illustrated in Figure 4.11. As in phase 2 the closeness criterion is not satisfied but an IC ($\underline{z}_{228}$) was added to the ICL. Hence phase 4 is another equal-best diversification phase with start solution $\underline{z}_{228}$ and $d_{228} = 29$.

Figure 4.12 shows the search history for phase 4. An improvement in the minimum distance is made at move 307 with $d_{best} = d_{307} = 28$. The closeness criterion is still not satisfied so the next phase (phase 5) is a most-improvement diversification phase with start solution $\underline{z}_{307}$.

In phases 6 to 10 the minimum distances associated with the search solutions were all greater than $d_{best} = 28$ and so these phases utilised random-start diversification.

Minimum distance results using the developed tabu search algorithm are presented in Table 4.6. These particular results are the improvements in the minimum distance values compared to the equivalent results obtained using the basic tabu search algorithm of Chapter 3.

With reference to Table 4.6, for prime number (p), G denotes the generator polynomial (see Chapter 2), and d_{bound}, d_{QR} and d_{best} denote the minimum distance bound used in the closeness criterion, the (range of) the minimum distance bound(s) [MacWilliams and Sloane 1977] and the obtained minimum distance, respectively. Associated with d_{best} is the move number, m_{best}, and execution time, t_{best}.

Table 4.6 shows that the use of the developed tabu search algorithm may improve some of the minimum distances obtained in Chapter 3. It is observed in Table 4.6 that there is still scope to improve minimum distances, particularly for the

larger QR codes. In the next chapter a developed ACO algorithm will be presented and focus on the larger QR codes.

Code	p	$g(x)$	d_{bound}	d_{QR}	d_{best}	m_{best}	t_{best} (h:m:s)
1	71	$q(x)$	11	11	11	413	0:00:08.57
2	79	$q(x)$	15	15	15	213	0:00:05.76
3	97	$(x-1)q(x)$	16	16	20	643	0:00:31.45
4	103	$n(x)$	19	19	19	277	0:00:48.01
6	137	$n(x)$	13	13-21	28	307	0:01:23.87
6	137	$q(x)$	21	13-21	27	154	0:00:43.17
7	167	$n(x)$	23	15-23	32	633	0:04:41.00
7	167	$q(x)$	15	15-23	32	381	0:03:06.92
7	167	$(x-1)q(x)$	16	16-24	32	164	0:01:09.26
8	191	$q(x)$	27	15-27	35	233	0:02:25.56
8	191	$(x-1)n(x)$	16	16-28	36	109	0:01:09.75
8	191	$(x-1)n(x)$	28	16-28	36	210	0:02:27.20
9	193	$q(x)$	15	15-27	38	110	0:01:04.26
10	199	$q(x)$	15	15-31	40	107	0:01:08.05
10	199	$(x-1)q(x)$	16	16-32	40	136	0:01:38.59
10	199	$(x-1)q(x)$	32	16-32	40	324	0:04:37.65

Table 4.6 Minimum distances using the developed tabu search algorithm.

Chapter 5

Ant Colony System for Minimum Distances

5.1 Introduction

In this chapter an ant colony system (ACS) algorithm is presented with improved decision-making capability compared to the ant system (AS) algorithm explained in Chapter 3. Here ants deposit pheromone as a means of communication in a dynamic environment, known as stigmergy [Dorigo and Caro 1999, Grassé 1959].

In the basic AS algorithm of Chapter 3 each ant generates an alpha-vector (independently of the other ants) before autocatalysis and trail evaporation are used to obtain the colony trail intensity for the next time step. The combination (summation) of the trail intensity matrices for each ant (independently obtained) acts as a communication tool to provide collective (i.e. colony) information. As time progresses the quality of the colony information improves so that alpha-vectors with associated low minimum distance values will have an increased probability of being generated.

The ACS has improved decision-making capability compared to AS through the use of a more general decision rule which incorporates the processes of exploration and exploitation (see Section 5.3). Also, ACS uses local trail updating to allow ants to modify their individual trail intensities (i.e. see Section 5.5) and produce a diversity of solutions (alpha-vectors). Finally, a global trail updating rule is used with autocatalysis and trail evaporation to focus on the selection of good solutions in the next time step. This process is similar to reinforcement learning [Kaelbling et al 1996] by which an agent (here ant) learns through interaction with a dynamic environment

in which better solutions are rewarded with higher reinforcements (pheromone deposits).

A key feature of the developed ACS of this chapter is 'ant co-operation'. Here an interplay strategy between local trail updating (of ACS) and diversification (of tabu search) is presented which enables ants to share information (i.e. co-operate) within and between time steps (explained in Section 5.6).

5.2 Overview of the Developed ACS Algorithm

In this chapter an ACO algorithm is presented which includes several features (explained in Sections 5.3 to 5.8) which are absent from the basic ant system (AS) explained in Chapter 3. The presented ACO algorithm, which is of the ant colony system (ACS) type [Dorigo and Gambardella 1997], is designed to improve the capability of an 'ant' approach to optimisation (compared to the AS algorithm of Chapter 3) in terms of obtaining lower minimum distances. An overview of the developed ACS algorithm, showing strategic features, is given in the flowchart in Figure 5.1.

With reference to the flowchart in Figure 5.1, input data includes code details such as (for QR codes) prime number p and the generator matrix G. Also required are ACO parameters with user-specified values (e.g. indices a and b, see Chapter 3). search details are also required; the number of ants ($x_{\max}$), the number of time steps ($t_{\max}$) and the number of moves per phase (m_{phase}) used in the diversification phase (see Section 5.4).

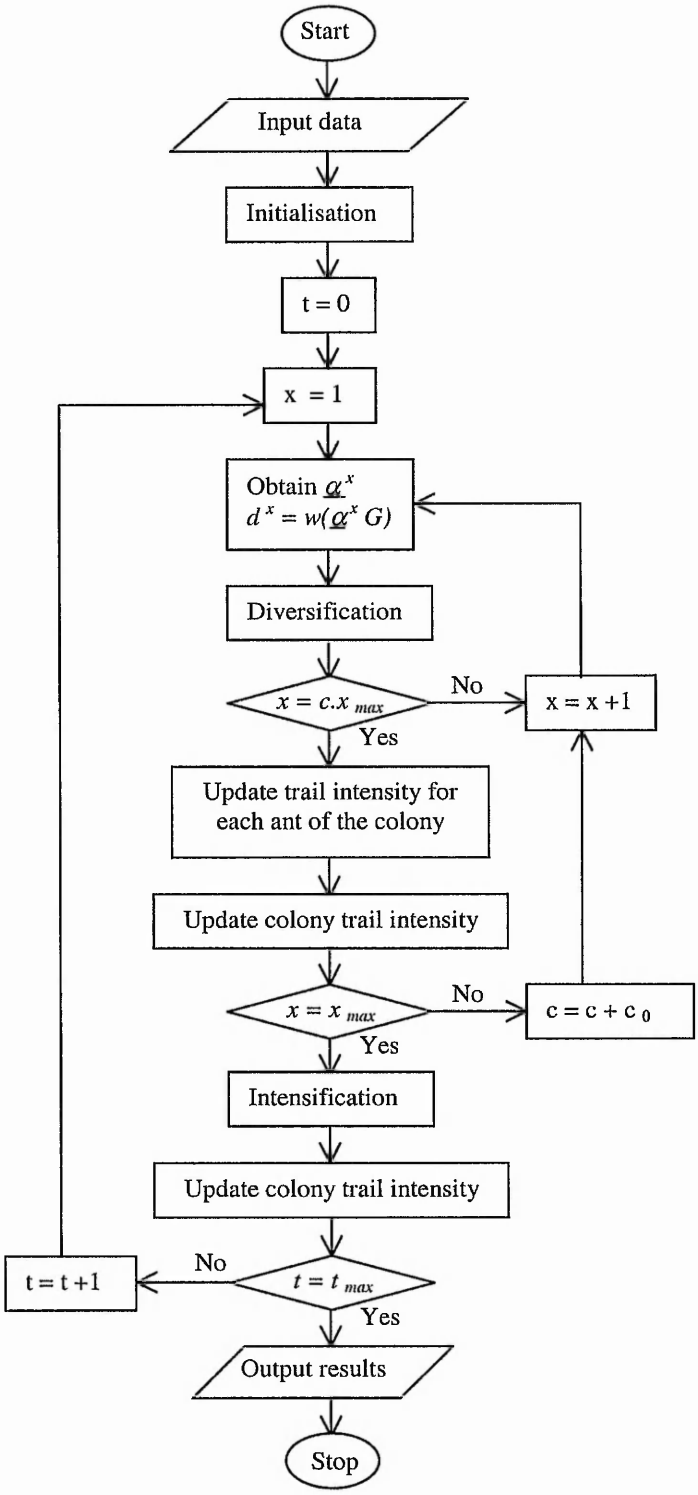


Figure 5.1 Flowchart for the developed ACS algorithm.

The initialisation phase sets the initial values of the colony trail intensity and trail desirability, $\tau_{ij}(t)$ and $\eta_{ij}(t)$, respectively, where time step $t = 0$, bit state $i = 0,1$ and bit number $j = 1,2,...,k$ with $k = \frac{p+1}{2}$ for augmented QR codes and $k = \frac{p-1}{2}$ for expurgated QR codes. Also, the alpha-vector for each ant is set to $\underline{0}$ (explained in Section 5.6).

On entering the time step loop an inner and outer ant loop (variable x denotes ant number $x = 1,2,...,x_{max}$) is also entered. The ant loops are used to model ant 'co-operation', in terms of 'sharing' information, as represented by the trail intensity values, $\tau_{ij}(t)$, for each ant (explained in Section 5.6). The parameter c (with initial input value c_0) in the decision point in the inner ant loop is used to allow an ever increasing fraction of the colony of ants to generate alpha-vectors within the current time step (the fraction is increased in the outer ant loop until $x = x_{max}$; see Section 5.6).

Alpha-vectors are generated on entering the (inner) ant loop. Here a state-transition decision rule is used that allows both exploration and exploitation [Dorigo and Gambardella 1997], explained in Section 5.3. Next a diversification phase is performed in which ants are encouraged to generate a diverse 'colony' of alpha-vectors as an exploration aid (see Section 5.4). During this phase certain alpha-vectors are identified for the intensification phase (performed later).

After each fraction of the colony of ants has generated their alpha-vectors the trail intensity for each ant of the colony is obtained and the colony trail intensity is modified so that ants may share information and co-operate in exploration, i.e. a wide search for low minimum distances, see Section 5.6. On exiting the outer ant loop

intensification is performed on those alpha-vectors identified during the diversification phase, as explained in Section 5.7.

Finally, on leaving the time step loop, the overall lowest minimum distance together with the associated alpha-vector and codeword are output, prior to termination.

5.3 ACS State-transition Decision Rule

The state-transition decision rule used in the AS algorithm of Chapter 3, called the random-proportional rule [Gambardella and Dorigo 1995] concentrated on exploration only, i.e. obtaining (a diversity of) alpha-vectors in a stochastic manner. In this chapter the pseudo-random proportional rule [Dorigo and Gambardella 1997] is used, as follows,

$$\begin{aligned} &\textbf{If } r_1 > \upsilon \quad \textbf{then} \text{ exploration} \\ &\qquad\qquad\quad \textbf{else} \text{ exploitation} \end{aligned} \tag{5.1}$$

where υ is a user-specified input parameter, $0<\upsilon<1$, which expresses the balance between exploration and exploitation, and $r_1 \in [0,1]$ is a random number drawn from a uniform distribution.

- **Exploration**

Exploration is a stochastic process in which the construction of an alpha-vector is performed using a probabilistic rule. For ant x, at time step t, with alpha-vector $\underline{\alpha}^x = \{\alpha_j^x\}$, $j = 1,2,...,k$,

$$\alpha_j^x = \begin{cases} 0 & \text{if } r_2 \le P_{0j}^x(t) \\ 1 & \text{otherwise} \end{cases} \tag{5.2}$$

where probability $P_{0j}^x(t)$ is given by equation (3.17) and $r_2 \in [0,1]$ is a random number drawn from a uniform distribution.

As previously stated, the above decision rule is identical to that used in the AS of Chapter 3, except here it is used only if $r_1 > \upsilon$. When $r_1 \leq \upsilon$ exploitation is used.

- **Exploitation**

Exploitation is a process in which the construction of an alpha-vector is performed using a simple comparison of probability values,

$$\alpha_j^x = \begin{cases} 0 & \text{if } P_{0j}^x(t) \geq P_{1j}^x(t) \\ 1 & \text{otherwise} \end{cases} \qquad (5.3)$$

where, from equation (3.18), $P_{1j}^x(t) = 1 - P_{0j}^x(t)$.

The above pseudo-random-proportional rule is more versatile than the random-proportional rule of Chapter 3 in the sense that the parameter υ allows a combination of exploration (i.e. the generation of a diversity of alpha-vectors) and exploitation (i.e. the favouring of those couplings with high amounts of pheromone, in order to seek alpha-vectors with relatively low minimum distances).

Once ant x has constructed its alpha-vector, $\underline{\alpha}^x = \{\alpha_j^x\}, j = 1,2,...,k$ then the associated codeword and minimum distance are $\underline{c}^x = \underline{\alpha}^x G$ and $d^x = w(\underline{c}^x)$, respectively. As with the AS algorithm, the codeword $\underline{c}^x = \underline{0}$ is forbidden and so $\underline{\alpha}^x = \underline{0}$, $x \in \{1,2,..., x_{\max}\}$ is used as a stopping condition.

5.4 Diversification Phase

The diversification phase of the developed ACO algorithm is illustrated by the flowchart in Figure 5.2. In ACO 'diversification' has a slightly different meaning to that used in TS, where the search is encouraged to move into previously unexplored regions of $\underline{\alpha}$-space. In an ACO context, diversification refers to the generation of a population (colony) of alpha-vectors with a high level of diversity (or variation) i.e.

different alpha-vectors. However, to achieve ACO diversification the TS diversification strategies of Chapter 4 Section 4.8 are employed (explained in this section) together with (local) updating of the trail intensity, $\tau_{ij}^{x}(t)$, $i=0,1$, $j=1,2,...,k$, for each ant x (see Section 5.5).

With reference to the flowchart in Figure 5.2, first the intermediate-term and long-term memories (i.e. tabu list, T^x, and influential candidate list (ICL), I^x) for each ant x are initialised; I^x is cleared and T^x contains the zero integer vector $\underline{0}$ and the integer vector associated with the constructed (start) alpha-vector, $\underline{\alpha}_0^x$, i.e. $\underline{z}_0^x$, $x \in \{1,2,...,x_{\max}\}$. Next a phase of TS moves (comprising m_{phase} moves) is performed during which the current lowest minimum distance value, $d_{\min}^x$, is recorded and the lists T^x and I^x are managed in the manner explained in Chapter 4 Section 4.8. After the initial m_{phase} moves the distance ratio for ant x, R^x, is calculated, where

$$R^x \equiv \frac{d_{bound}}{d_{\min}^x} \tag{5.4}$$

and matched against the (initial) threshold value D_0, where $0 < D_0 < 1$. As with the closeness criterion of Chapter 4 Section 4.5, the value of D_0 is taken to be 0.6.

With reference to the flowchart in Figure 5.2, if $R^x > D_0$ then the search is in a 'promising' region of solution-space in terms of low minimum distances. This region is investigated further by means of an intensification phase, which is performed later. For the moment, the intensification phase start solution (integer vector) is identified to be the most recent influential candidate (IC) in the ICL for ant x, i.e. in I^x.

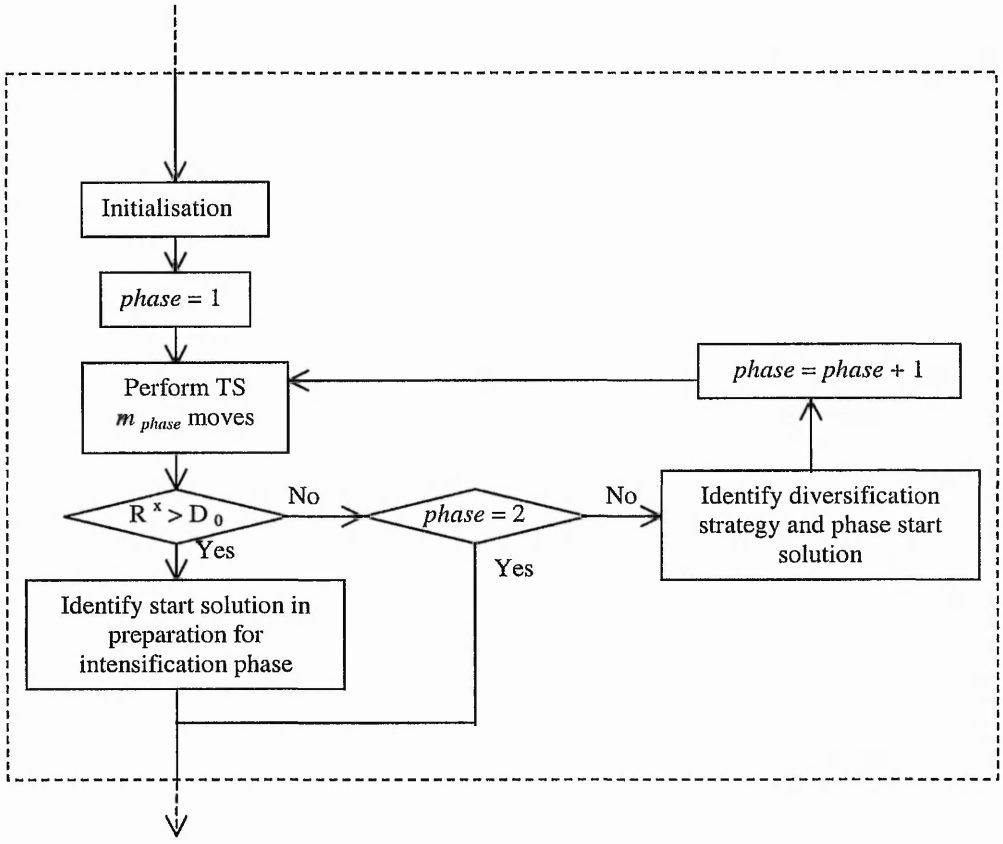


Figure 5.2 Flowchart for the diversification phase.

If $R^x \leq D_0$ and provided *phase* < 2, then, since the search is not in a particularly good region in terms of low minimum distances, a diversification strategy is identified together with the appropriate phase start solution. A second set of m_{phase} moves is performed prior to matching the current value of R^x against D_0 for the last time for ant x within the current time step. The current value of R^x is based on the current value of $d^x_{\min}$ which, if necessary, is updated during the second (diversification) set of TS moves.

The identification of the diversification strategy is given in the flowchart in Figure 5.3. The particular strategies, i.e. First-Time and Most-Improvement are the

same as those presented in Chapter 4, Section 4.8, and have the appropriate start integer vectors. Here the Random-Start diversification strategy has a start integer vector with elements randomly generated between 1 and $2^{31}-1$. The conditions for a diversification strategy and the particular strategy are based on the degradation of the threshold value; $D_i = D_0 - i.h$, $i = 0,1,2$, where, in this study $h = 0.1$. In other words, in general, the start integer vector is chosen further back in the search (i.e. I^x) as the threshold value degrades.

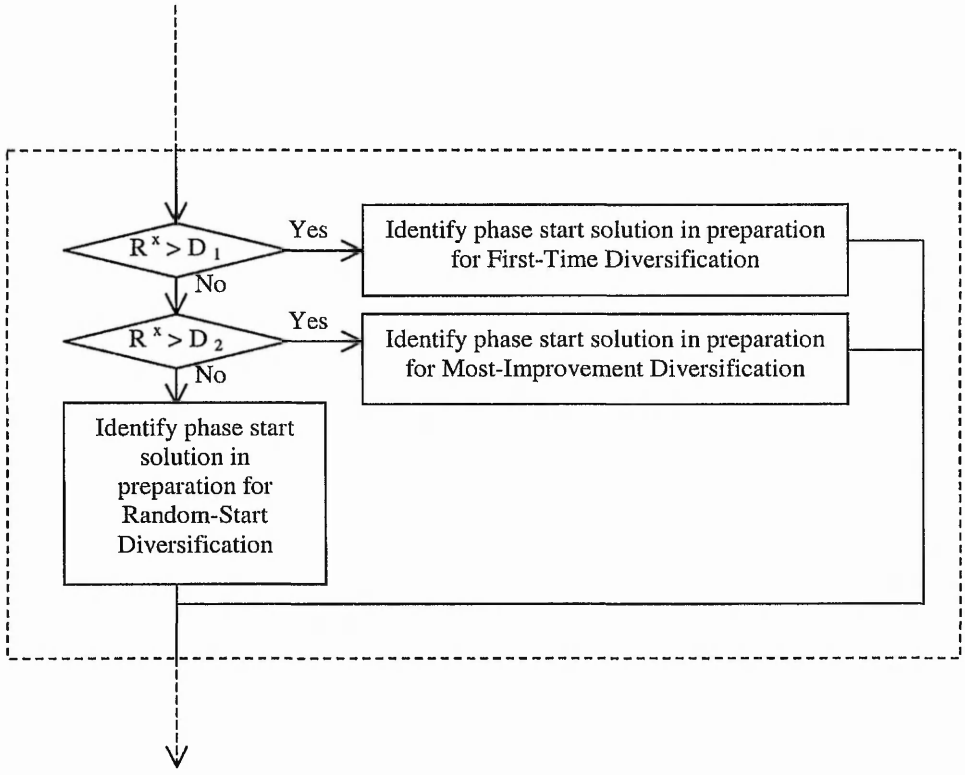


Figure 5.3 Flowchart for diversification strategies.

As previously stated, the aim of the diversification phase is to produce a diverse set of alpha-vectors, $\underline{\alpha}^x, x = 1,2,...,x_{max}$ (with, preferably, low minimum distances, d^x_{min}). To achieve this, TS diversification strategies are employed (explained above) together with a local updating phase for the trail intensity for each ant and for the

colony, which influences the start integer vector used in the initial phase of TS moves. The local pheromone level updating phase is now explained.

5.5 Local Trail Intensity Update

Besides the TS diversification strategies (presented in the previous section) another component to aid diversification/variation in the population of alpha-vectors (and the avoidance of stagnation) is the use of local updating [Dorigo and Gambardella 1997].

In the AS algorithm explained in Chapter 3 the alpha-vector for each ant is obtained stochastically (pure exploration) using the random-proportional rule (see equation (5.2)). This rule involves $P_{0j}^{x}(t)$ (see equation (3.17)) in which the colony trail intensity, $\tau_{ij}(t)$, remains fixed throughout the construction of all alpha-vectors, $\underline{\alpha}^{x}$, $x = 1,2,...,x_{\max}$.

Local updating of the trail intensity is a feature of ACS algorithms [Dorigo and Gambardella 1997] and involves the modification of $\tau_{ij}(t)$ within the current time step so that $P_{0j}^{x}(t)$ is not a fixed quantity for all ants $x = 1,2,...,x_{\max}$. In other words, at time step t, the value of $P_{0j}^{x}(t)$ depends on x, which will aid variation in the constructed alpha-vectors.

The particular trail intensity local updating rule used in this study is as follows, which applies to ant $x \in \{1,2,...,x_{\max}\}$ at time step t,

$$\tau_{ij} = \begin{cases} (1-\rho_L)\tau_{ij} & \text{if } \alpha_j^x = i \\ (1-\rho_L)\tau_{ij} + \rho_L \Delta\tau_j & \text{otherwise} \end{cases} \qquad (5.5)$$

where $\underline{\alpha} = \{\alpha_j^x\}$, $j = 1,2,...,k$ and $i = 0,1$. In equation (5.5) ρ_L is a user-specified input parameter, $0 < \rho_L < 1$, in which $(1-\rho_L)$ represents the local pheromone

where $\underline{\alpha} = \{\alpha_j^x\}$, $j = 1,2,...,k$ and $i = 0,1$. In equation (5.5) ρ_L is a user-specified input parameter, $0 < \rho_L < 1$, in which $(1-\rho_L)$ represents the local pheromone evaporation factor and $\rho_L \Delta\tau_{ij}$ is a positive amount of pheromone deposited to affect the value of $P_{0j}^x(t)$ in such a way as to aid variation in $\underline{\alpha}^x$, $x = 1,2,...,x_{\max}$ (described below). By taking $\Delta\tau_j \equiv |\tau_{0j} - \tau_{1j}|$, $j = 1,2,...,k,$, the influence of local updating (by equation (5.5)) will be directly proportional to the difference in the trail intensity values for bit states 0 and 1.

To investigate the influence of the local updating rule (given by equation (5.5)) on the expression for $P_{0j}^x(t)$ in the state transition rule used to generate alpha-vectors (see equations (5.2) and (5.3)), equation (3.17) is considered (with the notation for ant x and time step t omitted) and may be expressed as follows,

$$P_{0j} = \frac{1}{1+\lambda_j^a \varepsilon_j^b}, \quad j = 1,2,...,k \tag{5.6}$$

where

$$\lambda_j \equiv \frac{\tau_{1j}}{\tau_{0j}} \text{ and } \varepsilon_j \equiv \frac{\eta_{1j}}{\eta_{0j}} \tag{5.7}$$

Since the trail desirability values η_{ij}, $i = 0,1$, $j = 1,2,...,k$ are fixed quantities at each time step, ε_j is constant at a time step. A graph of P_{0j} against λ_j (for $a \geq 1$, $b \geq 0$) for equation (5.6) is given in Figure 5.4.

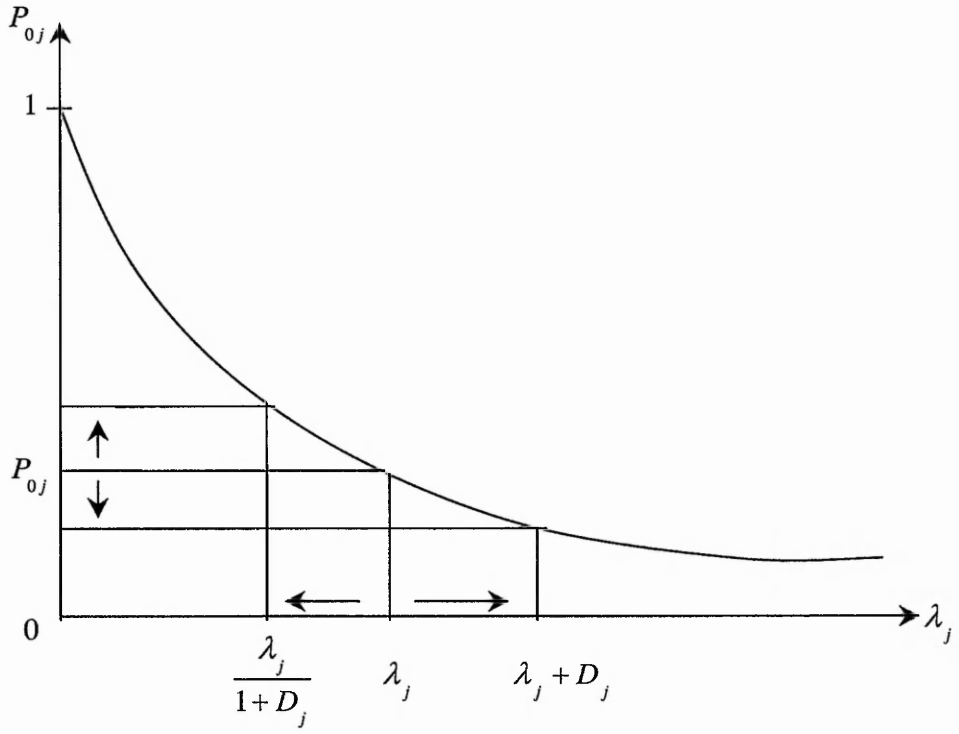


Figure 5.4 **Graph of** $P_{0j} = \dfrac{1}{1+\lambda_j^a \varepsilon_j^b}$.

Using equations (5.5), if $\alpha_j^x = 0$ then, by equation (5.7),

$$\lambda_j \equiv \frac{(1-\rho_L)\tau_{1j} + \rho_L \Delta\tau_j}{(1-\rho_L)\tau_{0j}} \tag{5.8}$$

i.e. $$\lambda_j = \lambda_j + D_j \tag{5.9}$$

where $$D_j \equiv \frac{\rho_L \Delta\tau_j}{(1-\rho_L)\tau_{0j}} \geq 0 \tag{5.10}$$

Hence, with reference to the graph in Figure 5.4, when bit number j has state 0, then the effect of the local updating rule is to decrease the value of P_{0j} so that the chance

of a subsequent ant choosing bit state 0 is (slightly) reduced. Alternatively, if $\alpha_j^x = 1$, then equation (5.5) leads to

$$\lambda_j = \frac{\lambda_j}{1 + D_j} \tag{5.11}$$

and so the local updating rule (slightly) increases the chance of a subsequent ant choosing bit state 0 (see Figure 5.4).

5.6 Ant Co-operation

As explained in Sections 5.4 and 5.5, the use of TS diversification strategies and a local updating rule for the trail intensity aids diversity/variation in the 'colony' of alpha-vectors.

In this section an 'interplay' strategy is presented for the above two diversification mechanisms so that ants may share information and co-operate in the diversification process.

Ant co-operation, in a computational sense, is modelled by the inner and outer ant (x) loops in the flowchart in Figure 5.1. To explain the co-operation (information sharing) process an example is given for a colony comprising 4 ants only ($x_{\max} = 4$).

Before entering the double ant loop the initial trail intensity for the colony has been input, i.e. $\tau_{ij}(0) = \tau_0$, $i = 0,1$, $j = 1,2,...,k$, where τ_0 is a small positive number. Also, already input is the colony fraction, c_0, where the current colony fraction, c, is set to the input value; $c = c_0$. The value of c_0 represents the fraction of the colony of ants (in other words, $c_0 . x_{\max}$ ants) that use the current colony trail intensity matrix in the generation of alpha-vectors before applying the local updating rule to obtain a new

(updated) colony trail intensity matrix. In this example $c_0 = \frac{1}{2}$. Hence, initially, $c = c_0 = \frac{1}{2}$.

All alpha-vectors are arbitrarily initialised to $\underline{0}$; $\underline{\alpha}^x = \underline{0}$, $x = 1,2,...,x_{\max}$. Note that the initial bit states do not affect the state transition probabilities, $P_{0j}^x(0)$ (see equation (5.6)) when local updating is first applied because, initially, $\tau_{0j} = \tau_{1j} = \tau_0$, $j = 1,2,...,k$ so that $D_j = 0$ (see equations (5.9), (5.10) and (5.11)) and so $\lambda_j = 1$ is maintained.

With an ant colony such that $x_{\max} = 4$, using the state-transition rule (equations (5.1), (5.2) and (5.3)) with $P_{0j}^x(0)$ based on $\tau_{ij}(0)$, the inner loop generates (with TS diversification) alpha-vectors for ants $x = 1$ and 2, i.e. $\underline{\alpha}^1$ and $\underline{\alpha}^2$ ($\underline{\alpha}^1$ and $\underline{\alpha}^4 = \underline{0}$). Next, in the outer loop, based on the current colony of alpha-vectors and the current colony trail intensity matrix, the trail intensity matrix for each ant is obtained using the local updating rule (equation (5.5)), i.e. matrices $\delta\tau_{ij}^m(0)$, $m = 1,2,3,4$, where

$$\delta\tau_{ij}^m(0) = \{\tau_{ij}^m(0)\} \tag{5.12}$$

and $\tau_{ij}^m(0)$ is given by equation (5.5), in which the ant number m and time step $t = 0$ are omitted.

The colony trail intensity matrix for the current time step ($t = 0$) is then updated, as follows to become

$$\tau_{ij}(0)^{(1)} = \tau_{ij}(0) + \sum_{m=1}^{4} \delta\tau_{ij}^m(0) \tag{5.13}$$

As this stage $x = 2 \neq x_{\max}$ so the inner loop is re-entered with c and x incremented to 1 and 3, respectively. Now using equations (5.1) to (5.3) with $P_{0j}^{x}(0)$ based on $\tau_{ij}(0)^{(1)}$ (see equation (5.13)) the inner loop generates (with TS diversification) alpha-vectors for ants x = 3 and 4, i.e. $\underline{\alpha}^3$ and $\underline{\alpha}^4$ ($\underline{\alpha}^1$ and $\underline{\alpha}^2$ are those produced during the first tour of the inner loop). Again, in the outer loop, with the current alpha-vectors, $\underline{\alpha}^m$, $m = 1,2,3,4$ and current colony trail intensity matrix, $\tau_{ij}(0)^{(1)}$, the trail intensity matrix for each ant is re-calculated, $\delta\tau_{ij}^{m}(0)^{(1)}$, $m = 1,2,3,4$, using the local updating rule. The colony trail intensity matrix for the current time step is again updated to become

$$\tau_{ij}(0)^{(2)} = \tau_{ij}(0)^{(1)} + \sum_{m=1}^{4} \delta\tau_{ij}^{m}(0)^{(1)} \qquad (5.14)$$

Since $x = x_{\max}$ the outer loop is exited. Notice that after the second tour of the inner loop the generation of the second 50% of the colony of alpha-vectors is influenced by the alpha-vectors produced in the first tour of the inner loop. In other words, information (in terms of trail intensity values/pheromone levels) obtained by the first 50% of the colony of ants is 'shared' with the other 50% of the colony; i.e. the ants co-operate in the pursuit of alpha-vectors with diversity.

With reference to the flowchart in Figure 5.1, on leaving the outer ant loop and after an intensification phase and a (global) colony trail intensity update, which gives $\tau_{ij}(1)$ (explained in Sections 5.7 and 5.8, respectively), the double 'ant co-operation' loop is re-entered for time step t = 1. As at time step t = 0, during the first tour of the inner loop the current colony trail intensity matrix (now $\tau_{ij}(1)$) is used in the generation process for $\underline{\alpha}^1$ and $\underline{\alpha}^2$ (first 50% of the colony at $t = 1$). Note that $\underline{\alpha}^3$

and $\underline{\alpha}^4$ are those of the previous time step (i.e. $t=0$) and will have influence on the generation of $\underline{\alpha}^3$ and $\underline{\alpha}^4$ (second 50% of the colony at $t=1$) by the second tour of the inner loop.

Hence in the pursuit of a colony of diverse alpha-vectors, the interplay of TS diversification and local trail updating, in the strategy described above, not only enables ants to co-operate within a particular time step but also between time steps.

The example explained above is for $c_0 = \frac{1}{2}$. The two extremes are $c_0 = \frac{1}{x_{\max}}$ and 1.

With $c_0 = \frac{1}{x_{\max}}$ the inner ant loop is not utilised and a generated alpha-vector is influenced by the alpha-vectors generated on all preceeding outer loop tours at the same time step. Also, since the colony trail intensity matrix is updated $x_{\max}$ times in each time step (compared to twice when $c_0 = \frac{1}{2}$, see equations (5.13) and (5.14)), such extensive use of the local updating rule may diminish the need for the TS diversification component of ant co-operation. Furthermore, as with $c_0 = \frac{1}{2}$, the case $c_0 = \frac{1}{x_{\max}}$ is such that the current colony trail intensity matrix will contain information from the previous time step. Hence the case $c_0 = \frac{1}{x_{\max}}$ gives a particular co-operation strategy with good information sharing between ants with respect to the generation of diverse alpha-vectors.

At the other extreme, i.e. $c_0 = 1$, the outer ant loop is not utilised and there is no co-operation (information exchange) within a time step nor between time steps. Also the local updating rule is applied (to all ants) only once per time step. Similarly for

the updating of the colony trail intensity matrix. Hence, compared to $c_0 = \frac{1}{x_{\max}}$ and $\frac{1}{2}$, the case $c_0 = 1$ results in reduced ant co-operation and a co-operation strategy in which the TS diversification strategies are an important component for alpha-vector diversity.

5.7 Intensification Phase

For the current time step, on exiting the ant co-operation phase, i.e. the outer ant loop (see flowchart in Figure 5.1), the current best (i.e. lowest) minimum distance for each ant, $d^x_{\min}$, $x = 1,2,...,x_{\max}$, and the associated integer vector, $\underline{z}^x_{\min}$, have been obtained by the (TS) diversification phase (see Section 5.4) and recorded. The next procedure in the presented ACS algorithm is the intensification phase.

The intensification phase comprises a single phase of TS moves (m_{phase} moves) applied only to those integer vectors identified for intensification during the diversification phase (see Section 5.4). These particular integer vectors are in 'promising' regions of solution-space which are now investigated further.

Those ants x with integer vector $\underline{z}^x_{\min}$ identified for intensification have a short-term memory tabu list, T^x, $x \in \{1,2,...,x_{\max}\}$, initially comprising the zero vector $\underline{0}$ and the start integer vector $\underline{z}^x_0$, where $\underline{z}^x_0 = \underline{z}^x_{\min}$. A TS phase of m_{phase} moves is then performed to attempt to improve the current value of $d^x_{\min}$. The tabu list grows to a maximum size of (m_{phase} + 2) elements. If, during the intensification phase, an integer vector is encountered with associated minimum distance lower than the

current value of $d^x_{\min}$, then $d^x_{\min}$ is updated to the lower value and recorded, together with its associated integer vector, $\underline{z}^x_{\min}$.

On completion of the intensification phase the presented ACS algorithm has produced a population of minimum distances, $d^x_{\min}$, $x = 1,2,...,x_{\max}$, and associated integer vectors, $\underline{z}^x_{\min}$. For the current time step the overall lowest minimum distance, d_{best}, is then identified, where

$$d_{best} = \min\{ d^x_{\min} \} \quad x = 1,2,...,x_{\max} \tag{5.15}$$

together with the associated integer vector, $\underline{z}_{best}$.

5.8 Global Trail Intensity Update

The final stage in the time step loop is the updating of the colony trail intensity matrix, $\tau_{ij}(t)$. The updating technique is similar to that of the AS algorithm of Chapter 3 (see equations (3.20) to (3.22)) except in the ACS approach only the 'best ant', i.e. the ant x with the lowest minimum distance obtained within the current time step, deposits pheromone, by an amount denoted by $\delta\tau^{best}_{ij}(t)$. The purpose of only using 'best ant' information is, in the subsequent time step, to focus the generation of alpha-vectors in a region of good solutions (alpha-vectors with low minimum distances) and so speed up convergence.

The (global) trail intensity updating rule calculates the colony trail intensity matrix for use in the subsequent time step and is given by

$$\tau_{ij}(t+1) = (1-\rho_G)\tau_{ij}(t)^{(\frac{1}{c_0})} + \rho_G \delta\tau^{best}_{ij}(t) \tag{5.16}$$

In equation (5.16) the pheromone deposit at time step t, $\delta\tau_{ij}^{best}(t)$, is calculated with reference to the alpha-vector, $\underline{\alpha}^{best} = \{\alpha_j^{best}\}$, $j = 1,2,...,k$, associated with the d_{best} given by equation (5.15), in the following manner,

$$\delta\tau_{ij}^{best}(t) = \begin{cases} \dfrac{d_{bound}}{d_{best}} & \text{if } \alpha_j^{best} = i \\ 0 & \text{otherwise} \end{cases} \qquad (5.17)$$

where d_{bound} is the appropriate square root bound for the particular QR code.

The trail intensity matrix for the current time step is $\tau_{ij}(t)^{(\frac{1}{c_0})}$, where c_0 is the colony fraction. For example, if $c_0 = \frac{1}{2}$ then the current trail intensity matrix is $\tau_{ij}(t)^{(2)}$ given by equation (5.14) but at general time step t.

The first term on the right-hand side of equation (5.16) represents the autocatalytic (positive feedback) aspect of the process in which the factor (1- ρ_G) is a trail evaporation factor. The parameter ρ_G is user-specified, $0 < \rho_G < 1$, and acts as a weighting factor for the feedback and pheromone deposit components of the global updating.

The value of d_{best} given by equation (5.15) represents the lowest minimum distance value found during a time step. The current lowest minimum distance found during all time steps so far is d_{best}^* (where $d_{best}^* = d_{best}$ when $t = 0$) and is updated to the value of d_{best} if during a time step $d_{best} < d_{best}^*$.

On leaving the time step loop the current value of d^*_{best} is output together with the associated alpha-vector and codeword, $\underline{\alpha}^*_{best}$ and $\underline{c}^*_{best}$, respectively.

5.9 Minimum Distance Results

The ant colony system algorithm described in Sections 5.3 to 5.8 was used to determine the minimum distances of some large QR codes (p = 137, 167, 191, 193 and 199). The augmented QR codes (*n(x)* and *q(x)*) and expurgated QR codes (*(x-1)n(x)* and *(x-1)q(x)*) were used in this investigation.

The initial trail and desirability matrices for the colony were set to $\tau_{ij}(0) = 10^{-6}$ and $\eta_{ij}(0) = 0.5$, respectively, $i = 0,1$ and $j = 1,2,...,k$. The maximum number of time steps was $s_{\max} = 25$. Since there is 'ant co-operation' in the ant colony system, the number of ants in the colony was reduced from $x_{\max} = k$ (as used in the ant system of Chapter 3) but remained dependent on the size of the problem. Here $x_{\max} = 0.25k$ (or integer part) and all ants are used in the aspects of the developed tabu search algorithm (i.e. diversification or intensification, see Section 5.4 and Section 5.7, respectively).

First the values of the user-specified parameters υ, ρ_L and ρ_G were investigated to obtain the combination most beneficial to the optimisation process. The augmented QR code (*n(x)*) with p = 137 was used as a test code (since this was used in Chapter 3 and Chapter 4) with the following parameter values, a = 1, b = 0 (from Chapter 3) and Q = d_{bound} (upper bound = 21), $c_0 = 0.5$, υ = {0.1, 0.3, 0.5}, and ρ_L = ρ_G = {0.2, 0.3, 0.4}. For each combination of υ and ρ_L (= ρ_G), minimum distance results using the ant colony system algorithm were obtained with 10 different seeds for the random number

generator (random numbers r_1 and $r_2 \in [0,1]$ are used in the state-transition rule, see equations (5.1) and (5.2), respectively).

In order to visualise the analysis of the obtained minimum distance results, the ranges (i.e. $d^+_{best} - d^-_{best}$, where d^+_{best} and d^-_{best} denote the highest and lowest minimum distance value, respectively, obtained by the 10 runs) are shown in Figures 5.5, 5.6 and 5.7 for each combination of υ = (0.1, 0.3, 0.5) and $\rho_L = \rho_G$ = (0.2, 0.3, 0.4).

Inspection of Figures 5.5, 5.6 and 5.7 reveals that υ = 0.3 (i.e. 30% exploitation, hence 70% exploration) gives the least variation (greatest repeatability) in the obtained minimum distances for each value of ρ_L ($= \rho_G$).

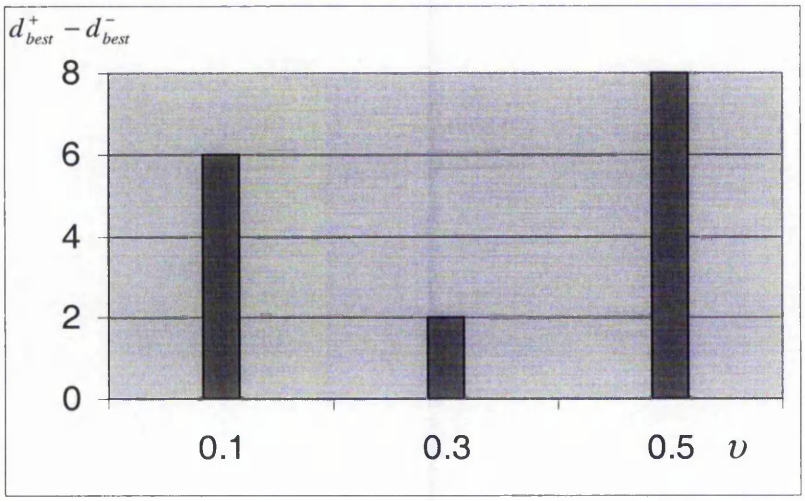


Figure 5.5 Ranges of minimum distance obtained using $\rho_L = \rho_G = 0.2$.

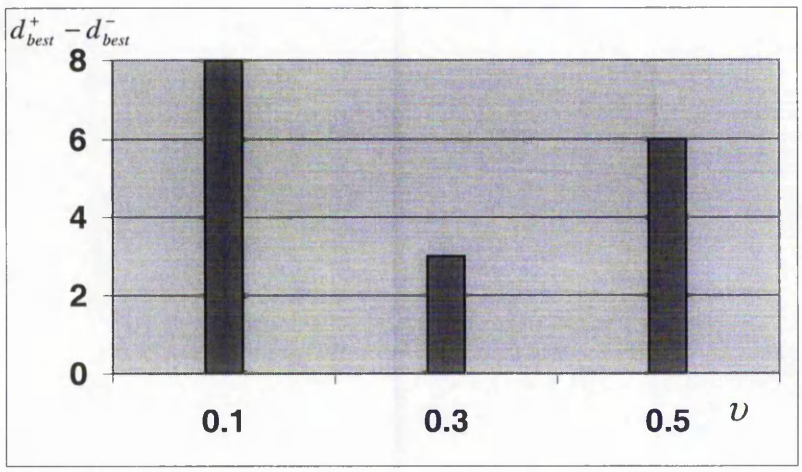


Figure 5.6 Ranges of minimum distance obtained using $\rho_L = \rho_G = 0.3$.

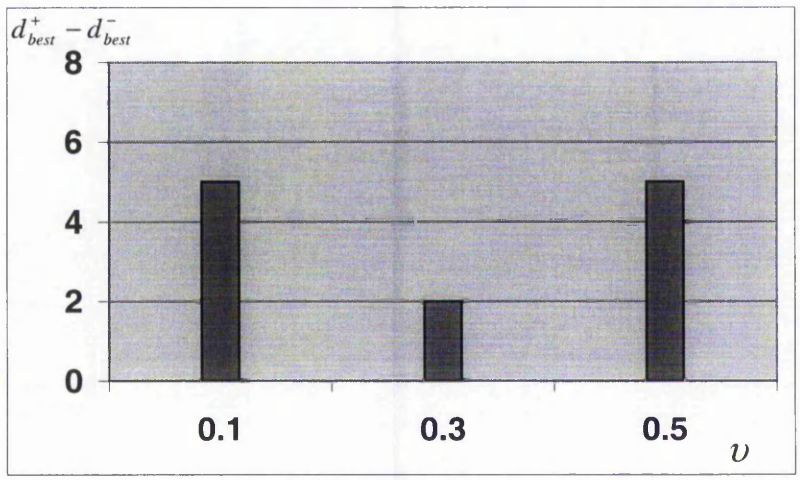
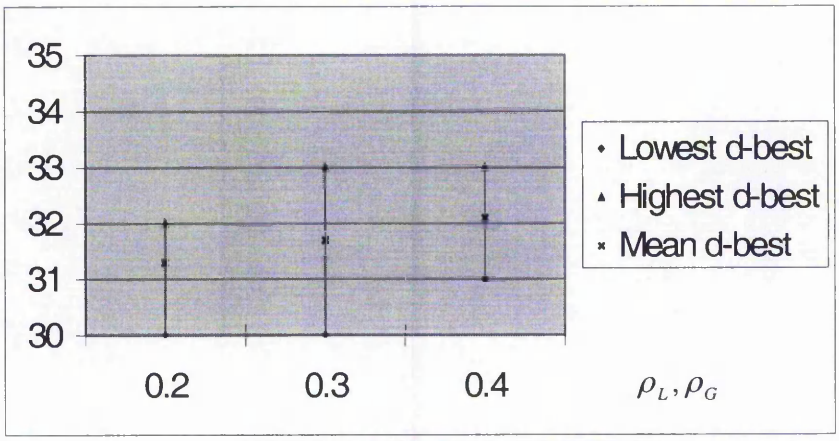


Figure 5.7 Ranges of minimum distance obtained using $\rho_L = \rho_G = 0.4$.

Next, with $\upsilon = 0.3$, actual values of d^+_{best}, d^-_{best} and the mean minimum distance of the 10 runs, $\overline{d}_{best}$, are plotted in the graph in Figure 5.8. As observed in Figure 5.8, $\rho_L = \rho_G = 0.2$ is the best value with the respect to solution quality and repeatability.

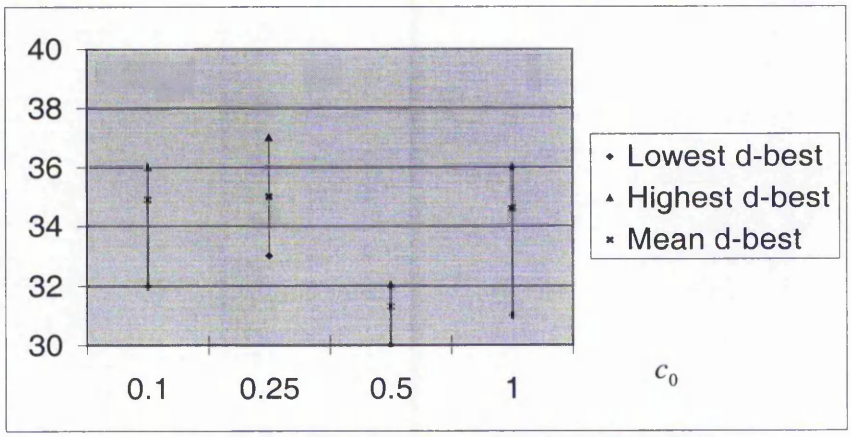
With $\upsilon = 0.3$ and $\rho_L = \rho_G = 0.2$ the (ant co-operation) colony fraction, c_0, was investigated by analysing the minimum distance results obtained by 10 runs (i.e. 10 different random number seeds) of the algorithm. Here (expressed as a decimal) c_0 =

0.1, 0.25, 0.5 and 1.0. The obtained minimum distance results are shown in Figure 5.9 for each value of c_0.



ρ_L, ρ_G	0.2	0.3	0.4
d^-_{best}	30	30	31
d^+_{best}	32	33	33
$\overline{d}_{best}$	31.3	31.7	32.1

Figure 5.8 Minimum distances obtained for p = 137.



c_0	0.1	0.25	0.5	1.0
d^-_{best}	32	33	30	31
d^+_{best}	36	37	32	36
$\overline{d}_{best}$	34.9	35	31.3	34.6

Figure 5.9 Minimum distances obtained for p = 137.

From Figure 5.9, it is clear that with $c_0 = 0.5$, the minimum distances obtained were both of smaller range and lower than with $c_0 = 0.1$, 0.25 or 1. To confirm the parameter settings as $a = 1$, $b = 0$ and $Q = d_{bound}$, $c_0 = 0.5$, $\upsilon = 0.3$ and $\rho_L = \rho_G = 0.2$, the frequency of minimum distances found in the 10 runs are shown in Figure 5.10.

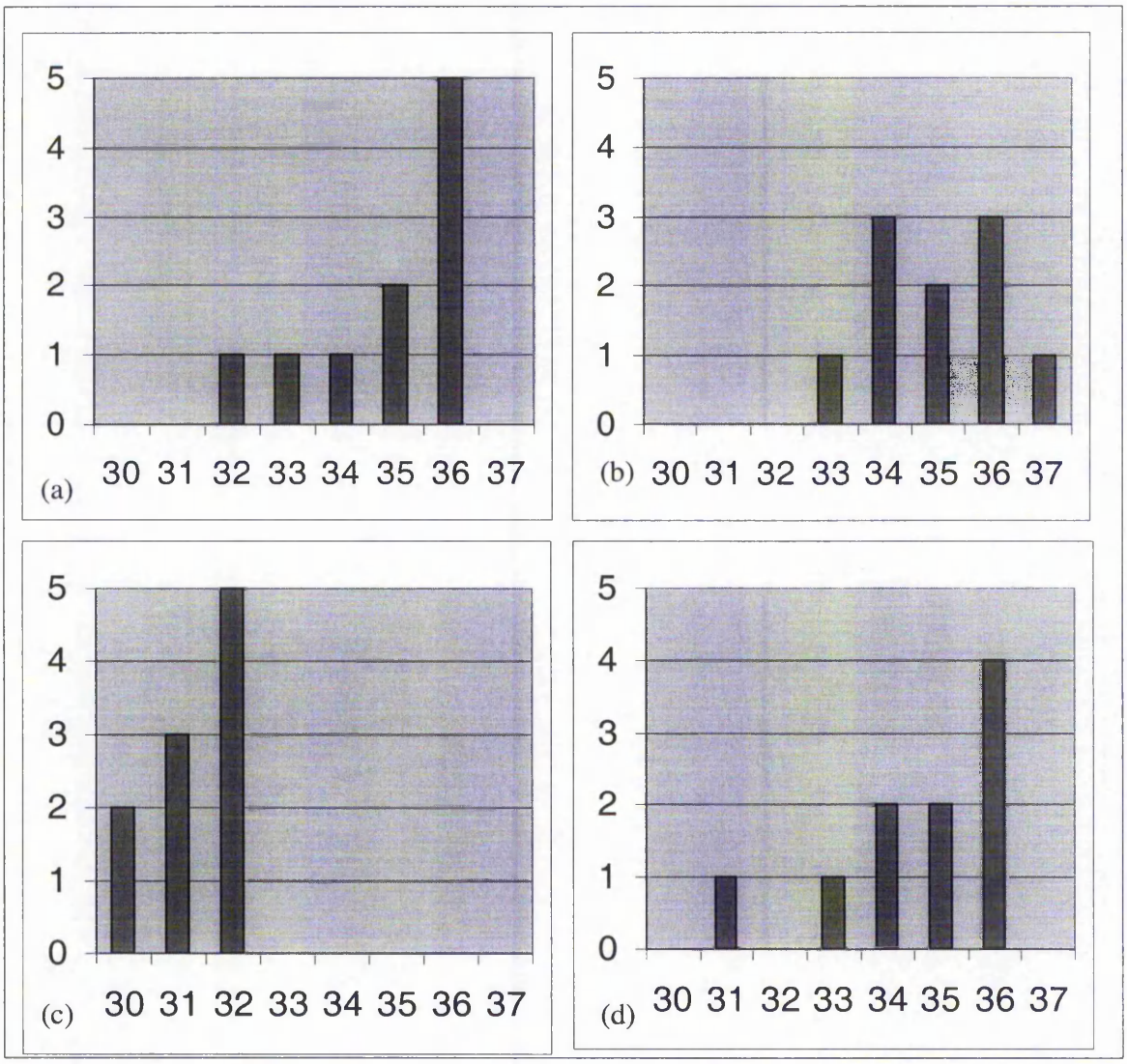


Figure 5.10 Frequency of the obtained minimum distances using different c_0. (a) 0.1, (b) 0.25, (c) 0.5 and (d) 1.

The bar charts in Figure 5.10 (a), (b), (c) and (d) illustrate the effect of ant co-operation, as represented by the value of c_0, on the 10 obtained values of d_{best} for c_0 = 0.1, 0.25, 0.5 and 1.0, respectively. As observed in Figure 5.10, c_0 = 0.5 gives the highest frequency of the lowest d_{best} values.

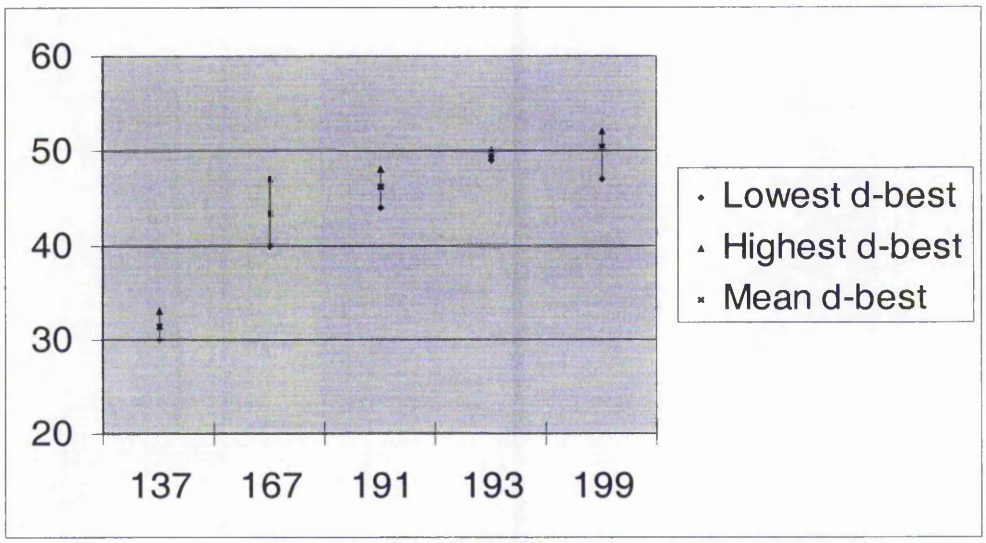
The computational results indicate that with high levels of ant co-operation (i.e. low values of c_0, here c_0 = 0.1 and 0.25) the algorithm finds d_{best}^- values very early in the optimisation process (s_{best}^- = 2 for c_0 = 0.1 and s_{best}^- = 5 for c_0 = 0.25). In this situation the combination of TS diversification strategies and the local updating rule induces relatively high diversity in the colony of alpha-vectors so that the focusing effect of the global updating rule may not have significant influence on the colony trail intensity matrix and hence may not lead to improved results.

With c_0 = 1 (i.e. low level of ant co-operation) the algorithm found d_{best}^- later on in the optimisation process (s_{best}^- = 15; $s_{\max}$ = 25). In this situation there is no sharing of information within and between time steps so that there is a delayed learning process which delays improved results.

With c_0 = 0.5 a balance between the amount of ant co-operation, in terms of diversity of alpha-vectors, as obtained by TS diversification and local updating, and the effect of focusing by the global updating rule, produces relatively good results that may be obtained reasonably quickly (in terms of time steps) and do not suffer from stagnation (8 of the 10 runs obtained d_{best} values with s_{best} between 5 and 24).

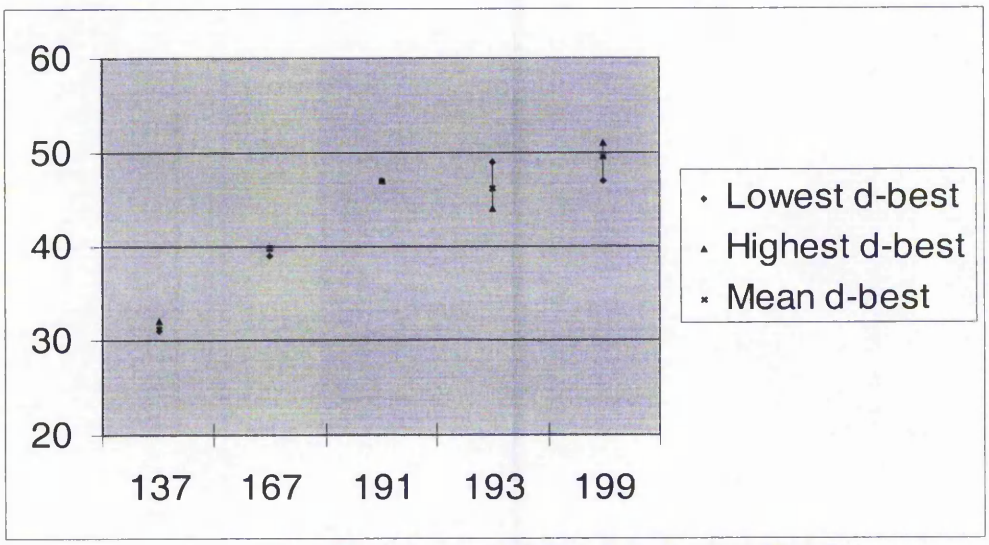
Analysis of the minimum distance results gave the best combination of parameter values (in terms of the obtained solution quality and repeatability) as a = 1, b = 0 and Q

$= d_{bound}$, $\upsilon = 0.3$, $\rho_L = \rho_G = 0.2$ and $c_0 = 0.5$. This combination of parameter values were then used by the ant colony system algorithm to obtain minimum distances for all other QR codes. As with the test code, the ant colony system algorithm was used with 10 different random number seeds for each code (i.e. 10 runs per code).



p	137	167	191	193	199
d_{bound}	21	23	27	27	31
d^{-}_{best}	30	40	44	49	47
d^{+}_{best}	33	47	48	50	52
$\overline{d}_{best}$	31.4	43.4	46.2	49.6	50.4

Figure 5.11 **Minimum distances obtained by the developed ant colony system using $n(x)$ and upper bound.**



p	137	167	191	193	199
d_{bound}	13	15	15	15	15
d^{-}_{best}	31	39	47	44	47
d^{+}_{best}	32	40	47	49	51
$\overline{d}_{best}$	31.3	39.7	47	46.3	49.6

Figure 5.12 **Minimum distances obtained by the developed ant colony system using $n(x)$ and lower bound.**

Results for QR codes with the generator polynomial $n(x)$ and d_{bound} as the upper and lower bound are given in Figure 5.11 and Figure 5.12, respectively. Results for other QR codes are given in Tables C1 to C6 in Appendix C. Computational results were obtained for the larger QR codes only (characterised by p = 137, 167, 191, 193 and 199); the exact minimum distances for the smaller QR codes (p = 71, 79, 97, 103 and 113) have been obtained by previous algorithms.

With reference to Figures 5.11 and 5.12, it is observed that, compared to the ant system results given in Figure 3.8, the developed ant colony system (ACS) may produce better quality results (see d^{-}_{best} values for p = 137, 167 and 199 in Figure 5.11 and d^{-}_{best}

values for p = 137, 167, 193 and 199 in Figure 5.12) and, furthermore, the algorithm gives far greater repeatability.

The overall best minimum distances (i.e. lowest of values of $\bar{d}_{best}$) obtained using the developed ant colony system algorithm are given in Table 5.1. Inspection of Table 5.1 reveals that the improvements gained by the developed ACS over AS are at the expense of very high execution times.

Code	p	$g(x)$	d_{bound}	$\bar{d}_{best}$	$\bar{s}_{best}$	$\bar{t}_{best}$ (h:m:s)
6	137	$(x\text{-}1)n(x)$	14	28	1	0:14:13.38
6	137	$(x\text{-}1)n(x)$	22	28	1	0:10:15.99
7	167	$n(x)$	15	39	10	5:51:13.80
7	167	$q(x)$	15	39	2	0:18:15.42
7	167	$q(x)$	23	39	2	0:21:31.08
8	191	$q(x)$	27	43	7	6:05:14.30
9	193	$n(x)$	15	44	4	5:41:48.50
9	193	$q(x)$	15	44	6	5:40:53.10
9	193	$q(x)$	27	44	6	3:28:06.10
10	199	$(x\text{-}1)n(x)$	32	44	6	1:50:36.48

Table 5.1 ACS results for QR codes.

Chapter 6

Conclusions

6.1 Summary

This thesis has investigated and developed two optimisation techniques, namely, tabu search (TS) and ant colony optimisation (ACO) to determine minimum distance estimates of error-correcting codes.

The minimum distance, d, is the smallest weight of the non-zero codewords comprising a linear code C and, in terms of computational effort, requires $|C|-1$ evaluations. A consequence of the linear nature of code C is that $\underline{c}=\underline{\alpha}G$, where codeword $\underline{c}\neq\underline{0}$, G is the generator matrix of dimension $k\times n$ and $\underline{\alpha}$ is a k-tuple. Furthermore $|C|=2^k$, so that when k is large the determination of d by complete enumeration is not practical. In this thesis values of $k\sim 10^2$ are investigated and the minimisation problem is combinatorial. The combinatorial nature of the determination of the minimum distance of a linear code gives rise to difficulties in the optimisation procedure. In the first instance this combinatorial optimisation problem is NP-complete, so the computation time quickly becomes prohibitive. Hence heuristic solution techniques become necessary. A second difficulty is the inability of most heuristic methods to determine global optima of non-convex (multiple optima) problems.

The particular linear codes studied were Bose-Chaudhuri-Hocquengham (BCH) and quadratic residue (QR) codes. BCH codes were used because the exact minimum distances of these codes are known and may be used as benchmark values for the

heuristic optimisation algorithms. In contrast the exact minimum distances of large QR codes are not known but have lower and upper bounds [Macwilliams and Sloane 1977].

In order to investigate the minimum distance problem (MDP) of various QR codes the mathematical derivation of the generator matrices was required. The Coding Theory algebra for generator matrices for QR codes was developed in Chapter 2. One form of generator matrix was obtained by cyclic shift of the first row of a codeword in which each codeword corresponding to a row of G has weight equal to $\frac{p-1}{2}$. Hence, for larger QR codes this generator matrix will have rows with a relatively large weight. With large initial weight it was difficult to search for low weight codewords in a feasible time. Another form of G was then developed, using the algebra of a polynomial representation of the codewords, in which the weight of the codewords in G have significantly lower weights than the generator matrix with weight $\frac{p-1}{2}$. Use of this developed generator matrix gave improved results.

Tabu search (TS) is a recent heuristic optimisation technique that has the ability to avoid entrapment by local optima and hence search for global optima. Using the derived generator matrices for QR codes this thesis presents the development of an effective and efficient TS algorithm to obtain minimum distances. Following Bland and Baylis (1995), a basic TS algorithm was first implemented to find minimum distances for different QR codes. The computational results show that with low weight generator matrices, the basic TS algorithm was able to deliver reasonable results in short execution times (see Chapter 3). In addition, a distributed processing algorithm called ant system was presented. Ant system is a basic ant colony optimisation algorithm (ACO) which uses information obtained by a number of

individual agents (computational 'ants') to form a population-based optimisation algorithm. The ant system does not search by means of a 'path' of successive solutions in an appropriate solution-space, rather, it aims to progressively improve an aid to decision-making so that, at each iteration, better decisions may be made (i.e. ones that lead to better objective function values). Although the ant system has positive-feedback as its main optimisation mechanism, better quality results may be obtained when used with tabu search as a local search improvement phase (i.e. the use of tabu search for local improvement is beneficial to the optimisation process and, therefore, will be used in the developed algorithm).

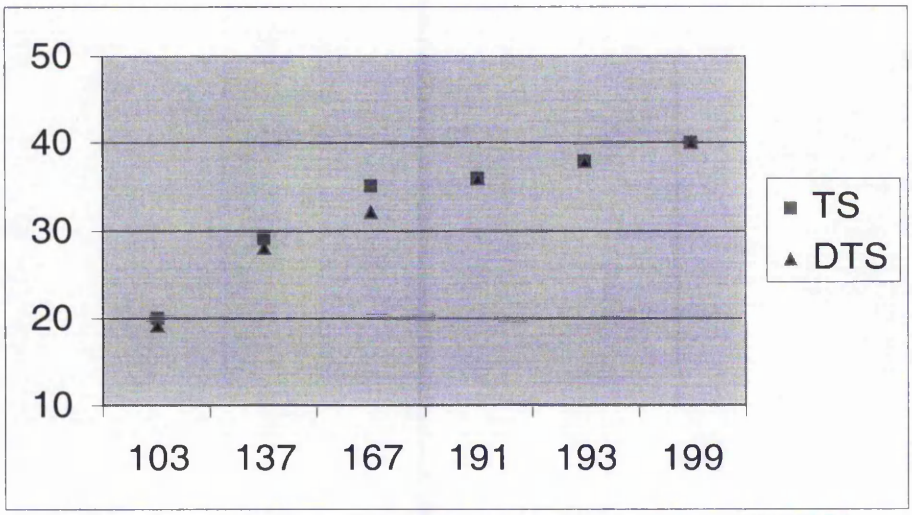
p	d_{QR}	d_{TS}	t_{TS}	$g(x)$
71	11	**11**	0:00:01.05	$n(x)$
79	15	**15**	0:00:00.99	$n(x)$
97	15	**15**	0:00:04.79	$q(x)$
103	20	**20**	0:00:00.93	$(x-1)\ n(x)$
113	12-16	**16**	0:00:00.22	$(x-1)\ n(x)$

Table 6.1 Best results of the basic tabu search algorithm.

The overall best results using the basic tabu search algorithm with the smaller QR codes (i.e. p = 71, 79, 97, 103 and 113) are presented in Table 6.1. With reference to Table 6.1, d_{QR} denotes the square root bound results, d_{TS} and t_{TS} denote the obtained overall best minimum distance and the associated execution time, respectively. Inspection of Table 6.1 shows that the basic tabu search algorithm was able to find the exact minimum distances (indicated by bold type). This indicates that with the use of relatively low weight generator matrices (i.e. augmented and expurgated QR codes), this algorithm is able to successfully tackle relatively small size problems.

To enhance the basic TS algorithm to tackle the larger problems (i.e. p = 137, 167, 191, 193 and 199), longer-term strategies and a number of 'memory' lists were developed within a TS algorithm (explained in Chapter 4). The features and strategies

include a two-way conversion mechanism, an influential candidate list, a dynamic tabu list, diversification (first-time, most-improved, equal-best and random-start) and intensification strategies. Like hashing functions the two-way conversion reduces the memory storage, however, unlike hashing functions, it has the capability of converting integers to binary alpha-vectors and vice versa. The influential candidate list and the dynamic tabu list were used to guide the search to explore for longer periods. Different levels of diversification and intensification strategies enabled the appropriate memory to be 're-activated' so that the search did not require the excessive memory that would be needed to store its complete history. Results show that improvements may be obtained with the use of these features and strategies.



p	d_{TS}	t_{TS}	d_{DTS}	t_{DTS}
137	29	0:00:24.06	28	0:01:23.87
167	35	0:00:02.91	32	0:04:41.00
191	36	0:00:19.72	36	0:00:19.72
193	38	0:00:01.31	38	0:00:01.26
199	40	0:00:08.40	40	0:00:08.40

Figure 6.1 Best results of the basic and the developed tabu search algorithms using *n(x)*.

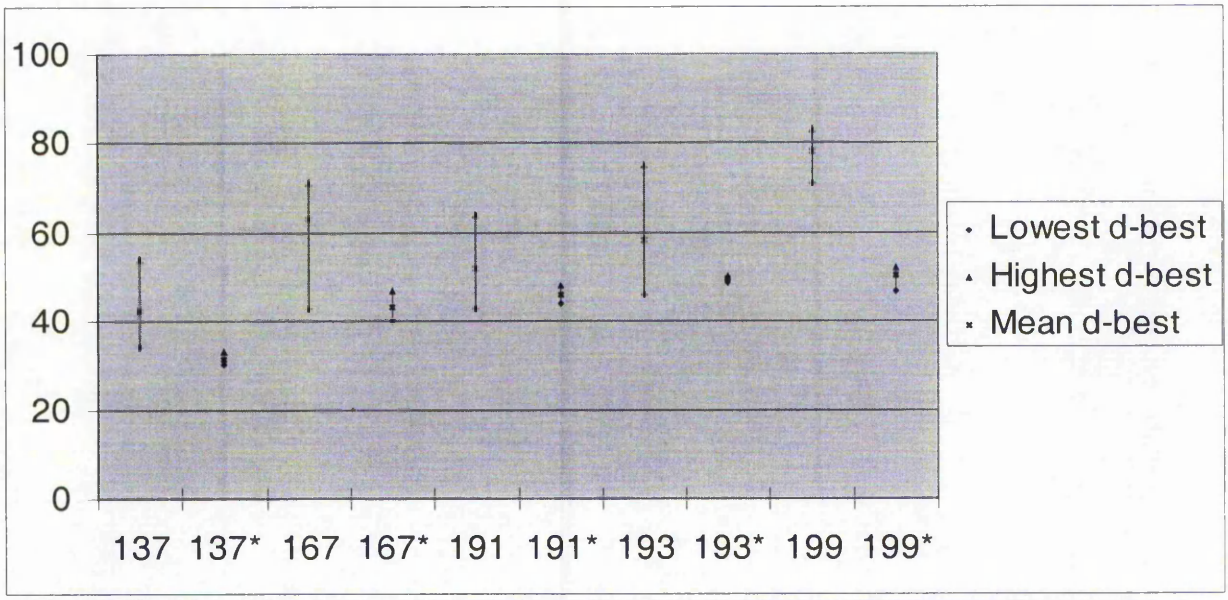
A comparison of the minimum distance results (with *n(x)*) for the larger problems obtained using the basic tabu search algorithm (TS) and the developed tabu search

algorithm (DTS), is presented in Figure 6.1. With reference to the table in Figure 6.1, d_{TS} and t_{TS} denote the best minimum distances and the execution times using the basic tabu search algorithm (see Chapter 3) and d_{DTS} and t_{DTS} denote the best minimum distances and the execution times using the developed tabu search algorithm (see Chapter 4). The results show that there are improvements for p = 137 and 167. Even with the use of more moves and longer-term strategies, the search process was still able to obtain d_{DTS} in a very fast execution time. However, the optimisation process was unable to find any improvements for some of the larger QR codes (i.e. those with p = 191, 193 and 199).

In the ant system explained in Chapter 3 the computational results show that the standalone AS algorithm produced reasonable results but improvements are possible when used in conjunction with a tabu search local improvement phase (used in the developed algorithm of Chapter 5). Another type of ACO algorithm is the ant colony system (ACS), with its aim to improve the 'intelligence' of the ant system (AS). The ACS (presented in Chapter 5) enhanced the capability of AS by using a state-transition rule that incorporated both exploration and exploitation. The local trail intensity update rule was formulated so that its use with some TS diversification strategies allowed ants to share the knowledge of the colony (i.e. ant co-operation) and avoid stagnation. The global trail intensity update rule was such that only the best ant was allowed to deposit pheromone, which enables the optimisation process to focus on the neighbourhood of the best quality alpha-vectors.

Figure 6.2 shows the ranges of the obtained minimum distances using the ant system (AS) of Chapter 3 and the ant colony system (ACS) of Chapter 5, for *n(x)*. Results for the ant colony system are indicated by * on the horizontal axis in the graph in Figure 6.2. Inspection of Figure 6.2 reveals that the developed ACS algorithm is

able to produce minimum distance values that are lower than those obtained using the AS algorithm (i.e. for p = 137, 167, 193 and 199). Also, it is noticeable that the ranges of the ACS minimum distances have low variation (based on 10 runs using 10 different random seed values). The developed ACS algorithm is superior to the AS algorithm in terms of repeatability of results.



p	d^-_{best} (AS)	d^+_{best} (AS)	$\bar{d}_{best}$ (AS)	d^-_{best} (ACS)	d^+_{best} (ACS)	$\bar{d}_{best}$ (ACS)
137	34	54	42.5	30	33	31.4
167	43	71	63	39	40	39.7
191	43	64	52.1	44	48	46.2
193	46	75	58.5	44	49	46.3
199	71	83	78	47	51	49.6

Figure 6.2 Results of the ant system and the ant colony system using $n(x)$.

As a final comparison, the best minimum distances (using $n(x)$) and associated execution times for all algorithms investigated in this thesis, are presented in Table 6.2 and Table 6.3, respectively. In Table 6.2, d^-_{AS} and d^-_{ACS} denote the lowest minimum distances obtained out of 10 runs of the AS and ACS algorithm, respectively, and the associated execution times are denoted by t^-_{AS} and t^-_{ACS}, respectively, in Table 6.3.

p	d_{QR}	d_{TS}	d_{DTS}	$\bar{d}_{AS}$	$\bar{d}_{ACS}$
137	13-21	29	28	34	30
167	15-23	35	32	43	39
191	15-27	36	36	43	44
193	15-27	38	38	46	44
199	15-31	40	40	71	47

Table 6.2 Best results for the MDP using $n(x)$.

p	t_{TS} $(h{:}m{:}s)$	t_{DTS} $(h{:}m{:}s)$	$\bar{t}_{AS}$ $(h{:}m{:}s)$	$\bar{t}_{ACS}$ $(h{:}m{:}s)$
137	0:00:11.26	0:00:43.17	0:05:03.19	3:46:33.40
167	0:00:01.81	0:03:06.92	0:09:35.67	5:51:13.80
191	0:00:02.31	0:01:09.75	0:06:15.47	3:05:14.30
193	0:00:01.31	0:01:04.26	0:11:46.84	5:41:48.50
199	0:00:08.40	0:01:08.05	0:07:22.37	3:05:23.18

Table 6.3 Best Execution times for the MDP.

Inspection of Table 6.2 reveals that the most successful algorithm, in terms of the obtained minimum distance estimates, is the developed tabu search algorithm (DTS).

A disadvantage of using an ACO algorithm (AS and ACS), rather than a tabu search algorithm (TS and DTS), is the relatively large number of user-specified parameters that are required to be 'tuned' to the particular problem under investigation. Also, this type of algorithm is stochastic, which may make its use less attractive for practical reasons (because of repeatability problems) when compared to a deterministic technique such as tabu search. As a final (practical) consideration, compared to tabu search, ACO algorithms require much longer execution times (see Table 6.3).

Possible modifications to the developed ACS algorithm aimed at improving solution quality without (significantly) increasing the execution times are suggested in Sections 6.3 and 6.4.

For all the QR codes investigated in this thesis, a summary of the obtained lowest minimum distances that are closest to the exact (or upper bound) value (d_{QR}), and

have the fastest execution time, are given in Table 6.4. All minimum distances (d_{best}) and execution times (t_{best}) reported in Table 6.4 were obtained by the developed tabu search algorithm Obtained exact minimum distances are indicated by bold type.

p	d_{QR}	d_{best}	t_{best}	$g(x)$
71	11	**11**	0:00:01.05	$n(x)$
79	15	**15**	0:00:00.99	$n(x)$
97	15	**15**	0:00:04.79	$q(x)$
103	19	**19**	0:00:48.01	$n(x)$
113	12-16	**16**	0:00:00.22	$(x-1)\ n(x)$
137	13-21	27	0:00:43.17	$q(x)$
167	16-24	32	0:01:09.26	$(x-1)\ q(x)$
191	15-27	35	0:02:25.56	$q(x)$
193	16-28	38	0:00:01.21	$(x-1)\ n(x)$
199	15-31	40	0:00:08.05	$q(x)$

Table 6.4 Overall best results.

6.2 Achievements

The main aim of this study is to investigate the use and application of heuristic optimisation techniques to determine minimum distance estimates of error-correcting codes (BCH and QR codes). In order to investigate the minimum distance of (augmented and expurgated) QR codes, the required research has been presented and discussed in the previous chapters. The main achievements are now stated.

- **Development of the Generator Matrices for QR Codes**

In order to investigate the MDP for error-correcting codes (e.g. BCH and QR codes) the generator matrix, G, is required. The minimum distances for BCH codes are known and were used in this thesis as benchmarks for the obtained minimum distances. The generator matrix for this class of codes was that used by Bland and Baylis (1995). However, for QR codes, the exact minimum distances are unknown, particularly for the larger codes investigated in this thesis. For QR codes the generator matrix was required to be developed mathematically in a way that could be adapted into

a computational algorithm that produced an appropriate generator matrix, G for a given prime number, p.

In this thesis the mathematical development of generator matrices for QR codes used the algebra of a polynomial representation of codewords. The mathematical theory was then converted to a general algorithm that was able to automatically produce the generator matrices for both augmented and expurgated QR codes for any appropriate value of p.

- **Adaptation of the Ant System to the Minimum Distance Problem**

A feature of the ant system is the trace intensity matrix in which information is held for decision-making and its contents represents the level of 'pheromone' deposited by a colony of computational ants. The trace intensity matrix is updated each iteration with the information obtained during the iteration so that decisions that lead to improved objective function values may be made and, like tabu search, algorithm termination at local optima may be avoided.

In this thesis the ant system algorithm was adapted and used in a Coding Theory context. The computational ants produced a colony of alpha-vectors $\underline{\alpha} = \{\alpha_j\}$, $j = 1,2,...,k$, by probabilistically assigning bit state $i \in \{0,1\}$ to bit number $j \in \{1,2,...,k\}$ to form (i, j) couplings. The 'cost function' values for the MDP are the codeword weights, $w(\underline{\alpha} G)$, and the required minimum distance is d^*, where d^* $= \min\{w(\underline{\alpha} G)\}$.

- **Development of the Tabu Search Algorithm**

In this thesis, to enhance the basic TS algorithm, a two-way conversion has been formulated to convert alpha-vectors (binary strings) to integers with the aim of reducing the size of the tabu list and to improve the search efficiency (explained in

Chapter 4). Also, the developed algorithm incorporated candidate list strategies such as the influential candidate list and the dynamic tabu list, which enabled the search to memorise certain important moves for long periods, so that the search did not require excessive memory to store its complete history. Based on the square root bound for QR codes [Macwilliams and Sloane 1977] the closeness criterion utilised the given bound as a threshold value to establish the quality of the search which was then used to identify optimisation strategies such as diversification and intensification. For intensification, the influential candidate list was used as a 'backtracking' mechanism to guide the search to return to desirable regions and, for diversification, the influential candidate was treated as a 'penalty' to force the search into new regions. These two strategies enabled the search to both explore the search-space and exploit the search process.

- **Development of the Ant Colony System**

The ant colony system (ACS) is an algorithm within the general framework of ACO which is designed to improve decision-making (compared to AS) through the use of a more general decision rule which incorporates the processes of exploration and exploitation. The development of ACS uses a particular form of local trail updating to allow ants to modify their individual trail intensities and produce a diversity of solutions so that ants explore different solutions. In addition, a global trail updating rule was used to guide ants to focus on the selection of good solutions so that ants learn as time progresses. The main feature of the developed ACS is 'ant co-operation'. Ant co-operation is an interplay strategy between local trail updating (of ACS) and diversification (of tabu search) which enables ants to share information (i.e. co-operate) so that ants learn and produce a diversity of solutions to avoid stagnation.

6.3 Possible Improvements

In this section some possible improvements to the developed TS and ACS algorithms are presented. The following modifications have no conceptual basis in terms of the backgrounds of TS and ACS; they are purely computational features, which, when included in the developed algorithms may produce better results for the codes of this study.

For the developed ACS algorithm the following modifications are a consequence of observations that the alpha-vectors associated with the lowest minimum distances are such that the elements are predominantly in state 0.

- **Exploration**

Based on the observations concerning alpha-vector elements, the following modifications to the exploration case of the state-transition rule (see equation (5.2)) is designed to increase the chances of an element of an alpha-vector being in state 0 when it has the greater chance of being in state 1. In other words, if, for ant x at time t, $P_{1j}^{x}(t) > P_{0j}^{x}(t)$, $j \in \{1,2,...,k\}$, then $P_{0j}^{x}(t)$ is slightly increased.

The probability that an alpha-vector element α_j, $j \in \{1,2,...,k\}$ has bit state 0 is given by equation (5.6) (with x and t omitted)

$$P_{0j} = \frac{1}{1+\lambda_j^a \varepsilon_j^b} \tag{6.1}$$

where

$$\lambda_j \equiv \frac{\tau_{1j}}{\tau_{0j}} \text{ and } \varepsilon_j \equiv \frac{\eta_{1j}}{\eta_{0j}} \tag{6.2}$$

in which τ_{ij} and η_{ij}, $i = 0,1$, $j = 1,2,...,k$, denote trail intensity and desirability, respectively.

A typical graph of P_0 against λ (omitting j) for equation (6.1) is given in Figure 6.3 and is shown as a continuous curve.

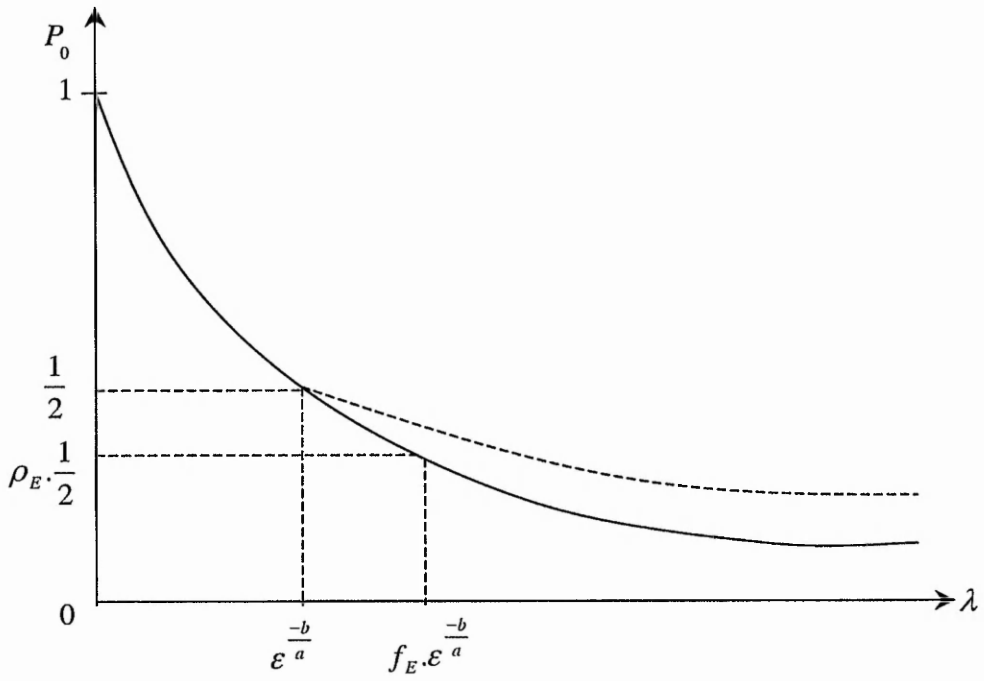


Figure 6.3 Graph of P_0 against λ .

When $P_1 > P_0$, that is, by equation (3.18), $P_0 < \frac{1}{2}$, equation (6.1) leads to $\lambda > \varepsilon^{\frac{-b}{a}}$ (see Figure 6.3). Hence, to increase the chance of $\alpha_j = 0$, $j \in \{1,2,...,k\}$, in cases where $P_1 > P_0$ (i.e. $\lambda > \varepsilon^{\frac{-b}{a}}$), the local updating rule is used with $\alpha_j = 1$ (see equation 5.5);

$$\lambda_j = \frac{\lambda_j}{1+D_j} \tag{6.3}$$

where

$$D_j \equiv \frac{\rho_L \Delta\tau_j}{(1-\rho_L)\tau_{0j}} \tag{6.4}$$

and

$$\Delta\tau_j \equiv |\tau_{0j} - \tau_{1j}| \tag{6.5}$$

Hence, for ant x at time step t, in the construction of alpha-vector $\underline{\alpha}^x = \{\alpha_j^x\}$, $j = 1,2,...,k$, the modified state-transition rule for exploration becomes

$$\alpha_j^x = \begin{cases} 0 & \text{if } r \le P_{0j}^x(t) \\ 1 & \text{otherwise} \end{cases} \tag{6.6}$$

where random number $r \in [0,1]$ and

$$P_{0j}^x(t) = \begin{cases} \dfrac{1}{1+\lambda_j^a \varepsilon_j^b} & \text{if } \lambda_j \le \varepsilon_j^{\frac{-b}{a}} \\ \dfrac{1}{1+\left(\dfrac{\lambda_j}{1+D_j}\right)^a \varepsilon_j^b} & \text{otherwise} \end{cases} \tag{6.7}$$

A typical graph of P_0 against λ (omitting j, x and t) for the equation,

$$P_0 = \frac{1}{1+\left(\dfrac{\lambda}{1+D}\right)^a \varepsilon^b}, \quad \lambda > \varepsilon^{\frac{-b}{a}} \tag{6.8}$$

is shown in Figure 6.3 as a broken curve.

Inspection of the graph in Figure 6.3 shows that for values of λ such that $\lambda > \varepsilon^{\frac{-b}{a}}$ (i.e. $P_1 > P_0$), P_0 (modified) given by equation (6.8) is greater than P_0 (unmodified) given by equation (6.1).

- **Exploitation**

As with stochastic exploration, the modification to the deterministic exploitation case of the state-transition rule is designed to increase the possibility of an alpha-vector element being in state 0 when it should be in state 1.

From equation (5.3), i.e. exploitation, for $\underline{\alpha}^x = \{\alpha_j^x\}$, $j = 1,2,...,k$, and ant x at time step t,

$$\alpha_j^x = 1 \text{ if } P_{0j}^x(t) < \frac{1}{2} \tag{6.9}$$

Hence, with reference to Figure 6.3, the range of λ such that $\alpha_j^x = 1$ is chosen is given by

$$\lambda > \varepsilon^{\frac{-b}{a}} \tag{6.10}$$

The following modification is designed to decrease the range of λ that will result in $\alpha_j^x = 1$, in other words, increase the range of λ that will result in $\alpha_j^x = 0$. The modified state-transition rule for exploitation is taken to be

$$\alpha_j^x = \begin{cases} 0 & \text{if } P_{0j}^x(t) > \rho_E \cdot \frac{1}{2} \\ 1 & \text{otherwise} \end{cases} \tag{6.11}$$

Where user-specified parameter ρ_E is such that $0 < \rho_E < 1$. Hence, by equation (6.11) the range of λ such that $\alpha_j^x = 1$ is chosen is now

$$\lambda > \left(\frac{2 - \rho_E}{\rho_E} \right)^{\frac{1}{a}} \varepsilon^{\frac{-b}{a}} \tag{6.12}$$

In other words the range of λ that results in $\alpha_j^x = 0$ has increased by a factor f_E (see Figure 6.3), where

$$f_E \equiv \left(\frac{2-\rho_E}{\rho_E} \right)^{\frac{1}{a}} > 1 \qquad (6.13)$$

- **Random-Start Diversification**

The Random-Start diversification strategy is a component of both the developed TS and ACS algorithm. The proposed modification is designed to perform a 'grape shot' in $\underline{\alpha}$-space using candidate random-start integer vectors in which the associated alpha-vectors have many zero entries.

If random-start integer vector, $\underline{z}_r$, is such that

$$\underline{z}_r = \{z_1, z_2, ..., z_B\} \qquad (6.14)$$

where z_i, $i = 1,2,..., B$, are integer blocks, then a further B start integer vectors may be formed by 'exploding' $\underline{z}_r$ in the following manner,

$$\begin{aligned} \underline{z}_{r1} &= \{ z_1, 0, ..., 0 \} \\ \underline{z}_{r2} &= \{ 0, z_2, ..., 0 \} \\ &\vdots \\ &\vdots \\ \underline{z}_{rB} &= \{ 0, 0, ..., z_B \} \end{aligned} \qquad (6.15)$$

Hence in the modified Random-Start diversification strategy, for a randomly generated start integer vector, $\underline{z}_r$, further candidate start integer vectors, $\underline{z}_{ri}$, $i = 1,2,..., B$, given by equation (6.15), are considered. The actual start integer vector selected for Random-Start diversification is the one with the associated lowest minimum distance.

6.4 Further Research

- **Code-based Research**

In this study, the minimum distances for various binary QR codes were investigated. Although the exact minimum distances for these codes are not known for the larger codes, researchers have been able to find ranges of these minimum distances in which the minimum distance estimates could be calculated by using the square root bound. With the rapid growth of computer technology, it is more efficient to investigate minimum distances of codes computationally. The computational results obtained in this thesis for large size QR codes indicate that heuristic optimisation techniques are able to deliver good minimum distance estimates in reasonable computation time, and for smaller size QR codes, the search was able to find minimum distances equal to or very close to the square root bound results. An area for further research is the investigation of some ternary QR codes [Higgs and Humphreys 1995] as they have the rich algebraic structure and, to date, the minimum distances for even small size codes are still unknown.

- **Algorithm-based Research**

In the ant colony system, there are many ways to improve or to change the ACS approach. Unlike the TSP or routing problems where these problems have many states to choose from, the MDP chooses between two particular states, 0 and 1, so a major aim is to avoid stagnation. Ants were discouraged from choosing the same alpha-vectors by using a local trail intensity updating rule to reduce the chances of stagnation. There are different ways to formulate the local trail intensity updating rule, for example, for those ants that generate alpha-vectors with high distance values, the

pheromone evaporation parameter ρ_L may be increased so that the poor solutions (i.e. high distance values) will become less attractive. Another change to the ACS could be to allow ants to decide when the local trail intensity updating rule should be used. If ants generate alpha-vectors that are different from each other, yet their corresponding distance values are of good quality, then the local trail intensity updating rule could be de-activated until the quality of the distance values obtained begins to decline.

Apart from the local trail intensity updating rule, the formulation of the global trail intensity updating rule may also be modified. In ant system all ants contribute pheromone to the next time step, while the ACS uses only the ant that obtained the lowest distance (i.e. the best ant) to deposit pheromone for the next time step. It was observed there is often more than a single ant that generates the same distance values (with different alpha-vectors); the first encountered was used. An alternative strategy to investigate would be to allow all best ants to deposit pheromone so that the search procedure will not be focused by a single best alpha-vector, but a set of equally best alpha-vectors.

Appendix A

Table A1, A2 and A3 give the minimum distances, d_{best}, and associated move numbers and execution times, m_{best} and t_{best}, respectively, for various QR codes using tabu search (explained in Chapter 3).

Code	p	d_{QR}	$w(G)$	d_{best}	m_{best}	t_{best} $(h:m:s)$
1	71	11	15	15	2	0:00:00.04
2	79	15	23	16	9	0:00:00.24
3	97	15	17	15	96	0:00:04.79
4	103	19	27	20	21	0:00:03.46
5	113	11-15	33	21	24	0:00:04.89
6	137	13-21	39	29	2	0:00:00.71
7	167	15-23	43	35	2	0:00:01.81
8	191	15-27	47	36	31	0:00:21.53
9	193	15-27	43	40	1	0:00:00.41
10	199	15-31	51	43	4	0:00:04.23

Table A1 Tabu search results using $q(x)$.

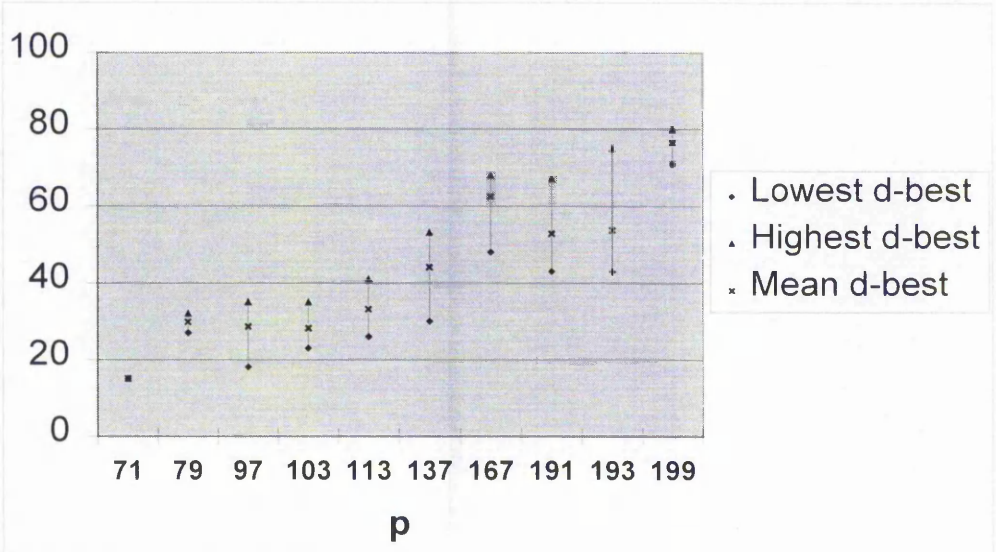
Code	p	d_{QR}	$w(G)$	d_{best}	m_{best}	t_{best} $(h:m:s)$
1	71	12	16	12	20	0:00:00.40
2	79	16	20	16	15	0:00:00.41
3	97	16	22	16	37	0:00:04.28
4	103	20	24	20	7	0:00:00.93
5	113	12-16	26	16	1	0:00:00.22
6	137	14-22	34	36	45	0:00:11.26
7	167	16-24	56	40	11	0:00:05.99
8	191	16-28	48	40	1	0:00:01.21
9	193	16-28	42	38	1	0:00:01.21
10	199	16-32	60	44	2	0:00:01.98

Table A2 Tabu search results using $(x - 1)n(x)$.

Code	p	d_{QR}	$w(G)$	d_{best}	m_{best}	t_{best} $(h:m:s)$
1	71	12	16	12	34	0:00:00.68
2	79	16	20	16	14	0:00:00.34
3	97	16	30	22	1	0:00:00.05
4	103	20	24	20	7	0:00:01.26
5	113	12-16	26	20	1	0:00:00.28
6	137	14-22	34	28	57	0:00:16.42
7	167	16-24	56	36	3	0:00:02.25
8	191	16-28	48	36	3	0:00:02.31
9	193	16-28	42	40	29	0:00:19.83
10	199	16-32	60	44	22	0:00:16.70

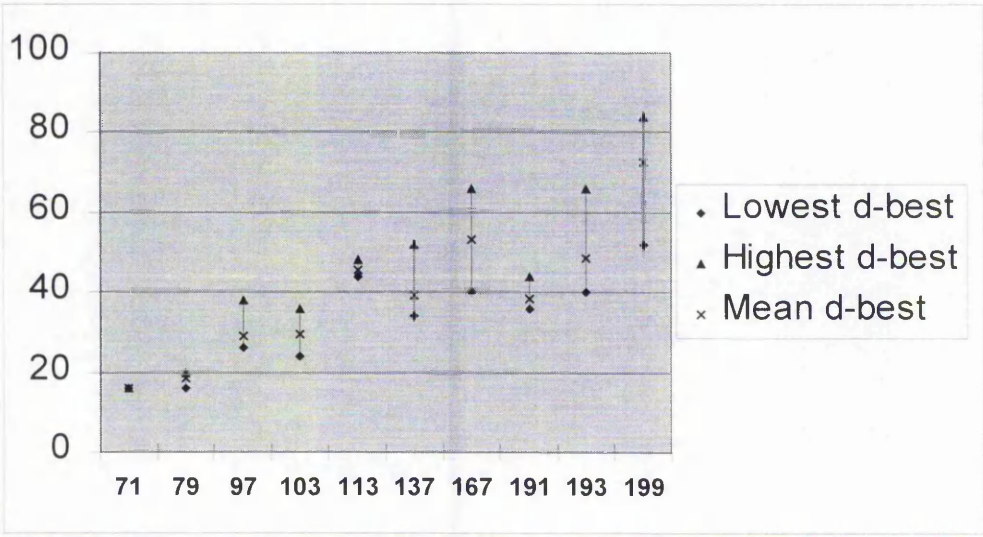
Table A3 Tabu search results using *(x - 1)q(x)*.

Figures A1, A2 and A3 give the lowest, highest and mean value (d^{-}_{best}, d^{+}_{best} and $\overline{d}_{best}$, respectively) of the minimum distances obtained with 10 runs of the ant system (explained in Chapter 3) for various QR codes.



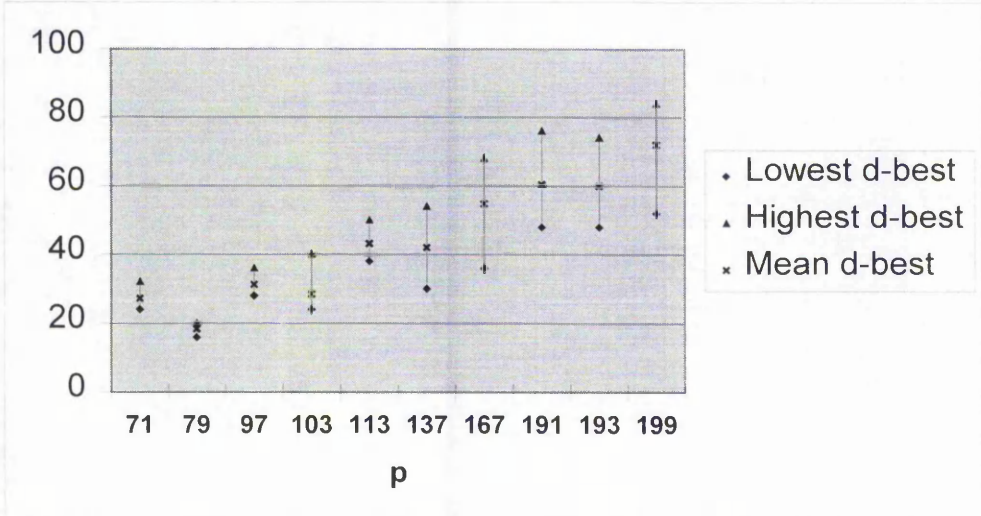
P	71	79	97	103	113	137	167	191	193	199
d_{QR}	11	15	15	19	11-15	13-21	15-23	15-27	15-27	15-31
d^{-}_{best}	15	27	18	23	26	30	48	43	43	71
d^{+}_{best}	15	32	35	35	41	53	68	67	75	80
$\overline{d}_{best}$	15	29.9	28.6	28.2	33.2	44.1	62.5	52.8	53.7	76.6

Figure A1 Minimum distances obtained by ant system using *q(x)*.



	71	79	97	103	113	137	167	191	193	199
d_{QR}	12	16	16	20	12-16	14-22	16-24	16-28	16-28	16-32
d^{-}_{best}	16	16	26	24	44	34	40	36	40	52
d^{+}_{best}	16	20	38	36	48	52	66	44	66	84
$\overline{d}_{best}$	16	18.4	29	29.6	45.6	39.2	53.2	38.4	48.4	72.4

Figure A2 Minimum distances obtained by ant system using *(x-1)n(x)*.



p	71	79	97	103	113	137	167	191	193	199
d_{QR}	12	16	16	20	12-16	14-22	16-24	16-28	16-28	16-32
d^{-}_{best}	24	16	28	24	38	30	36	48	48	52
d^{+}_{best}	32	20	36	40	50	54	68	76	74	84
$\overline{d}_{best}$	27.2	18.4	31.2	28.4	43.2	42	54.8	60.4	60	72

Figure A3 Minimum distances obtained by ant system using *(x-1)q(x)*.

Appendix B

The tables in this appendix (Tables B1 to B8) give the minimum distances (d_{best}) obtained by the developed tabu search algorithm presented in Chapter 4, together with the associated move (m_{best}) and execution times (t_{best}), for various QR codes (characterised by prime number p). For each type of QR code; those with generator matrix with weight = $\frac{p-1}{2}$, augmented QR codes ($q(x)$) and expurgated QR codes ($(x-1)n(x)$ and $(x-1)q(x)$), results are presented when both the lower bound and upper bound value of the minimum distance (d_{bound}) is used in the closeness criterion.

p	d_{bound}	d_{best}	m_{best}	t_{best} $(h:m:s)$
71	11	19	549	0:00:21.69
79	15	23	567	0:01:32.60
97	15	30	492	0:01:29.75
103	19	31	10	0:00:01.37
113	11	34	649	0:02:05.33
137	13	46	28	0:00:09.18
167	15	56	729	0:05:42.02
191	15	67	193	0:02:20.28
193	15	70	219	0:02:59.55
199	15	71	313	0:04:16.39

Table B1 Minimum distances obtained using $w(G)=\frac{p-1}{2}$ and lower bound.

p	d_{bound}	d_{TS}	m_{best}	t_{best} $(h:m:s)$
71	11	19	549	0:00:21.69
79	15	23	567	0:01:32.60
97	15	30	492	0:01:29.75
103	19	31	10	0:00:01.37
113	15	34	649	0:02:05.33
137	21	46	28	0:00:09.18
167	23	56	729	0:05:42.02
191	27	67	193	0:02:20.28
193	27	70	219	0:02:59.55
199	31	71	313	0:04:16.39

Table B2 Minimum distances obtained using $w(G)=\frac{p-1}{2}$ and upper bound.

p	d_{bound}	d_{best}	m_{best}	$t_{best}(h{:}m{:}s)$
71	11	11	413	0:00:06.21
79	15	15	213	0:00:04.78
97	15	15	96	0:00:04.39
103	19	20	21	0:00:03.46
113	11	21	24	0:00:04.89
137	13	29	2	0:00:00.71
167	15	32	381	0:03:06.92
191	15	36	31	0:00:21.53
193	15	38	110	0:01:04.26
199	15	40	107	0:00:08.05

Table B3 Minimum distances obtained using $q(x)$ and lower bound.

p	d_{bound}	d_{best}	m_{best}	$t_{best}(h{:}m{:}s)$
71	11	11	413	0:00:06.21
79	15	15	213	0:00:04.78
97	15	15	96	0:00:04.39
103	19	20	21	0:00:03.46
113	15	21	24	0:00:05.38
137	21	27	154	0:00:43.17
167	23	35	2	0:00:01.81
191	27	35	233	0:02:25.56
193	27	40	1	0:00:00.41
199	31	43	4	0:00:04.23

Table B4 Minimum distances obtained using $q(x)$ and upper bound.

p	d_{bound}	d_{best}	m_{best}	$t_{best}(h{:}m{:}s)$
71	12	12	24	0:00:08.83
79	16	16	15	0:00:0066
97	16	16	69	0:00:03.55
103	20	20	7	0:00:00.93
113	12	16	1	0:00:00.22
137	14	28	45	0:00:11.26
167	16	36	11	0:00:05.99
191	16	36	109	0:01:09.75
193	16	38	1	0:00:01.21
199	16	44	2	0:00:01.98

Table B5 Minimum distances obtained using $(x-1)\,n(x)$ and lower bound.

p	d_{bound}	d_{best}	m_{best}	$t_{best}(h{:}m{:}s)$
71	12	12	24	0:00:08.83
79	16	16	15	0:00:0066
97	16	16	69	0:00:03.55
103	20	20	7	0:00:00.93
113	16	16	1	0:00:00.22
137	22	28	45	0:00:11.26
167	24	36	11	0:00:05.99
191	28	36	210	0:02:27.20
193	28	38	1	0:00:01.21
199	32	44	2	0:00:01.98

Table B6 Minimum distances obtained using $(x-1)\,n(x)$ and upper bound.

p	d_{bound}	d_{best}	m_{best}	$t_{best}(h{:}m{:}s)$
71	12	12	34	0:00:00.83
79	16	16	13	0:00:00.60
97	16	20	643	0:00:23.01
103	20	20	7	0:00:01.26
113	12	20	1	0:00:00.28
137	14	28	57	0:00:16.42
167	16	32	164	0:01:09.26
191	16	36	3	0:00:02.31
193	16	40	29	0:00:19.83
199	16	40	136	0:01:38.59

Table B7 Minimum distances obtained using $(x-1)\,q(x)$ and lower bound.

p	d_{bound}	d_{best}	m_{best}	$t_{best}(h{:}m{:}s)$
71	12	12	34	0:00:00.83
79	16	16	13	0:00:00.60
97	16	20	643	0:00:23.01
103	20	20	7	0:00:01.26
113	16	20	1	0:00:00.22
137	22	28	57	0:00:16.42
167	24	36	3	0:00:02.25
191	28	36	3	0:00:02.31
193	28	40	29	0:00:19.83
199	32	40	324	0:04:37.65

Table B8 Minimum distances obtained using $(x-1)\,q(x)$ and upper bound.

Appendix C

Tables C1 to C6 give the lowest, highest and mean value (d^{-}_{best}, d^{+}_{best} and $\overline{d}_{best}$, respectively) of the minimum distances obtained with 10 runs of the developed ant colony system algorithm (explained in Chapter 5) for various large QR codes.

p	137	167	191	193	199
d_{bound}	13	15	15	15	15
d^{-}_{best}	29	39	47	44	52
d^{+}_{best}	31	39	48	47	55
$\overline{d}_{best}$	30.5	39	47.3	45.7	52.5

Table C1 **Minimum distances obtained by ant colony system using *q(x)* and lower bound.**

p	137	167	191	193	199
d_{bound}	21	23	27	27	31
d^{-}_{best}	29	39	43	44	48
d^{+}_{best}	31	39	48	47	52
$\overline{d}_{best}$	30.6	39	44.8	45	50.6

Table C2 **Minimum distances obtained by ant colony system using *q(x)* and upper bound.**

p	137	167	191	193	199
d_{bound}	14	16	16	16	16
d^{-}_{best}	28	44	48	48	52
d^{+}_{best}	28	44	52	48	56
$\overline{d}_{best}$	28	44	44	48	53.3

Table C3 **Minimum distances obtained by ant colony system using *(x-1)n(x)* and lower bound.**

p	137	167	191	193	199
d_{bound}	22	24	28	28	32
d^{-}_{best}	28	44	48	48	44
d^{+}_{best}	30	48	48	48	44
$\overline{d}_{best}$	28.4	46.4	48	48	44

Table C4 **Minimum distances obtained by ant colony system using *(x-1)n(x)* and upper bound.**

p	137	167	191	193	199
d_{bound}	14	16	16	16	16
d^{-}_{best}	32	44	48	46	52
d^{+}_{best}	32	44	52	46	52
$\overline{d}_{best}$	32	44	49.7	46	52

Table C5 **Minimum distances obtained by ant colony system using *(x-1)q(x)* and lower bound.**

p	137	167	191	193	199
d_{bound}	22	24	28	28	32
d^{-}_{best}	32	44	48	46	52
d^{+}_{best}	32	44	48	54	52
$\overline{d}_{best}$	32	44	48	48.4	52

Table C6 **Minimum distances obtained by ant colony system using *(x-1)q(x)* and upper bound.**

References

1. Aarts EHL and Laarhoven V (1992). Local search in coding theory. *Discrete Mathematics*, **106/107**, 11-18.

2. Augot D, Charpin P and Sendrier N (1992). Studying the locator polynomial of minimum weight codewords of BCH codes. *IEEE Transactions on Information Theory*, **38**, 960-973.

3. Augot D and Levy-dit-Vehel F (1996). Bounds on the minimum distance of the duals of BCH codes. *IEEE transactions on information theory*, **42**, 4, 1257-1260.

4. Backhouse PG, Fotheringham AF and Allan G (1997). A comparison of a genetic algorithm with an experimental design technique in the optimization of a production process. *Journal of the Operational Research*, **48**, 247-254.

5. Balas E (1965). An additive algorithm for solving linear programs with zero-one variables. *Operations Research*, **13**, 517-546.

6. Battiti R and Protasi M (1995). Reactive local search for the maximum clique problem. *Technical Report*. TR-95-052, ICSI, 1947 Center St.- Suite 600 - Berkeley, California.

7. Battiti R and Tecchiolli G (1994). The Reactive Tabu Search. *ORSA Journal on Computing*, **6**, 126-140.

8. Battiti R and Tecchiolli G (1995). The continuous reactive tabu search: Blending combinatorial optimisation and stochastic search for global optimisation. *Annals of Operations Research*, **63**, 153-188.

9. Battiti R and Tecchiolli G (1995a). Training neural nets with the reactive tabu search. *IEEE Transactions on Neural Networks,* **6**, 1185-1200.

10. Battiti R and Tecchiolli G (1995b). Local search with memory: Bench-marking RTS. *Operations Research Spektrum*, **17**, 67-86.

11. Baykasoglu A, Owen A and Gindy N (1999). Solution of goal programming models using a basic taboo search algorithm. *Journal of the Operational Research Society*, **50**, 960-973.

12. Baylis DJ (1998). *Error-correcting codes.* Chapman and Hall Publishers, London.

13. Beckers R, Deneubourg JL and Goss S (1992). Trail laying behaviour during food recruitment in the ant Lasius niger (L). *Ins Soc*, **39**, 59-72.

14. Beckers R, Deneubourg JL and Goss S (1992a). Trails and u-turns in the selection of a path by the ant lasius niger. *J. theor. Biol*, **159**, 397-415.

15. Berlekamp E, McEliece R and Van Tilborg H (1978). On the inherent intractability of certain coding problems. *IEEE Trans. Inform. Theory*, **IT-24**, 384-386.

16. Ben-Daya M and Al-Fawzan M (1998). A tabu search approach for the flow shop scheduling problem. *European Journal of Operations Research*, **109**, 88-95.

17. Bland JA (1998). Structural design optimization with reliability constraints using tabu search. *Eng., Opt.*,**30**, 55-74.

18. Bland JA (1998a). A memory-based technique for optimal structural design, *Engineering applications of artificial intelligence*, **11**, 319-325.

19. Bland JA (1999). Layout of facilities using an ant system approach. *Eng. Opt.*, **32**, 101-115.

20. Bland JA and Baylis DJ (1995). A tabu search approach to the minimum distance of error-correcting codes. *Int. J. Electronics*, **79**, 829-837.

21. Bland JA and Baylis DJ (1997). Modelling constant weight codes using tabu search. *Appl. Math. Modelling*, **21**, 667-672.

22. Bland JA and Dawson GP (1991). Tabu search and design optimisation, *Computer-aided Design*, **23**, 195-201.

23. Bland JA and Dawson GP (1994). Large-scale layout of facilities using a heuristic hybrid algorithm. *Appl. Math. Modelling*, **18**, 500-503.

24. Bonabeau E, Dorigo M and Theraulaz G (1999). Swarm Intelligence – From natural to artificial systems. Oxford University Press, New York.

25. Bullnheimer B, Kotsis G and Strauss C (1998). Parallelization strategies for the ant system. In: R. De Leone, A. Murli, P. Pardalos, G. Toraldo (eds.), High Performance Algorithms and Software in Nonlinear Optimization; series: *Applied Optimization*, **24**, Kluwer:Dordrecht, 87-100.

26. Bullnheimer B, Hartl F and Strauss C (1997). A new rank-based version of the ant system: a computational study. Technical Report POM-03/97, Institute of Management Science, University of Vienna. Accepted for publication in the Central European Journal for Operations Research and Economics.

27. Camazine S (1991). Self-organizing pattern formation on the combs of honey bee colonies. *Behavioral Ecology and Sociobiology*, **28**, 61-76.

28. Carlton W and Barnes JW (1995). A note on hashing functions and tabu search algorithms. *European Journal of Operational Research*, 237-239

29. Carlton W and Barnes JW (1996). Solving the travelling-salesman problem with time windows using tabu search. *IIE transactions*, **28**, 617-629.

30. Cawsey A (1998). The essence of artificial intelligence. Prentice Hall Europe.

31. Černy V (1985). Thermodynamical approach to the travelling salesman problem: An efficient simulation algorithm. *J Optimisation Theory Application,* **45**, 41-51.

32. Chan E (1998). Discrete Optimisation in coding theory, *Nottingham Trent University transfer report.*

33. Chen X, Reed IS and Truong TK (1994). Decoding the (73,37,13) quadratic residue code. *IEEE Proc-Comput. Digit. Tech.,* **141**, 253-258.

34. Chiang WC and Kouvelis P (1996). An improved tabu search heuristic for solving facility layout design problems. *Int J Prod. Res.,* **34**, 2565-2585.

35. Cho J-H and Kim Y-D (1997). A simulated annealing algorithm for resource constrained project scheduling problems. *Journal of Operational Research Society*, **48**, 736-744.

36. Colorni A, Dorigo M and Maniezzo V (1991). Distributed Optimization by Ant Colonies, Proceeding of European Conference on Artifical Life, Paris, France, Elsevier Publishing 134-142.

37. Colorni A, Dorigo M and Maniezzo V (1992). An investigation of some properties of an "ant algorithm". *Elsevier Science Publishers B.V*, 509-520.

38. Colorni A, Dorigo M, Maffioli F, Maniezzo V, Righini G and Trubian M (1999). Heuristics from nature for hard combinatorial optimization problems. *International Transactions in Operational Research*, **3**, 1, 1-21.

39. Coppersmith D and Seroussi G (1984). On the minimum distance of some quadratic residue codes. *IEEE Transactions on Information Theory,* **30**, 407-411.

40. Corne D, Dorigo M and Glover F (1999). New ideas in optimization. McGraw-Hill Publishing Company, England.

41. Costa D and Hertz A (1997). Ants can colour graphs. *Journal of the Operational Research Society*, **48**, 295-305.

42. Dakin RJ (1965). A tree search algorithm for mixed integer programming problems, *Computer J*, **8**, 250-255.

43. Dorigo M and Caro GD (1999). The ant colony optimization meta-heuristic. *To appear in D Corne, M Dorigo and F Glover, editors, New ideas in optimization.* CGraw-Hill, 1999.

44. Dorigo M and Gambardella LM (1996). A study of some properties of ant-q. Proceedings of PPSN IV-Fourth international conference on parallel problem solving from nature, H-M. Voigt, W. Ebeling, I, Rech

45. Dorigo M and Gambardella LM (1997). Ant Colony System: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on evolutionary computation*, **1**, 1, 53-66.

46. Dorigo M, Caro GD and Gambardella LM (1999a). Ant algorithms for discrete optimisation. *To appear in Artificial life, MIT Press.*

47. Dorigo M, Maniezzo V and Colorni A (1991). Ant System: An autocatalytic optimizing process. Technical Report *91-016, Diparimento di Elettronica e Informazione, Politecnico di Milano, Piazza Leonardo da Vinci 32, 20133 Milano, Italy.*

48. Dorigo M, Maniezzo V and Colorni A (1996). The ant system: optimisation by a colony of co-operating agents. *IEEE Trans. Systems, Man and Cybernetics,* **26**, 1-13.

49. George FAW (1996). Hybrid Genetic Algorithms with Immunisation to Optimise Networks of Retail Outlets. *Studies in Locational Analysis*, **8**.

50. Feng GL and Tzeng KK (1994). A new procedure for decoding cyclic and BCH codes up to actual minimum distance. *IEEE Transactions on information theory,***40**, 5, 1364-1374.

51. Forsyth P and Wren A (1997). An Ant System for Bus Driver Scheduling. University of Leeds technical report.

52. Foulds LR (1981). Optimization techniques. Springer-Verlag Inc, New York.

53. Foulds LR (1984). Combinatorial Optimization for Undergraduates. Springer-Verlag Inc, New York.

54. Gambardella LM and Dorigo M (1995). Ant-Q: A reinforcement learning approach to the TSP, *Proceedings of ML-95, Twelfth Intern. Conf .on machine learning, Morgan Kaufamn,* 252-260.

55. Gambardella LM, Taillard ED and Dorigo M (1999). Ant colonies for the QAP. *Journal of the Operational Research Society*, **50**, 167-176.

56. Garey MR and Johnson DS (1979). *Computers and Intractability: A Guide to the Theory of NP-completeness*, W.H. Freeman, San Francisco.

57. Glover F (1989). Tabu Search – Part I. *ORSA Journal on Computing,* **1**, 190-206.

58. Glover F (1990). Tabu Search – Part II. *ORSA Journal on Computing*, **2**, 4-32.

59. Glover F (1990a). Tabu Search – A Tutorial. *Interfaces*, **20**, 4, 74-94.

60. Glover F and Hubscher R (1994). Applying TS with influential diversification to multiprocessors scheduling. *Computers Ops. Res.,* **21**, 877-884.

61. Glover F and Laguna M (1997). *Tabu Search.* Kluwer Academic Publishers, Boston.

62. Glover F, Kochenberger GA, Alidaee B (1998). Adaptive memory Tabu Search for binary quadratic programs. *Institute for Operations Research and the Management Sciences*, **44**, 3, 336-344.

63. Grassé P (1959). La reconstruction du nid et les coordinations interindividuelles chez bellicositermes natalensis et cubitermes sp. La théorie de la stigmergie: essai d'interprétation du comportement des termites constructeurs. *Insectes Sociaux*, 6,41-89.

64. Hang Y (1993). Efficient priority-first search maximum-likelihood soft-decision decoding of linear block codes. *IEEE Transactions on Information Theory,* **39**, 1514-1523.

65. Hertz A, Taillard E and Werra DE (1997). Tabu Search. *Local Search in Combinatorial Optimisation.* 121-136. J Wiley and sons.

66. Hifi M (1997). A genetic algorithm-based heuristic for solving the weighted maximum independent set and some equivalent problems, *Journal of the Operational Research*, **48**, 612-622.

67. Higgs RJ and Humphreys JF (1995). Decoding the ternary (23,12,8) quadratic residue code. *IEEE Transactions on information theory,* **142**, 3, 129-134.

68. Hill R (1986). *A first course in coding theory.* Oxford University Press, Oxford.

69. Hoffman DG, Leonard DA, Lindner CC, Phelps KT, Rodger CA and Wall JR (1991). *Coding theory – the essentials.* Marcel Dekker Publishers, New York.

70. Holzmann GJ (1998). An analysis of bitstate hashing. *Formal methods in system design*, **13**, 289-307.

71. Hübscher R and Glover F (1994). Applying tabu search with influential diversification to multiprocessor scheduling. *Computers Ops Res.*, **21**, 8, 877-884.

72. Islam A and Eksioglu (1997). A tabu search approach for the single machine mean tardiness problem. *Journal of the Operational Research*, **48**, 751-755.

73. Kaelbling LP, Littman ML and Moore AW (1996). Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, **4**, 237-285.

74. Kanal LN and Lemmer JF (1986). Uncertainty in artificial intelligence. Elsevier Science Publishers B.V., North-Holland, Oxford, Amsterdam.

75. Kelly JP, Laguna M and Glover F (1994). A study of diversification strategies for the Quadratic Assignment Problem. *Computers Ops Res.*, **21**, 8, 885-893.

76. Kim JK and Hahn SG (1998). A new upper bound for binary codes with minimum distance four. *Discrete mathematics*, **187**, 1-3, 291-295.

77. Kincaid RD and Laba KE (1998). Reactive Tabu Search and Sensor Selection in Active Structural Acoustic Control Problems, *Journal of Heuristics*, **4**, 199-220.

78. Kirkpatrick S, Gelett CD and Vecchi MP (1983). Optimisation by simulated annealing. *Science*, **220**, 671-680.

79. Laguna M (1992). Tabu Search Primer. *Graduate School of Business and Administration, Campus Box 419, University of Colorado at Boulder, Boulder, CO 80309-0419.*

80. Laguna M, Barnes JW and Glover F (1991). Tabu search methods for a single machine scheduling problem. *J. Intelligent Manuf.*, **2**, 63-74.

81. Larranaga P, Kuigpers CMH, Murga PH, Inza I and Dizdarevic S (1999). Genetic algorithm for the travelling salesman problem: A review of representations and operators. *Artificial Intelligence Review*, **13**, 2, 129-170.

82. Lidl R and Niederreiter Harald (1986). Introduction to finite fields and their applications. *Cambridge University Press.*

83. Lim MH, Yuan Y and Omatu S (2000). Efficient genetic algorithms using simple genes exchange local search policy for the quadratic assignment problem. *Computational Optimization and Applications*, **15**, 3, 248-268.

84. Lin S (1970). An introduction to error-correcting codes. Prentice-Hall Inc., Englewood Cliffs, New Jersey.

85. Linial N and Sasson ORI (1998). Non-expansive hashing. *Combinatorica*, **18**, 1, 121-132.

86. Logendran R and Sonthinen (1997). A tabu search-based approach for scheduling job-shop type flexible manufacturing systems. *Journal of the Operational Research Society*, **48**, 264-277.

87. Macwilliams FJ and Sloane NJA (1977). *Theory of error-correcting codes.* Elsevier Science Publishers, Amsterdam.

88. Matsuo T, Araki Y and Imamura K (1997). Relations between several minimum distance bounds of binary cyclic codes. *IEEE transactions,* **E80-A**, **11**, 2253-2255.

89. Metropolis, N, Rosenbluth A, Rosenbluth M, Teller A and Teller E (1953). Equation of state calculations by fast computing machines. *Journal of Chem. Physics*, **21**, 1087-1092.

90. Moccellin JV and Nagano MS (1998). Evaluating the performance of tabu search procedures for flow shop sequencing. *Journal of the Operational Research*, **49**, 1296-1302.

91. Mühlenbein H (1989). Parallel genetic algorithms, Population genetics and combinatorial optimization, in Schaffer (ed.). *Proceedings of the third international conference on genetic algorithms, Morgan Kaufmann.*

92. Ohlemüller M (1997). Tabu search for large location-allocation problems. *Journal of the Operational Research Society*, **48**, 745-750.

93. Östergård PRJ (1997). Constructing covering codes by Tabu Search. *Journal of combinatorial design,* **5**, 1, 71-80.

94. Papadimitriou CD and Steiglitz K (1982). *Combinatorial Optimization: Algorithms and Complexity*, Prentice-Hall, Englewood Cliffs, NJ.

95. Pless V (1989). *Introduction to the theory of the error-correcting codes – 2^{nd} ed.* John Wiley and Sons Publishers.

96. Pless V (1996). Cyclic codes and Quadratic Residue Codes over Z_4. *IEEE Transactions on Information Theory*, **42**, 5, 1594-1600.

97. Pretzel O (1992). *Error correcting codes and finite fields.* Oxford University Press, Oxford.

98. Reeves CR (1993) *Modern Heuristic Techniques for Combinatorial Problems.* Blackwell Scientific Press, Oxford.

99. Roman Steven (1992). Coding and information theory. Springer-Verlag publishers, New York.

100. Roman Steven (1997). Introduction to coding and information theory. Sopringer-Verlag publishers, New York.

101. Rosen K (1986). *Elementary number theory and its applications.* Addison-Wesley Publishers, Canada.

102. Roux O, Fonlupt C, Talbi EG and Robilliard D (1999). AnTabu, *technical Report of Université du Littoral, BP 719, 62228 Calais, France.*

103. Sewell MJ (1987). Maximum and Minimum Principles – A unified approach, with applications, Cambridge University Press.

104. Skorin-Kapov J (1990). Tabu search applied to the quadratic assignment problem. *ORSA Journal on Computing*, **2**, 33-45.

105. Skorin-Kapov J (1994). Extensions of a Tabu Search adaptation to the quadratic assignment problem. *Computers Ops Res,* **21**, 855-865.

106. Song L and Vanelli A (1992). A VLSI placement method using tabu search. *Microelectronics*, **23**, 167-172.

107. Sriskandarajah C, Jardine AKS and Chan CK (1998). Maintenance scheduling of rolling stock using a genetic algorithm. *Journal of the Operational Research*, **49**, 1130-1145.

108. Stützle T and Hoss H (1997). Improvements on the ant-system: introducing the max-min ant system. *Proceedings of the International Conference on Artificial Neural Networks and Genetic Algorithms.* Page 245-49, Springer Verlag, Vienna.

109. Stützle T and Hoos H (1998). Max-Min Ant System and Local Search for the Traveling Salesman Problem. In S Voss, S. Martello, I.H Osman and C Roucairol, editors, *Meta-heuristics Advances and Trends in Local Search Paradigms for Optimization.* Page 313-329. Kluwer Academics, Boston.

110. Srivastava B (1998). An effective heuristic for minimising makespan on unrelated parallel machines. *Journal of the Operational Research Society*, **49**, 886-894.

111. Taillard E (1991). Robust taboo search for the quadratic assignment problem. *Parallel Comput.*, **17**, 443-455.

112. Vermani LR (1996). *Elements of algebraic coding theory.* Chapman and Hall publishers, India.

113. Voss S (1997). Optimisation by strategically solving feasibility problems using tabu search. *Modern Heuristics for Design Support, Unicom, Uxbridge,* 29-47.

114. Welsh D (1988). *Codes and Cryptography*. Oxford Publisher.

115. Whittle P (1983). Optimization over time – dynamic programming and stochastic control, Volume II, John Wiley & Son Ltd,Chichester.

116. Wilhelm M and Ward T (1987). Solving Quadratic Assignment Problems by 'Simulated Annealing'. *IIE Transactions*, 107-119.

117. Wodrich M and Bilchev G (1997). Cooperative distributed search: The ant's way. *Control and Cybernetics*, **26**, 3, 413-445.

118. Woodruff DL and Zemel E (1993). Hashing vectors for tabu search. *Annals of Operations Research*, **44**, 123-138.

119. Xu J, Chiu SY and Glover F (1996). Tabu Search for dynamic routing communications network design. *Graduate School of Business, University of Colorado at Boulder, CO 80309-0419.*

120. Zachariasen M, Dam M (1996). Tabu search on the geometric traveling salesman problem. *Meta-heuristics. Theory and Applications,* Kluwer, Boston, 571-587.

121. Zhang M and Ma F (1994). Simulated annealing approach to the minimum distance of error-correcting codes. *International Journal of Electronics,* **76**, 377-384.