School of Science and Technology
Nottingham Trent University
Clifton Lane
Nottingham, NG11 8FN

Baldwinian-based meta-heuristic for robust engineering optimisation

Ralph Krause
May 2015

A dissertation submitted in partial fulfillment of the requirements for the degree of Master of Philosophy.

Abstract

The aim of this research was to identify problems in engineering optimisation and then to develop novel solutions to the identified problems. First, principles of computational optimisation were studied and a literature review was conducted. It emerged that the latest research in the area of automated engineering design optimisation tends to combine different optimisation algorithms to improve either effectiveness or efficiency. Three basic types of such hybrid configurations were identified: nested algorithms, sequential algorithms and meta-optimiser. Two problems were then identified that inexperienced practitioners encounter when trying to apply computational optimisation to real-world engineering problems. The first is the problem of parameter tuning and the second is the problem of finding robust solutions. A well-known engineering design problem, the pressure vessel problem, was selected as a case study. A problem of engineering optimisation is that the theoretical solutions have to be implemented in the physical world using manufacturing processes, which have a limited accuracy. If an optimum is too narrow or located too close to a constraint, slight deviations from that location will result in a dramatic drop in fitness. In the real-world this could have catastrophic consequences for practical engineering applications, such as designing a bridge. To overcome these problems, a Baldwinian-based meta-heuristic (BMH) was proposed. As well as identifying the fitness of a solution, it also probes its neighbourhood in order to estimate the goodness of the region of the solution. This meta-heuristic can be combined with any arbitrary optimisation algorithm. SASS was chosen because it only has one control parameter to tune, which makes it most suitable to overcome the problems with parameter tuning. It was shown that BMH/SASS was able to outperform standard SASS as well as particle swarm optimisation (PSO) and hybrid particle swarm branch-and-bound (HPB). In conclusion, it can be said that the new method proposed in this work has the potential to find more robust solutions for engineering optimisation applications.

Table of content

Table of figures

Abbreviations

ACO	ant colony optimisation
ACS	ant colony system
AIS	artificial immune system
AS	ant system
BB	branch and bound
BMH	Baldwinian-based meta-heuristic
CI	computational intelligence
CNC	computerized numerical control
CRO	chemical reaction optimization
DE	differential evolution method
EP	evolutionary programming
FSA	fish school algorithm
GA	genetic algorithm
GSA	gravitational search algorithm
HC	hill climbing algorithm
HGSO	hybrid glow worm swarm optimization algorithm
HPB	hybrid particle swarm branch-and-bound
IGSO	improved glow-worm swarm optimization
L-GPS	genetic particle swarm with enhanced local search ability
MDNLP	mixed discrete nonlinear programming
MMAS	max-min ant system
PCE	polynomial chaos expansion
PSO	particle swarm optimization
PSOGSA-E	PSO-GSA with enhanced local search ability
RBAS	rank-based ant system
RO	robust optimization

RQP	recursive quadratic programming
SA	simulated annealing
SACO	simple ant colony optimisation algorithm
SASS	self-adaptive step size search
SFS	stochastic fractal search
SSC	simplex stochastic collocation
S2M	simplex2 method
TLBO	teaching learning based optimization
TSP	travelling salesman problem

1 Introduction

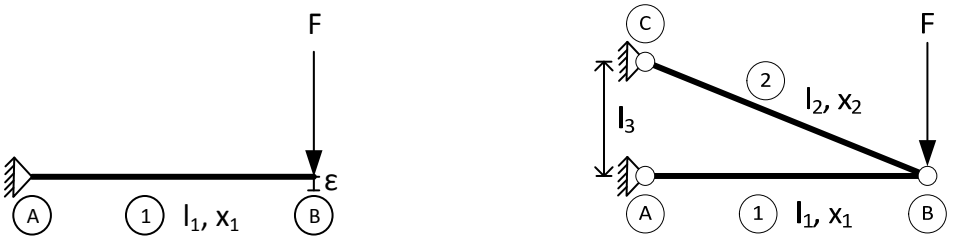
The Oxford English Dictionary defines engineering as the "… branch of science and technology concerned with the development and modification of engines […], machines, structures, or other complicated systems and processes using specialized knowledge or skills …" (OED, 2015). In other words, engineering uses sound scientific methods and technologies to solve real-world problems and develop new products and services. This chapter introduces the design aspect of engineering using a simple example. It then discusses the process of finding the optimal design and constraint optimisation before providing the principles of automated optimal design. It then states the research aims and objectives of this research project.

1.1 Engineering design

A typical task for an engineer is to design a system that is capable of fulfilling a desired function. This could be, for example, a bridge that spans over a defined distance that can resist a predefined maximum load. There is not one 'right' design that solves the problem. Instead many different designs are feasible. However, some designs are preferable to others, for example because they are cheaper to build or easier to manufacture. The challenge for the engineer is to choose the 'best' design from all of the feasible designs, where 'best' is defined in relation to the specific case by interested parties, i.e. stakeholders.

Figure 1 shows a simple yet typical example of an engineering design problem: point B has to withstand a force F at distance l_1 from point A without exceeding a maximum displacement ε. This can be achieved by putting a beam, also called a *member*, between points A and B (Figure 1a). Such a design, however, could be prone to plastic deformation under load if the cross-sectional area of the beam is not sufficiently large enough. On the other hand, if the cross-sectional area is large enough, the beam uses a lot of material and hence is very expensive and heavy. In order to save material and thus make the structure lighter, usually the load is distributed by adding additional supporting beams (Figure 1b). In structural engineering, it is

common to treat such structures as trusses, which consist of two-force members only (Costanzo, 2013). This means that forces apply only to the endpoints of a member and the joints between the members are treated as revolutes, i.e. they do not carry moments (torques). Therefore, at least one more member, member 2 in Figure 1b, is necessary to hold point B in position.



a) Functional requirement b) Practical implementation

Figure 1 – Required function and practical implementation.

The length l_1 of member 1 is defined by the problem and therefore fixed but length l_2 of member 2 can be chosen freely by the designer within limits. This length then determines the position of point C and therewith the length l_3 (Figure 1b). The material used has a specific Young's module E and a density ρ. The volume of a member, which is given by its length l multiplied by its cross-section x, and its density ρ, determines the weight of the member. Each design can be fully described by its design vector $\vec{d}$. Equation 1 shows the design vector for the example above:

$$\vec{d} = \begin{Bmatrix} l_1 \\ l_2 \\ x_1 \\ x_2 \\ E \\ \rho \end{Bmatrix} \tag{1}$$

Some of the design variables are predetermined by the problem (l_1, E, ρ in the example) and hence are called the *preassigned parameters* (Rao, 2009). They cannot be changed. However,

others can be chosen freely by the designer (l_2, x_1, x_2 in the example) and these are called *design* or *decision variables*.

A system described by its design vector can be judged in terms of goodness using a so-called *objective function* $f(\vec{d})$. In structural engineering, this objective function usually relates to the costs of a particular design, which is reflected in the weight of the resulting structure (Shanley, 1949). For example, the weight of the truss in Figure 1 can be calculated as follows:

$$f(\vec{d}) = l_1 x_1 \rho + l_2 x_2 \rho \tag{2}$$

If better designs are associated with smaller function values, the objective function is usually referred to as the *cost function*. If, on the contrary, better designs are associated with larger objective function values, the objective function is usually referred to as the *fitness function*.

1.2 Optimal design

As mentioned before, the challenge for the engineer is to find the optimal design. The objective function can be used to compare candidate designs with the aim of selecting the best one, i.e. the lightest one in the example. The optimal design is the design $\vec{d}'$ that minimises the objective function $f(\vec{d})$:

$$\vec{d}' = \arg\min_{\vec{d}' \in C} f(\vec{d}) \tag{3}$$

In Equation 3, C denotes the space of all feasible designs. If the objective function is differentiable twice and the design space C is unconstrained, calculus based methods (Christensen and Klarbring, 2008) can be applied to calculate the optimal solution. However, often, this is not the case, for example because of the complexity of the system and hence the complexity of the objective function. In such a case, a common approach to design optimisation is stepwise refinement: based on an initial feasible design, a human expert, i.e. the engineer, makes small adjustments to the design and evaluates its fitness. If the fitness increases the

adjustments are kept; otherwise the adjustments are discarded. This is then carried out iteratively until a design is arrived at that satisfies the requirements (Figure 2).

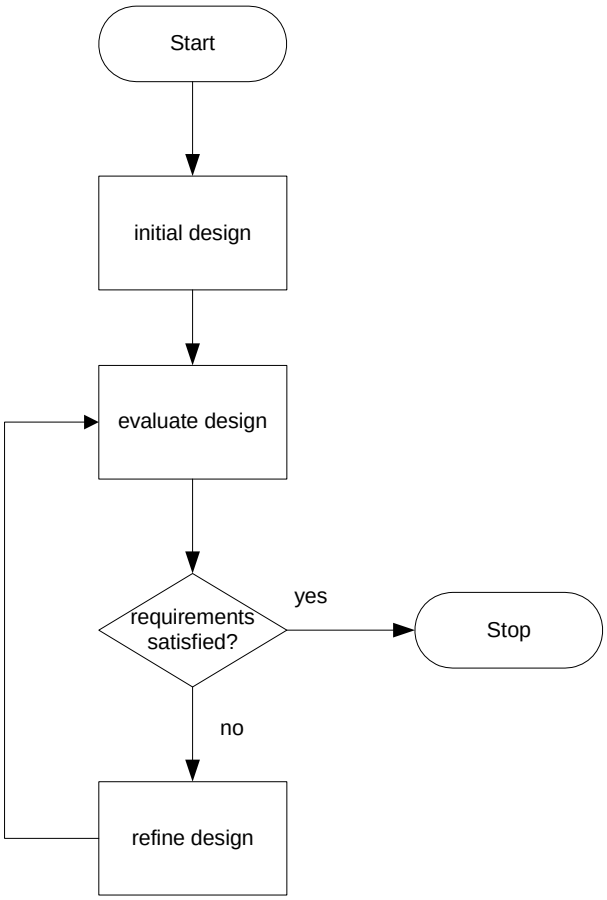


Figure 2 – The design process.

1.3 Constraint optimisation

However, not every design is actually feasible; the objective function in Equation 2 would return the lowest objective value if both x_1 and x_2 were equal to or smaller than zero. Obviously, such a design is impossible. Therefore, the cross-sections have to be chosen to be as small as possible. But if the cross-sections are too small, the members cannot withstand external load and the structure will fail. Hence, the design space, i.e. the collection of all possible designs, is limited by constraints. Thus, the optimisation problem can be re-formulated as follows (Rao, 2009):

$$\text{Find } \vec{d} = \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{Bmatrix} \text{ that minimises or maximises } f(\vec{d}) \tag{4}$$

subject to the following constraints:

$$g_i(\vec{d}) \leq 0, \quad i = 1,2,\ldots,m$$
$$h_i(\vec{d}) = 0, \quad i = 1,2,\ldots,p$$

In Equation 4, $g_i(\vec{d})$ denotes *inequality* constraints and $h_i(\vec{d})$ denotes *equality* constraints. The number of decision variables n of a design and the number of inequality or equality constraints (m respectively p) are usually not related to each other for a given problem. If one or more inequality constraints or equality constraints are violated the design is not feasible. In order to accommodate constraints in the optimisation process, equalities are usually handled by converting them into inequality constraints (Deb, 2000):

$$h_{m+i} \equiv \delta - \left| h_i(\vec{d}) \right| \geq 0 \tag{5}$$

In equation 5, δ is a small positive value. This conversion increases the total number of inequalities to $q = m + p$, and allows for handling inequalities and equalities using the same process. For constraint optimisation problems, a penalty term $p(\vec{d})$ is added to the objective function $f(\vec{d})$:

$$F(\vec{d}) = f(\vec{d}) + p(\vec{d}) \tag{6}$$

Instead of using the objective function value, the value returned by $F(\vec{d})$ is used in the optimisation process. In equation 6, the penalty term is computed as follows:

$$p(\vec{d}) = \sum_{i=1}^{m} R_i(g_i(\vec{d})) \tag{7}$$

Here, $R_i(g_i(\vec{d}))$ is a penalty function for the *i*-th constraint, i.e. a function that returns a non-zero value if the *i*-th constraint is violated or otherwise zero. This has the effect that a design that violates a constraint is penalised by its fitness decreasing in the case of maximising or its costs increasing in the case of minimising.

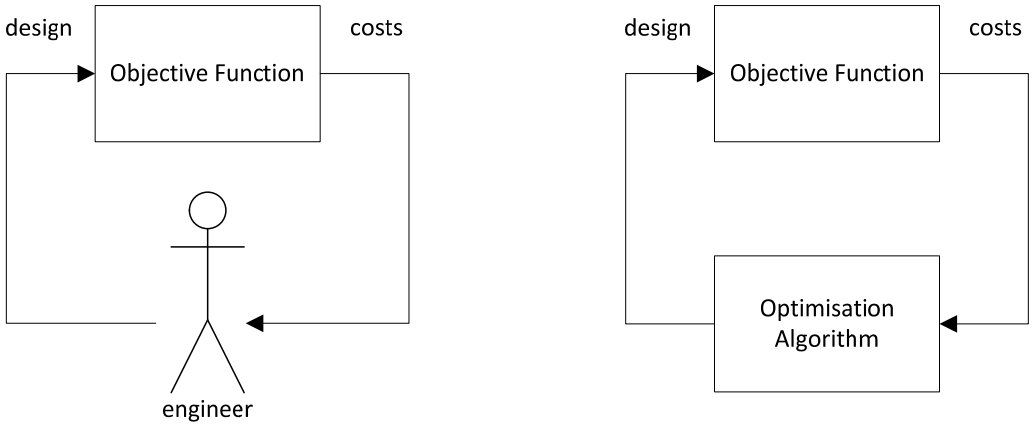
Choosing a suitable penalty function for a particular optimisation problem is not a straightforward task. In the simplest case, the penalty function adds a constant number to the objective function for each constraint violated. However, many researchers have developed more sophisticated methods for calculating penalty values. For example, Deb (2000) proposed a constraint handling method that does not require any fixed penalty parameter. Runarsson and Yao (2000) developed stochastic ranking for constraint handling. Li and Zhang (2013) introduced the minimum penalty coefficient method. A comprehensive survey of the most popular constraint-handling techniques can be found in Coello (2002).

By using a penalty function, a constraint optimisation problem is transformed into an unconstraint problem, and hence can be solved automatically as explained in the next section.

1.4 Automated design

The drawback of the design process outlined in Section 1.2 is that it relies on human expertise and experience. For example, if the engineer is inexperienced, they might limit themselves subconsciously to a certain area of the design space due to their personal biases. Therefore, an automated approach to the optimisation problem that removes this bias would be of benefit.

Figure 3a shows the manual approach to design optimisations whereas Figure 3b presents a computerised automated optimisation approach; a computational optimisation algorithm is used instead of a human engineer in a closed *optimisation loop* to experiment iteratively on the system in order to find the best solution (Nolle, 2006).



a) Manual design process b) Automated design process

Figure 3 – Optimisation loop for optimal design.

Starting with an initial design x_0, the fitness of that design is evaluated using an objective function. Based on the fitness, an optimisation algorithm makes adjustments to the decision variables, resulting in a new design vector x_1. The new design is then evaluated and the new fitness respectively cost information is used by the algorithm to further adjust the design vector. Depending on the algorithm used and the number of iterations, the objective value will improve until it converges towards an optimum value.

Automated design optimisation appeared in the early 1970s when computers became more readily available to researchers and engineers. For examples see Templeman (1970) or Pappas and Amba-Rao (1971). Since then, researchers have developed a large variety of optimisation algorithms, such as simulated annealing (Kirkpatrick *et al.,* 1984), genetic algorithms (Holland, 1975) and particle swarm optimisation (Kennedy and Eberhart, 1995), to name a few, and applied them to engineering design problems. Recent examples include the design of rotor-bearing systems (Lopez *et al.,* 2014) and the design of sub-structures for offshore wind turbines (Yang *et al.*, 2015). All of these applications have shown that automated design optimisation is capable of solving complex design optimisation problems. However, these

techniques have not yet been widely adopted by the engineering industry (Nolle, 2006). The reasons for this will be outlined in Chapter 2.

1.5 Aims of research

One aim of this research was to identify the main obstacles that practitioners in the field of engineering design are faced with when trying to use automated design optimisation for practical applications. The second aim of the research was to find practical solutions to the problems identified.

1.6 Methodology

Knowledge of automated engineering design and optimisation was obtained by reviewing the literature. Two main hurdles that impede engineers from using automated design optimisation were identified: the problem of parameter tuning and the problem of finding robust solutions. This led to the development of a novel method for robust optimisation that does not require parameter tuning. The new method was then tested using a well-known engineering problem with a large number of constraints, which is often used as a benchmark by researchers for comparing optimisation algorithms. The solutions obtained were analysed and compared with solutions found by other state of the art search algorithms using a Monte Carlo simulation of a production process.

1.7 Dissertation outline

This chapter introduced the design aspect of engineering using a simple example. It then discussed the process of finding the optimal design and constraint optimisation before providing the principles of automated optimal design. It then stated the research aims and objectives of this research project. Chapter 2 starts with a review of the current literature on optimisation algorithms for engineering design problems and identifies emerging trends within this body of work. It then identifies problems that practitioners encounter when trying to apply computational optimisation to real-world problems before summarising the main findings of this section.

Chapter 3 introduces an interesting benchmark problem from the field of mechanical engineering, the pressure vessel problem. Here, the aim is to minimise the total cost of the materials used as well as the production costs. The required accuracy for the pressure vessel problem is discussed and analysed under the condition that the manufacturing process cannot achieve the required accuracy and hence result in a different fitness value as a consequence when the parameter values are slightly different. Chapter 4 describes problems caused by inaccuracies in engineering processes, i.e. manufacturing processes, for real-world applications of computational optimisation techniques. The case study presented in Chapter 3 is used to demonstrate this problem. Based on an analysis of the problem, a novel meta-heuristic is proposed, based on Baldwinian learning, to temporarily shape the fitness landscapes so that solutions are still fit for purpose even if the implemented solution deviates slightly from the theoretical one due to the nature of the manufacturing processes. The proposed meta-heuristic is applied to the pressure vessel design problem. BMH is combined with SASS and the resulting algorithm is employed to optimise the pressure vessel design in Chapter 5. The solutions found by BMH/SASS, as well as the best solutions for PSO, HPB and SASS reported in the literature, are virtually manufactured using a Monte Carlo simulation. Chapter 6 revisits the research question presented in Chapter 1 and discusses the findings obtained by applying the novel Baldwinian-based meta-heuristic. Following a discussion, it draws the dissertation to a close by suggesting future work that could be undertaken.

2 Practical limitations of automated engineering design

This chapter starts with a review of the current literature on optimisation algorithms and identifies emerging trends within this body of work. It then identifies problems that practitioners encounter when trying to apply computational optimisation to real-world problems before summarising the main findings of this section.

2.1 Current research

Many computational optimisation algorithms, or direct search methods, have been proposed in the past. Early algorithms included, for example hill climbing (Bach, 1969), simulated annealing (Metropolis *et al.*, 1953; Kirkpatrick *et al.*, 1984), the simplex method (Nelder and Mead, 1965), evolutionary strategies (Rechenberg, 1973) and genetic algorithms (Holland, 1975). This was followed by particle swarm optimisation (Kennedy and Eberhart, 1995), ant colony optimisation (Dorigo and Gambardella, 1997), differential evolution (Storn and Price, 1997) and self-adaptive stepsize search (Nolle and Bland, 2012), amongst others. Later it became popular to combine these methods with local search schemes to improve the search capabilities of the algorithms, especially with genetic algorithms.

Many new schemes have since been proposed. For example, Nema *et al.* (2008) introduced a new hybrid algorithm, called hybrid particle swarm branch-and-bound (HPB), which uses particle swarm optimisation (PSO) as the global optimisation technique and combines it with branch-and-bound (BB) to solve mixed discrete nonlinear programming problems (MDNLP) (Kitayama *et al.*, 2006). PSO showed good global but slow local search abilities and was therefore combined with the BB technique, which features fast local search ability. The combination of these two search techniques exhibits improved optimisation accuracy and reduced demand for computational resources (Xu *et al.*, 2007). The HPBs hybridisation phase uses mainly a selective temporary varying from PSO to BB when the current optimum could be improved (Dimopoulos, 2007). HPB can handle random and discrete constraints without the need for parameterising a penalty function (Jeet and Kutanoglu, 2007).

This architecture makes the HPB simple, generic and easy to implement. Nema *et al.* were also able to show that the combination of both search techniques results in better solutions and a lower number of function evaluations than using BB or PSO separately.

Most recently, for example, Li (2015) proposed an improved, chemical reaction optimisation (CRO) method, called OCRO, to solve global numerical optimisation problems. The CRO method mimics the interactions between the molecules in chemical reactions to find a low energy stable state. The proposed algorithm is a hybrid of two local search operators in CRO and a quantisation orthogonal crossover (QOX) search operator. An advantage of this approach is that no complicated operators are necessary, which makes it easy to implement in various real-world problems. The experimental results have shown that the algorithm is effective and fast in solving common optimisation problems, but is less efficient in solving low-dimensional functions.

Wang (2015) has tried a different approach and introduced a hybrid algorithm combining genetic particle swarm optimisation with advanced local search ability (L-GPS) and a quasi-newton genetic algorithms (GA) with limited memory. Because of its wide range search ability, GA is used to search global minima/maxima. The global extremes are then further investigated in terms of local extremes with an algorithm that shows good local search abilities (L-GPS). The hybrid algorithm was then evaluated using numerical experiments that showed that the hybrid algorithm is much faster and more efficient than most stochastic methods.

Another recent approach has been published by Zhouab (2015), who proposed in his paper a new hybrid glow-worm swarm optimisation algorithm, named the Hybrid Glow Worm Swarm Optimization Algorithm (HGSO), to solve constrained optimisation problems. A fish school algorithm (FSA)) in which predatory behaviour is exhibited is integrated with a glow-worm swarm optimisation to obtain the improved glow-worm swarm optimisation (IGSO). Here, the IGSO algorithm is integrated with the differential evolution method (DE) to establish the HGSO. The local search strategy used to escape from the local minimum is based on simulated

annealing (SA) and is applied to the best solution found by the IGSO. The experimental results have shown that the HGSO is very effective in terms of efficiency, reliability and precision.

In their recent paper, Mahia (2015) suggested a new hybrid method based on particle swarm optimisation (PSO), artificial ant colony optimisation (ACO) and 3-opt algorithm to solve the travelling salesman problem (TSP). In this approach the PSO is used to determine parameters that affect the performance of the ACO. The 3-Opt algorithm is subsequently used to remove the local solution. The performance of this method was investigated by altering the average route length or the effective number of ants in the ACO. The experimental results show that the fewer the number of ants, the better the performance of the proposed method.

Jayaprakasam (2015) suggested a new algorithm, PSOGSA-E, which combines the local search ability of the gravitational search algorithm (GSA) with the social thinking of PSO. The new hybrid algorithm improves the beam-pattern of collaborative beam-forming to achieve lower side lobes by optimising the weight vector of the array elements. By using exploration and exploitation, a global optimum is determined by the hybrid algorithm and the problem of premature convergence is alleviated compared to the legacy optimisation methods.

Jamshidi (2015) introduced an optimisation method based on a set of performance indices to determine the effectiveness and robustness of a supply chain. These indices are categorised into qualitative and quantitative indices to optimise supply chain parameters like costs or waiting time. An artificial immune system (AIS) was utilised as a meta-heuristic method and it was then combined with the Taguchi method to optimise the problem. It was shown that this novel hybrid method leads to more accurate minimum average costs and less standard deviation. In computational tests, it was also shown that the procedure was effective and efficient and could be used for a variety of optimisation problems.

Another novel hybrid algorithm of teaching–learning-based optimisation (TLBO) and a DE was introduced by Pholdeea (2015). The algorithm was used to optimise a strip coiling process and it has been shown that it can also handle large-scale design problems for industrial

applications. The optimal processing parameters are determined by minimisation of the proposed objective function with a limited number of function evaluations. The hybrid algorithm was tested and compared with ten already existing EA strategies and it was able to outperform the other EAs in terms of efficiency and computational time.

In their recent paper, Martínez-Soto (2015) described the application of bio-inspired methods to designing logic controllers using genetic algorithms (GA), particle swarm methods (PSO), and hybrid PSO-GA methods. Some of these optimisation methods are based on populations of solutions and produce a set of solutions for each iteration step. All of these different solutions are selected, combined and replaced to find the optimised solution. This process requires much more computing time than other meta-heuristic methods and a greater numbers of iterations. Martínez-Soto's aim was to develop more aggressive methods by combining population based algorithms with single solution algorithms. The experimental procedure was realised for each method individually (GA, PSO) and for both methods combined (PSO-GA). Afterwards the results obtained using each method individually were compared with the results obtained using both methods combined.

Arnaud (2014) proposed a combinatorial optimisation algorithm matched to a continuous optimisation problem that is related to uncertain aero elastic optimisation. The algorithm optimises the sum of the expected value by stochastic annealing of the objective function. Additionally a penalised term regards the confidence interval bandwidth of the estimator. An advantage of this algorithm is that it chooses the number of samples used for estimator construction. By not using a fixed number of samples the computational efficiency is improved compared to a classical annealing algorithm. The test problem shows that the algorithm can be used efficiently for low dimension optimisation problems under uncertainty with no objective function gradient. Compared to gradient based approaches, the computation time of the proposed algorithm is more important so that it can be inefficient for high dimension industrial application.

Another interesting strategy was introduced by Congedo (2013), who proposed a multi-scale strategy called the simplex 2 method (S2M), to minimise the cost of robust optimisation procedures. It is based on simplex tessellation of uncertainty as well as of the design space. The algorithm coupled the simplex stochastic collocation (SSC) method employed for uncertainty quantification with the Nelder-Mead optimisation algorithm (Nelder and Mead, 1965). The robustness of the S2M method is caused by using a coupled stopping criterion and a high-degree polynomial interpolation with P-refinement on the design space. This method has been applied to various algebraic benchmark test cases like the Rosenbrock problem, for which it was able to reduce the global number of deterministic evaluations by about 50 %. Another test case was the short column problem, which was treated using the S2M method and a classical technique called polynomial chaos expansion (PCE) with genetic algorithms (GA). The comparison of both methods has shown that S2M is able to gain one order of magnitude in terms of convergence order while maintaining the same level of error.

Based on the review of the existing state-of-the-art optimisation algorithms, trends were identified. These trends are outlined below.

2.2 Trends identified

All of the methods described in the section above are hybrid algorithms. That means that they combine two or more optimisation algorithms to improve either effectiveness or efficiency. In the former case, they aim at increasing the quality of the solutions, i.e. finding solutions with a higher fitness value in the case of maximisation or lower cost values in the case of minimisation. In the latter case they aim at minimising the computational costs, which are caused by objective function evaluation.

Hybrid algorithms can be classified into three different types: nested optimisation algorithms, sequential optimisation algorithms and meta-optimisation algorithms. These types are discussed in more detail below.

Figure 4 shows the basic principle of direct search algorithms for optimisation. Starting with an initial solution (or a set of initial solutions for population based algorithms), the current solution is evaluated. If a stopping condition is not met, the current solution is adjusted based on the strategy of the algorithm. The adjusted solution is then evaluated again before the stopping criterion is tested. This is done iteratively until the stopping condition holds.

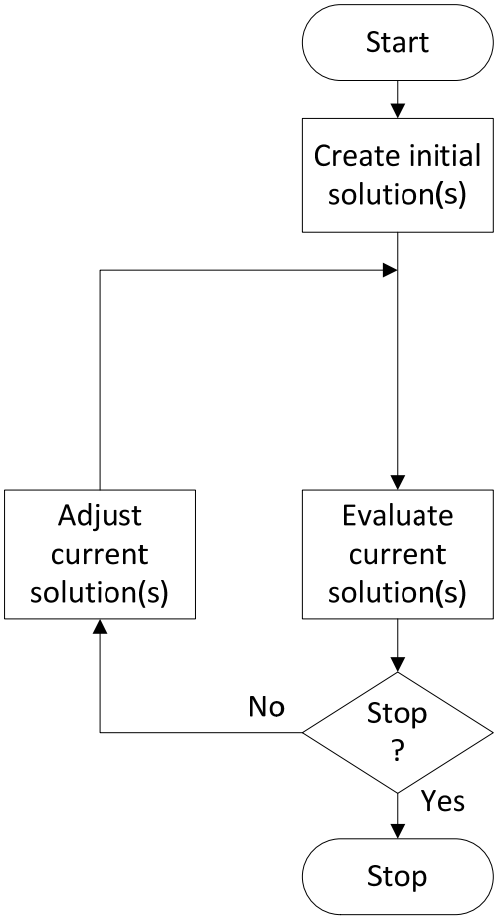


Figure 4 – Basic direct search algorithm for optimisation.

In the case of hybrid algorithms, at least two search algorithms are combined. The most common form here is the nested algorithm. Examples can be found in Martinez-Soto *et al.* (2015), Arnaud *et al.* (2014), Congedo *et al.* (2015), Li *et al.* (2015) and Nema *et al.* (2008). In the nested approach, a local search procedure, for example hill climbing, is inserted into the optimisation loop of a global search method like GA (Figure 5). In every iteration of the global search method, a complete local search run is carried out that aims at improving the current solution generated by the global search method. The advantage is that the outer search algorithm can be used to

explore the whole search space whilst the inner optimisation method is exploiting the region of the search space that contains the current solution.

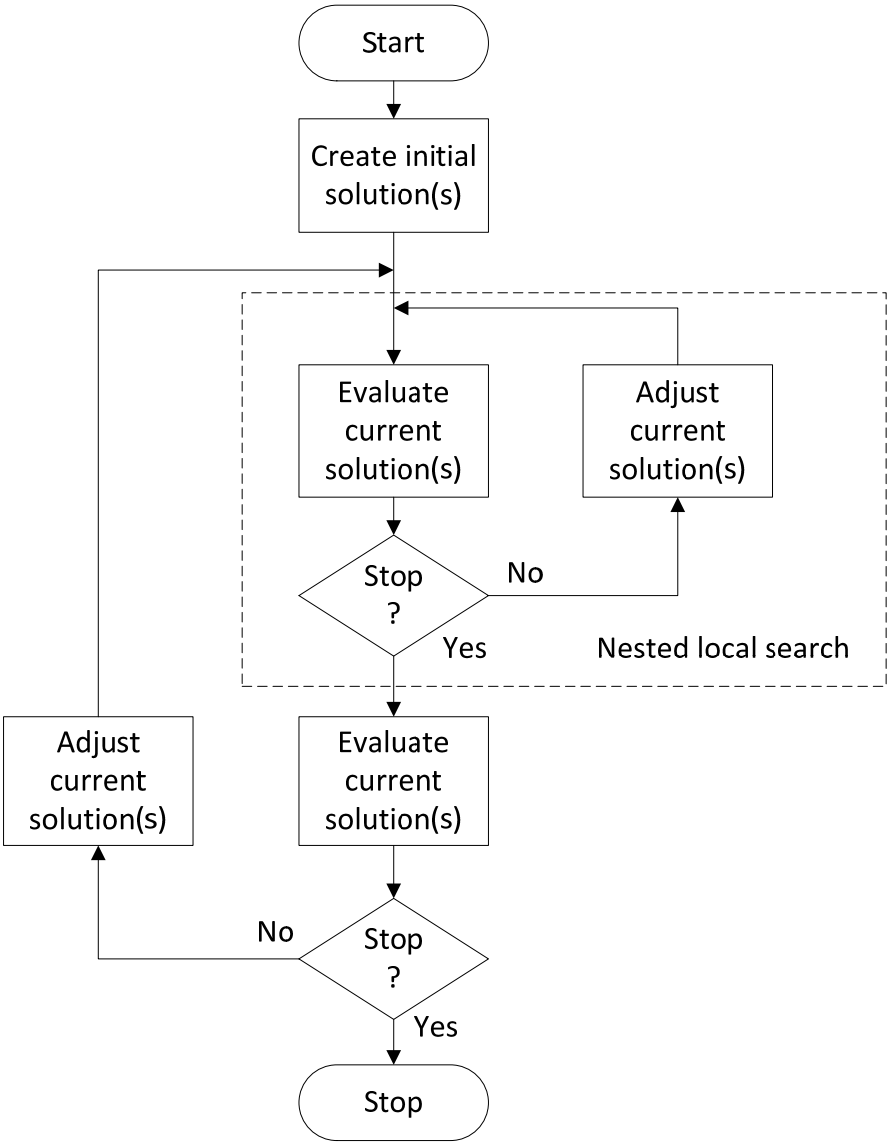


Figure 5 – Nested search algorithm.

There are two ways of combining the knowledge acquired through local search and global search, *Lamarkian* and *Baldwinean* leaning (El Mihoub *et al.*, 2006).

In genetic algorithms, the process of Lamarckian learning is connected with the inheritance of acquired characteristics reached through learning. This means that the genetic structure and the fitness of an individual are altered to match the solution. The genetic structure of an individual is modified by using the local search method, which acts as a refinement genetic operator. The local search method modifies the genetic structure of an individual and feeds it

back into the genetic population. The occurrence of this genetic feedback process has not yet been observed in real-world biological systems, but it can be simulated in a computational environment. By implementing a Lamarckian approach in genetic algorithms, the search process can be improved and accelerated but this may lead to premature convergence in some cases (Whitley *et al.*, 1994). To adopt a Lamarckian approach, however, it is necessary to invert the mapping from phenotype to genotype to ensure the correct function of the method. In many simple applications, the reverse mapping may be computable but it is intractable for most real-world problems (Turney, 1996). Lamarckian approaches have been widely used in hybrid genetic algorithms used for chromosome repair applications or the travelling salesman problem (TSP) (Julstrom, 1999).

In the Baldwin learning process, on the other hand, the genotype remains unchanged while the fitness of an individual is enhanced by applying a local search mechanism. This leads to an improvement in the solution's chances to pass on its structure to the next generations. In evolution, learning does not change the genetic structure of an individual but can increase its ability to survive. With the use of local knowledge, the local search method produces a new fitness score, which is then used by the global genetic algorithm to decide whether or not the individual's ability is improved. The Baldwin effect describes the interactions between learning and evolution by balancing benefit and cost of learning. The Baldwin effect can be described as follows (Turney *et al.*, 1996). Firstly, the process of learning enables individuals to change their phenotypes to enhance their fitness. Secondly, the individuals who improve their fitness through learning will preferentially breed into the next population. The cost accompanied with learning leads to the selection of favourable individuals who have certain properties. These properties are acquired by other individuals due to the learning skills coded into their genotypes. This process is called genetic assimilation and can indirectly accelerate the genetic acquisition of learned properties. In order for genetic assimilation to occur, a weighty correlation between genotype and phenotype space is mandatory (Mayley, 1996). Another advantage of the Baldwin effect is the so-called smoothing effect. This means that the fitness landscape of a difficult optimisation problem can be simplified into flat landscapes round about

the basin of attraction (Hinton and Nowlan, 1987). Despite all that, Baldwinian search can also cause an unwanted side effect called the hindering effect (Mayley, 1996). This effect obscures genetic differences by mapping different genotypes to the same or similar phenotypes, thereby hindering the evolution process.

One disadvantage of using the nested approach is that the number of objective function evaluations n increases to $n = i \cdot j$, where i is the number of iterations of the global method and j is the number of iterations of the local method. Another disadvantage is that two sets of control parameters have to be tuned manually.

A further possible configuration is the sequential application of two or more optimisation algorithms (Figure 6). Here, the first optimisation algorithm is used to explore the whole search space in order to find the most promising region. The solution found is then used as the initial solution for the second algorithm, which then exploits the region.

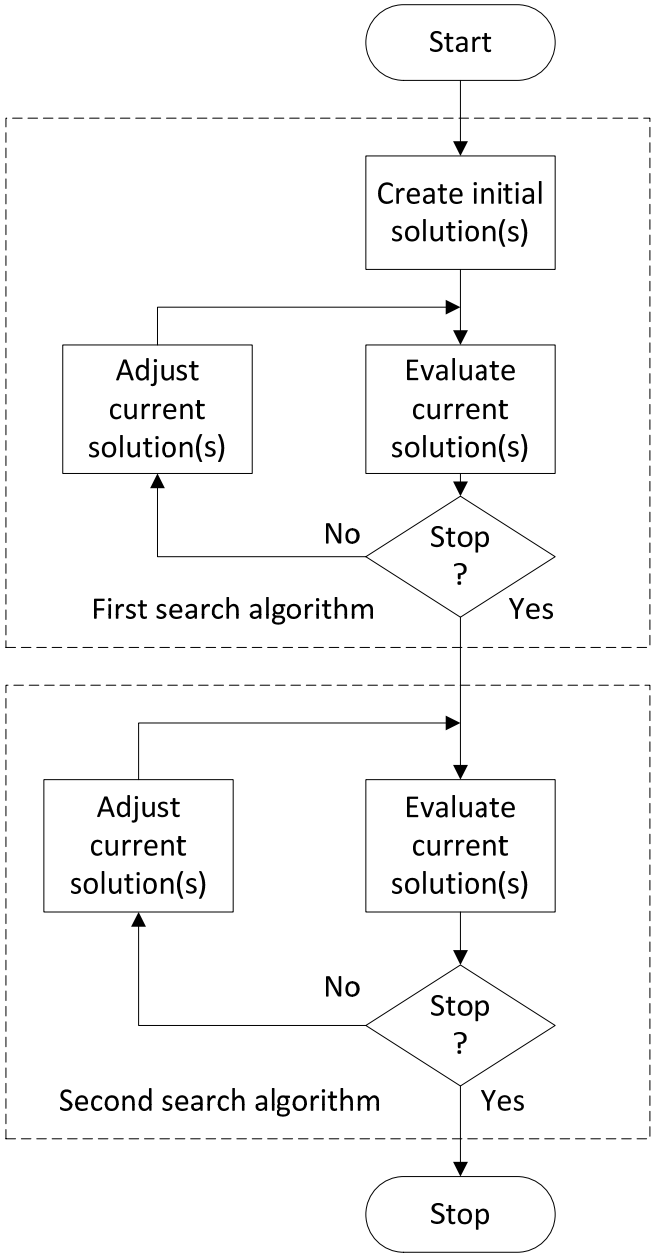


Figure 6 – Sequential configuration of search algorithm.

The balance between exploration and exploitation is not as good as in the nested approach, but nevertheless it allows for combining two specialised algorithms, a global search algorithm and a local search strategy. The number of objective function evaluations n increases to $n = i + j$, where i is the number of iterations of the global method and j is the number of iterations of the local method. Here, too, there is the disadvantage that two sets of control variables have to be tuned manually. Examples of sequential search algorithms can be found in Zhou *et al.* (2015) and Wang *et al.* (2015).

A third configuration is the meta-optimisation approach. In this, one optimisation algorithm is used to tune the control parameters of the second algorithm, which in turn carries out the actual optimisation (Figure 7).

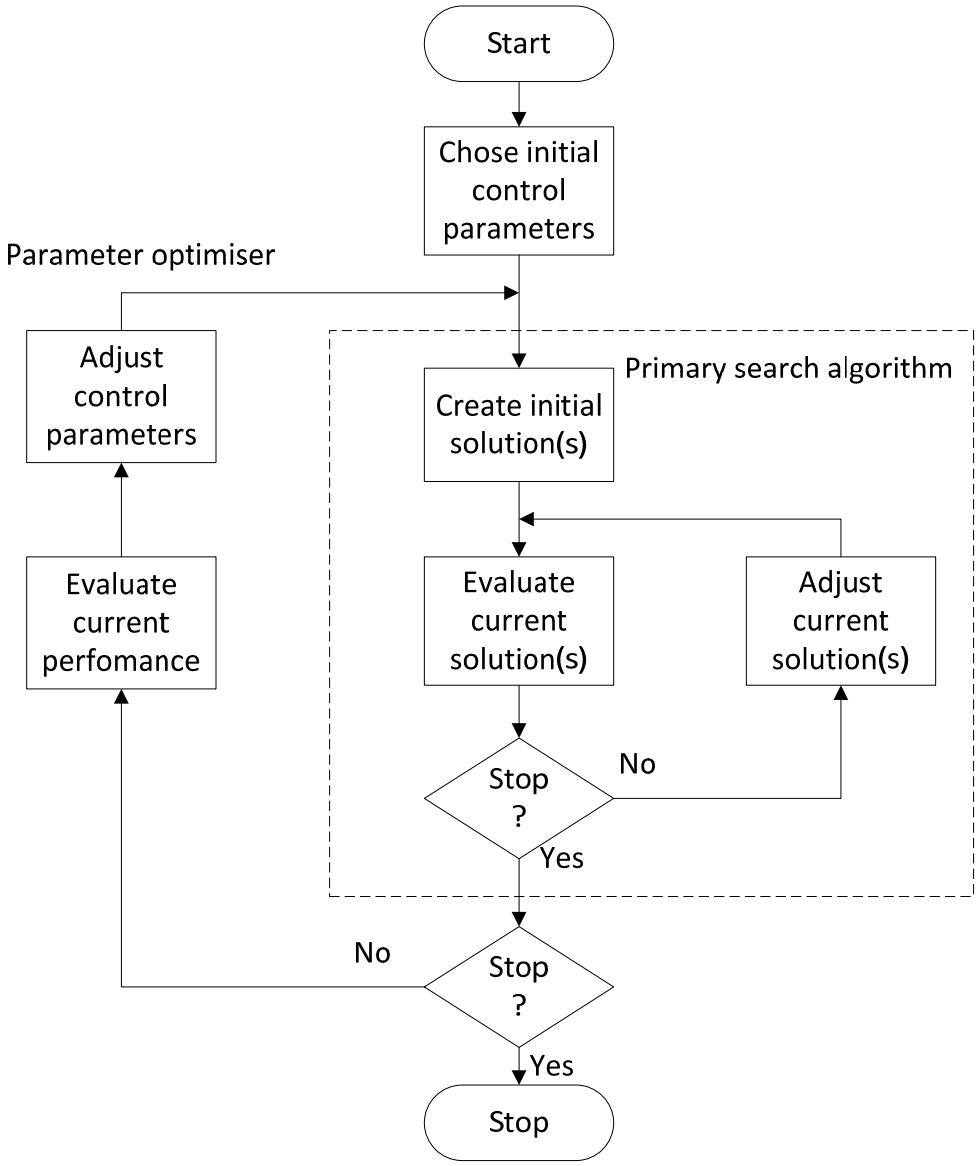


Figure 7 – Meta-optimisation approach for parameter tuning.

This removes the need for manually tuning the control parameters of the primary search algorithm, but it increases the number of objective function evaluations n to $n = i \cdot j$, where i is the number of iterations of the parameter optimiser and j is the number of iterations of the primary search method. This approach only makes sense if the parameter optimiser itself has fewer control parameters than the primary search method. Recent examples of meta-optimisers

can be found in Jayaprakasam *et al.* (2015), Mahia *et al.* (2015), Jamshidi *et al.* (2015) and Pholdee *et al.* (2015).

Although computational optimisation techniques have been proven to be very effective, they seem to be neglected by practitioners in the field of engineering (Nolle, 2006). The next section discusses the main reasons for this.

2.3 Practical problems

The tuning of optimisation algorithms, and hybrid algorithms in particular, presents problems to practitioners who want to use direct search techniques as black-box methods for practical engineering applications: All of these algorithms have a number of control parameters that need to be carefully adjusted to the optimisation problem at hand. This constitutes a meta-optimisation problem, i.e. the optimum set of control parameters has to be found, usually empirically, before the main optimisation process can start. This fine-tuning of the control parameters requires a lot of experience with optimisation algorithms and usually a large number of experiments.

Table 1 shows a selection of common population based search algorithms and lists their basic sets of parameters. For example, the basic GA has seven parameters that need to be tuned before the actual optimisation can be carried out. Due to combinatorial explosion and the continuous nature of some of the parameters, not every combination can be tried in an exhaustive search. Using unsuitable settings can easily lead to sub-optimal solutions.

Engineers are often not very experienced with search algorithms and hence often struggle to find suitable settings within the limited time available for tuning. Ideally, engineers would like to use an algorithm that has no control parameter at all. This has led to the development of self-adaptive stepsize search (SASS), an algorithm that has only one control parameter (Nolle, 2006; Nolle and Bland, 2012; Azad and Hasancebi, 2014a; Azad and Hasancebi, 2014b). It has been demonstrated in practical applications that SASS can be used

without the need for tuning control parameters (Nolle, 2007; Sayol *et al.*, 2008; Dias Junior and da Silva Junior, 2013).

Table 1 – List of control parameters for a selection of population-based search methods.

Algorithm	Parameter	Type
genetic algorithms	population size	discrete
	mutation probability	continuous
	crossover probability	continuous
	chromosome length	discrete
	crossover type	selection
	selection type	selection
	coding scheme	selection
ant colony optimisation	population size	discrete
	α	continuous
	β	continuous
	pheromone evaporation coefficient	continuous
	$\Delta\tau$	continuous
particle swarm optimisation	population size	discrete
	inertia weight	continuous
	cognitive learning parameter	continuous
	social learning parameter	continuous
self-adaptive stepsize search	population size	discrete

Although SASS does not require parameter tuning, apart from selecting the number of particles in the population, the algorithm might take a longer time to reach the global optimum and hence

is not as efficient as the latest hybrid algorithms outlined above. However, for practical applications, this is compensated for by the fact that no time needs to be spent experimenting with parameter settings.

Another problem for practical engineering applications is that the typical optimisation approach typically considers only optimal solutions with a high sensitivity to small changes rather than robust optimisation and solutions (Beyer and Sendhoff, 2007). The procedure for finding robust solutions is referred to as robust design optimisation. The aim of robust design optimisation is to ensure that the performance, as well as the solution of the objective function, will remain relatively stable even under uncertain conditions (Lee *et al.*, 1996; Suri *et al.*, 2001). These uncertainties are caused by fluctuations of real-world system parameters like design parameters, material properties, environmental influences, changed parts in a multi-part system or applied loadings (Phadke *et al.*, 1989; Sundaresan *et al.*, 1995). All of these uncertainties cause noise, dropouts or errors in the objective function's input dataset. To assure the correct performance of the algorithms and models when handling error-prone data it is necessary to lower the sensitivity of the objective function towards the errors contained in the dataset. Many papers and publications have been released that deal with the implementation of error insensitive optimisation methods (Du *et al.*, 2000).

Lee *et al.* (2001), for example, suggested an approach in which the robustness of the objective and the constraint functions is defined. His algorithm is a recursive quadratic programming (RQP) method developed in an optimisation system called IDESIGN3. Due to the robust design of the objective function the system's performance becomes insensitive to variations in the design variables (Lee *et al.*, 2001). To solve real-world optimisation problems, a multi-objective function is introduced and defined as the weight in the structural optimisation. Robustness is achieved by using a constraint function and making use of penalty terms with penalty coefficients. These coefficients are proportional to the gradients of the constraints and the tolerance of the design variables. By considering the tolerance of the design variables by the

overhauled constraint functions, the original constraints are satisfied. For larger penalty factors the operability of the constraint rises but the performance of the objective function worsens.

In their paper, Mulvey and Vanderbei (1994) suggested an alternative approach, named robust optimisation (RO), which integrates goal programming for simulations with a scenario-based description of problem data. RO outputs a series of solutions that are gradually less alive to realisations of the data from a scenario set. The advantages of RO are its good generally applicability and its benefit towards stochastic linear programming (Golub *et al.*, 1994).

2.4 Summary

This chapter started with a review of the current literature on optimisation algorithms. It emerged that the latest research tends to combine two or more optimisation algorithms to improve either effectiveness or efficiency. Three basic types of hybrid configurations were identified: nested algorithms, sequential algorithms and meta-optimisers. The later aim was to then to reduce the work load for practitioners by automatically tuning the control parameters of an optimisation algorithm. Two problems were then identified that practitioners encounter when trying to apply computational optimisation to real-world engineering problems: parameter tuning and robust optimisation. The next chapter presents a case study, which is used to investigate the problems of parameter tuning and robust optimisation in an engineering context.

3 Case study: pressure vessel problem

In engineering, designing a system that satisfies all of the user's requirements, without violating problem specific constraints, usually requires determining the system's design variable values in such a way that the resulting design is optimal in turns of cost function. Usually, the design space is too vast to evaluate every possible design and hence only a subset of the design space can be considered. An interesting standard benchmark problem from the field of mechanical engineering, which was introduced by Sandgren (1990), is the pressure vessel problem. It has been used as a standard problem by many researchers. Current examples include Liao *et al.* (2014), Kanagaraj, *et al.* (2015), Guo *et al.* (2015), and Salimi (2015). Although the problem only consists of four decision variables (see Chapter 1), it is not a trivial problem, as it involves the optimisation of both discrete and continuous design variables under a relatively large number of constraints. The problem is used as a test-bed here and is therefore explained in more detail in the next section.

3.1 Pressure vessel problem

The pressure vessel consists of a simple cylinder (shell) with each end capped by a hemi-spherical head, as illustrated in Figure 8. There are four design variables that can be chosen by the engineer: the thickness x_1 of the shell, the thickness x_2 of the heads, the inner radius x_3 and the length x_4 of the shell.

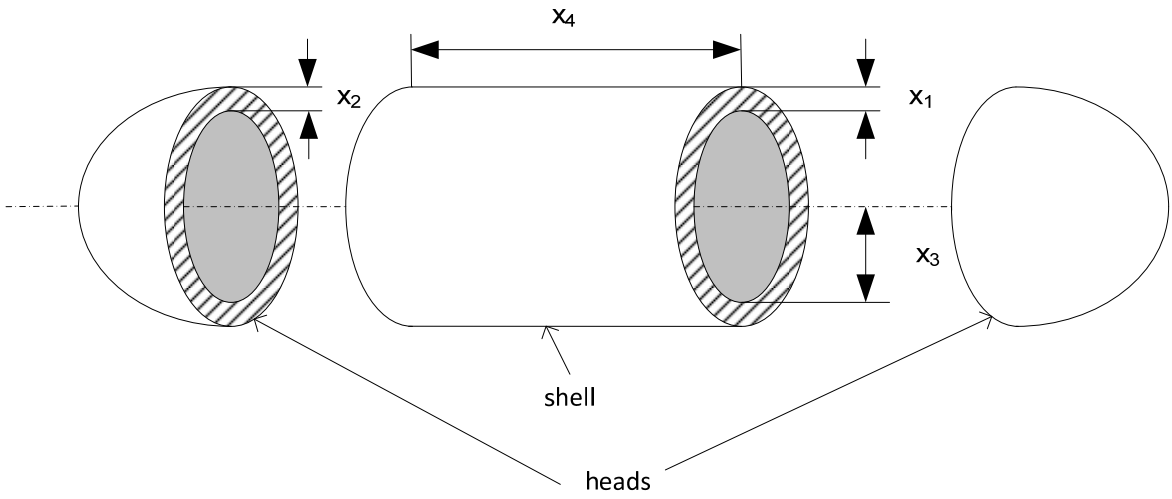


Figure 8 – Pressure vessel design problem.

The variables x_1 and x_2 are both discrete and vary in multiples of 0.0625 inches (please note that Sandgren used Imperial units), and variables x_3 and x_4 are both continuous (see Table 2).

Table 2 – Design parameter types and ranges for the pressure vessel problem.

Design variable	Definition	Variable type	Thickness increments
x_1	Thickness of cylinder	discrete	0.0625 inch
x_2	Thickness of sphere	discrete	0.0625 inch
x_3	Inner radius of cylinder	continuous	N/A
x_4	Length of cylinder	continuous	N/A

The design has to satisfy four constraint functions and an allowed range for each design variable. The design and constraint functions and the ranges of the design variables are given in the following equations and inequalities:

$$g_1(\mathbf{x}) = 0.0193x_3 - x_1 \leq 0$$

$$g_2(\mathbf{x}) = 0.00954\,x_3 - x_2 \leq 0$$

$$g_3(\mathbf{x}) = 1{,}296\,.000\; - \pi x_3^2 x_4 - 4/3\pi x_3^3 \leq 0$$

$$g_4(\mathbf{x}) = x_4 - 240 \leq 0$$

$$0.0625 \leq x_1 \leq 6.1875 \qquad (7)$$

$$0.0625 \leq x_2 \leq 6.1875$$

$$10 \leq x_3 \leq 200$$

$$10 \leq x_4 \leq 200$$

The aim is to minimise the total cost of the materials used and also the production costs, which consist of the costs of forming and welding the pressure vessel. Equation 8 provides the objective function $f(\vec{x})$, which was introduced by Sandgren (1990). It comprises the four design variables $\vec{x} = (x_1, x_2, x_3, x_4)^T$.

$$f(\vec{x}) = 0.6224\; x_1 x_3 x_4 + 1.7781\; x_2 x_3^2 + 3.1661\; x_1^2 x_4 + 19.84\; x_1^2 x_3 \qquad (8)$$

A number of computational optimisation methods have been used to find near-optimum designs. These methods are iterative in nature and use an optimisation loop (Figure 9). Based on an initial solution $\vec{x}^0$ the cost value $f(\vec{x}^0)$ is evaluated. If one or more constraints are violated, penalties are added to the costs, which are now $f'(\vec{x}^0)$. Based on the algorithm used, the design is altered and a new solution $\vec{x}^1$ is generated. This process is than carried out iteratively until a stopping criterion holds.

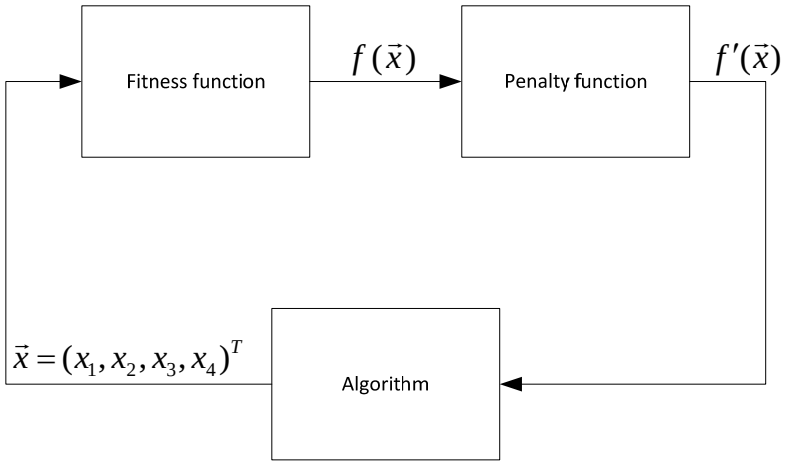


Figure 9 – Optimisation loop for pressure vessel.

Previous algorithms used are, amongst many others, BB (Sandgren, 1990), evolutionary programming (EP) (Cao and Wu, 1999), genetic algorithm (Coello and Montes, 2001), particle swarm optimisation (PSO) (He and Prempain, 2004), hybrid particle swarm branch-and-bound (HPB) (Nema *at al.*, 2008), self-adaptive stepsize search (SASS) (Nolle and Bland, 2012), stochastic fractal search (SFS) (Salimi, 2015) and biogeography-based particle swarm optimisation (BPSO) (Gao *et al.*, 2015). Interestingly, the latter two papers reported results that either showed no improvement over the existing methods (Salimi, 2015) or were even worse (Gao *et al.*, 2015).

Table 3 shows the best designs, as found by the three best methods, and the associated fitness values (Nolle and Bland, 2012). As can be seen, the fitness (costs) could be reduced dramatically compared to Sandgren's original value of 7982.5.

Table 3 – Comparison of solutions.

	PSO	**HPB**	**SASS**
Fitness	6,059.7143	6,059.6545	6,059.7143
x_1	0.8125	0.8125	0.8125
x_2	0.4375	0.4375	0.4375
x_3	42.09845	42.09893	42.09845
x_4	176.6366	176.6305	176.5366

This example clearly shows that computational optimisation methods are capable of finding very good solutions for engineering applications in terms of fitness function values. However, when implementing optimal designs physically, engineers are often confronted with a number of problems, which are outlined below.

3.2 Problems with optimal solutions

As can be seen from the example above, computational optimisation methods are capable of finding very good solutions for engineering applications in terms of fitness function values. However, the solutions presented in the literature often use design parameter values that have decimal places. This means, in reality, that those solutions cannot actually be manufactured because of the tolerances of both the materials involved and the manufacturing processes.

For the pressure vessel problem, for example, the solutions presented in the literature show up to five places after the decimal point. Since the manufacturing process cannot achieve the required accuracy, the parameter values are slightly different and hence result in a different fitness value. For example, using a modern CNC machine, tolerances of +/- 0.005 mm are common.

Also, controlling the dimensions of the heads is especially difficult (Pullarcot, 2002). Other unavoidable causes of error are the thickness tolerances for plate rolled on a plate mill

(Standards Australia, 1995). Table 4 shows the combined upper and lower tolerances for the pressure vessel design problem.

Table 4 – Upper and lower tolerances for the pressure vessel.

Tolerance [inches]	x_1	x_2	x_3	x_4
Lower limit	-0.0118	-0.0118	-0.1969	-0.2756
Upper limit	0.0472	0.0472	0.1969	0.2756

The upper and lower tolerance values were added to the design solutions presented in Table 3. Table 5 shows how the fitness values changed when the upper and lower tolerances were taken into account.

Table 5 – Fitness after applying upper and lower tolerances to the original design.

Method	**Original fitness**	**Fitness for lower tolerance values**	**Fitness for upper tolerance values**
PSO	6,059.7143	N/A (g_1 and g_3 violated)	6579.68
HPB	6,059.6545	N/A (g_1 and g_3 violated)	6579.60
SASS	6,059.7143	N/A (g_1 and g_3 violated)	6579.678

As can be seen from Table 5, the fitness decreases dramatically for all of the designs using the upper tolerance values, with an error of 8.6%. Even worse, when using the lower tolerances, none of the designs are feasible because they all violate constraints g1 and g3 and hence would not be fit for purpose!

There are two aspects of robust optimisation. The first is related to the narrowness of the global optimum for unconstraint optimisation. Figure 10 shows an example of a fitness landscape for a one-dimensional optimisation problem. In the example, there are two maxima, one at $x = 5$ and the other at $x = 10$. The global optimum is located at $x = 5$. However, the peak is too narrow, and slight deviations in implementing the solution will cause a significant

drop in the fitness value. The local optimum at $x = 10$ does not offer the same fitness, but slight deviations in x do not cause a dramatic drop in fitness when implementing the solution. If the fitness were acceptable, this would be the preferred solution for an engineering application. For a practical application, it would be more desirable to find a robust solution than the global optimum.

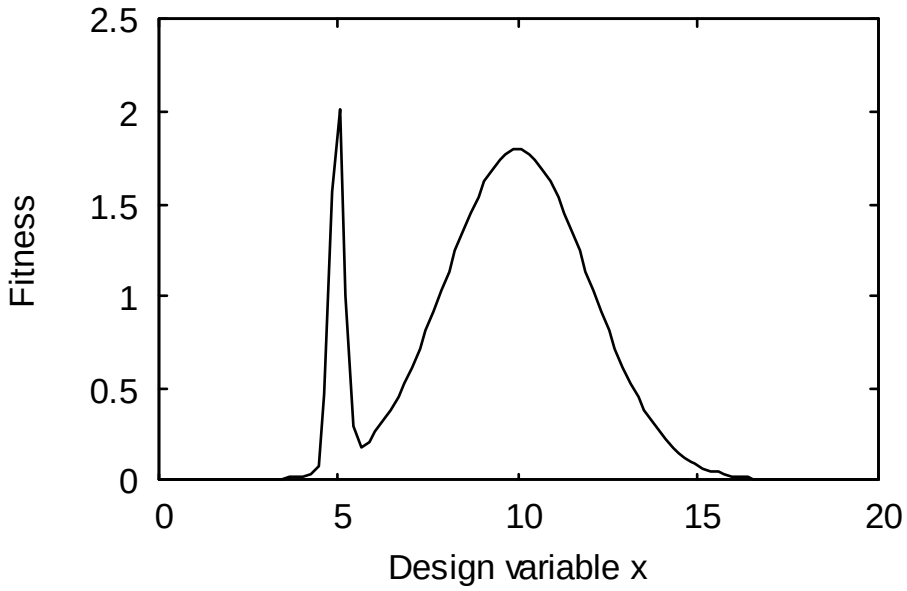


Figure 10 – Example of a fitness landscape for a one-dimensional optimisation problem.

The other problem relates to constraint optimisation. Here, constraints define areas in the input space that do not contain feasible solutions. Figure 3 shows an example of an input space for a two-dimensional constraint optimisation problem.

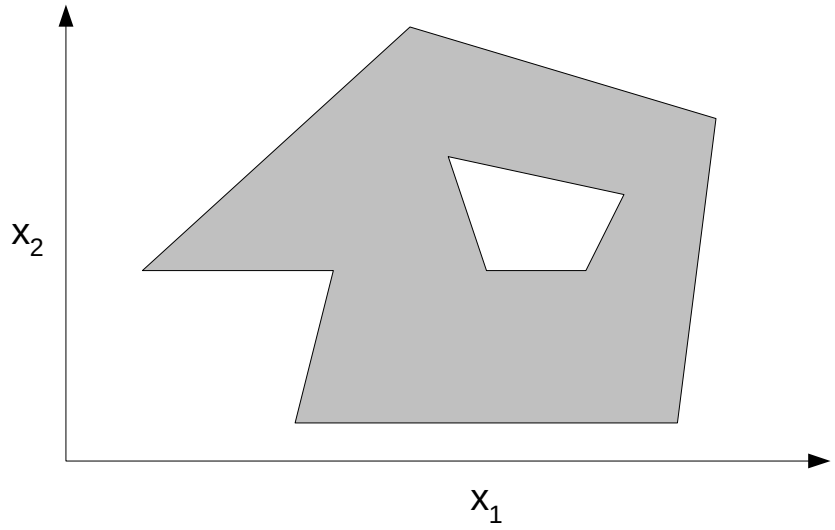


Figure 11 – Example of an input space for a two-dimensional constraint optimisation problem.

In the example, there are two design variables, x_1 and x_2. However, only combinations of x_1 and x_2 that lie in the grey area are feasible, i.e. allowed, whereas combinations that lie in the white areas are not feasible. The white areas are defined by the constraints. The more constraints there are, the more complicated the shape of the allowed area(s) can be and hence the more difficult the optimisation problem.

It is a well-known fact that optimum solutions are normally located at the edges of the feasible space; for example, this forms the basis for the simplex algorithm for linear programming (Murty, 1983). Figure 4 shows an example of a fitness landscape for a one-dimensional constrained optimisation problem.

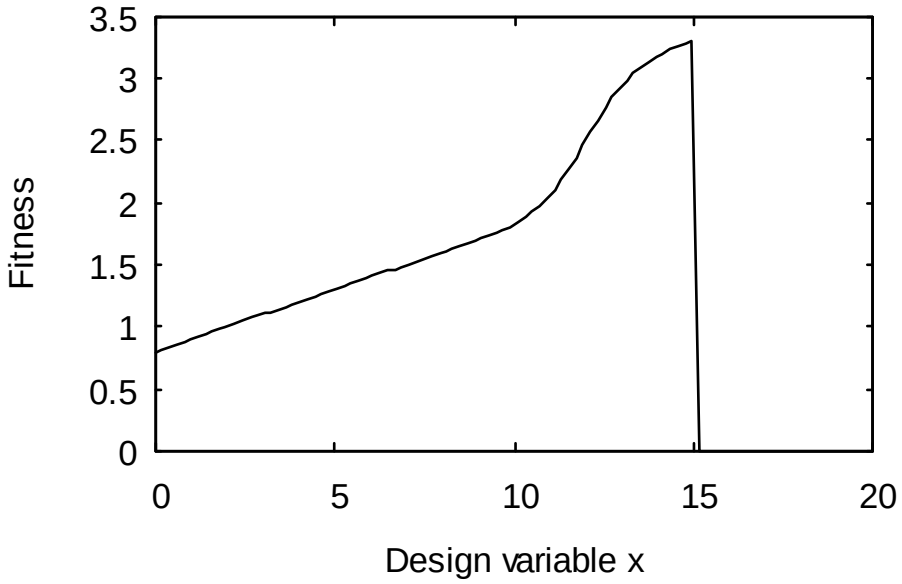


Figure 12 – Example of a fitness landscape for a one-dimensional constrained optimisation problem.

It can be seen that the global optimum is located at $x = 15$. However, all of the points for values $x > 15$ lie outside of the area of feasible solutions and hence, due to penalisation, result in a fitness value of zero. If an algorithm finds the global optimum correctly, but due to the engineering related problems mentioned above the implementation of the solution uses $x + \varepsilon$, all of solutions with a positive ε will cause a constraint violation, i.e. lead to infeasible solutions.

Clearly, it would be of benefit if an algorithm could find a solution close to the global optimum but with a safety distance $d > \varepsilon$. In this case, if the realisation process caused a deviation up to ε, the resulting solution would a) still be close to the global optimum, and b) not violate any constraint.

Both problems, i.e. narrow global optima and constraint optimisation using penalty functions, are equivalent and hence it is possible to address both issues using a single method.

This analysis led to the design of a meta-method that can be used in conjunction with any iterative optimisation algorithm. This method is presented in the next chapter.

3.3 Summary

This chapter introduced an interesting standard benchmark problem from the field of mechanical engineering, the pressure vessel problem, which was introduced by Sandgren (1990). The aim here is to minimise the total cost of the materials used as well as the production costs. The required accuracy for the pressure vessel problem was discussed. The solutions presented in the literature have up to five significant figures after the decimal point. However, the manufacturing process cannot achieve the required accuracy. As a consequence, when the manufactured parameters differ slightly from the design, different fitness values are achieved. Based on these problems, two aspects of robust optimisation were discussed. The first is related to the narrowness of the global optimum for unconstrained optimisation. The other problem relates to constraint optimisation, where constraints define areas in the input space that do not contain feasible solutions. This led to the design of a meta-method for robust design, which can be used in conjunction with any iterative optimisation algorithm. This method is presented in the next chapter.

4 Meta-method of robust design optimisation

Typical engineering optimisation problems deal with high-dimensional and multimodal design spaces. The goal of engineering design should be to find a good enough solution that is a) near-optimal in terms of its fitness criterion, and b) robust enough to maintain that fitness even if the manufacturing process causes slight deviations in the actual design parameter values.

Computational optimisation methods have been proven to find near-optimal solutions in a finite time and with a high degree of repeatability (Schwefel, 1995). These methods should be modified so as to learn from experience, i.e. guided away from dangerous, non-robust areas during the search. However, none of the common standard computational optimisation algorithms has such an in-build facility aimed at robustness. The closest one would be local search, a technique often used to improve the current solution of a global search algorithm by searching the neighbourhood of the current solution (see Section 2.2). If a better solution is found, the solution itself is changed (Lamarckian learning). Alternatively, for genetic algorithms, the fitness value of the original current solution can be improved, which increases the individual's chance of being selected to generate offspring for the next generation (Baldwin learning) (El-Mihoub *et al.*, 2006).

4.1 Narrow peaks

Figure 13 shows, as an example, a contour plot of a fitness landscape for a two-dimensional unconstrained optimisation problem. As can be seen from the figure, the global optimum is located at *(5,5)* and two local optima are located at *(7,7)* and *(8,13)*. Conventionally, one would try to find the global optimum. However, for engineering applications, when implementing the solution, the actual realisation might be slightly off the target value as described previously. Figure 14 presents a contour plot of the same landscape. Here, a solution s_1 is located at the global optimum and a solution s_2 is located at the second best local optimum. Due to the

engineering nature of the problem, the actual implementation might be slightly off but still within a radius ε .

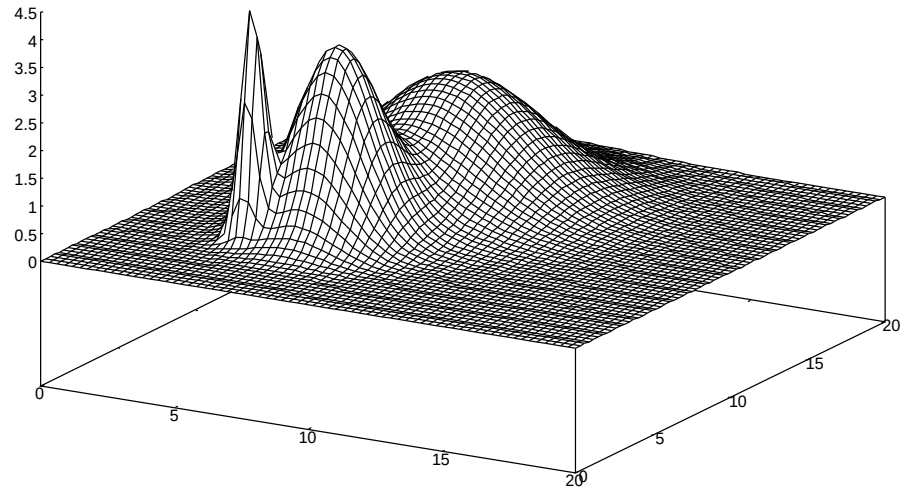


Figure 13 – Multimodal fitness landscape for two-dimensional unconstrained optimisation problem.

As can be seen, most deviations from s_1 will cause the solution to drop dramatically in fitness whereas solution s_2 is more robust in that respect. A local search here would not be of any help since the global optimum s_1 has already been found. However, if, through local probing, a measure of the deviation in fitness for the region around s_1 could be established, it could be used to adjust the fitness accordingly. This would be similar to the Baldwin approach for genetic algorithms but it could be used for any optimisation algorithm, for example the one described in Chapter 2.2.

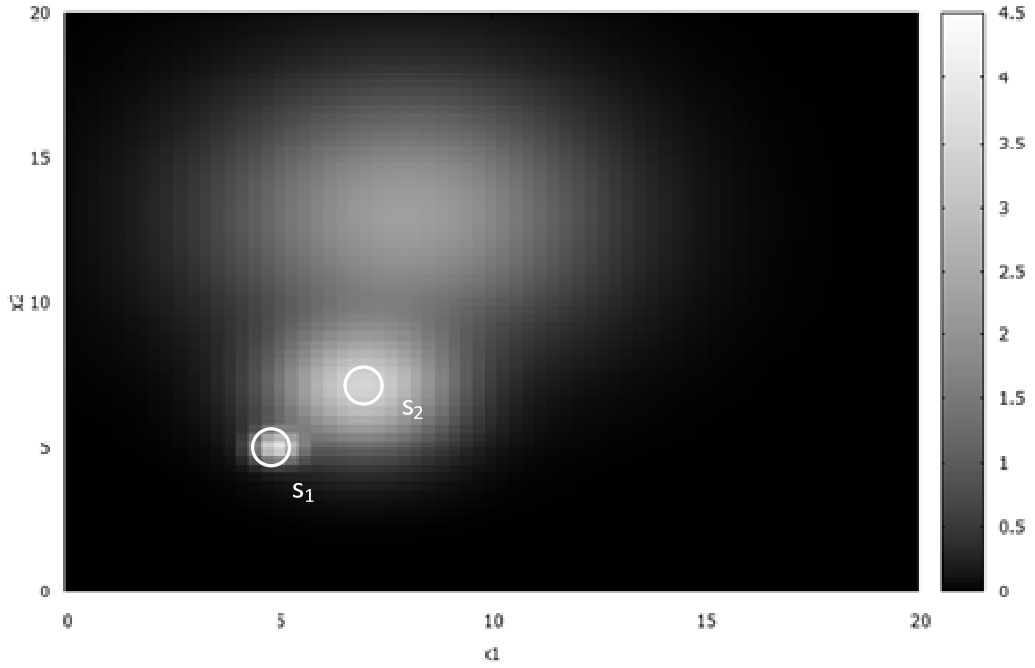


Figure 14 – Contour plot of the multimodal fitness landscape in Figure 13.

Similarly, for constraint optimisation (see below) using penalty functions, this measure could be used to penalise solutions that are too close to the border with the forbidden areas.

4.2 Constraint optimisation

Figure 15 shows two solutions s_1 and s_2 for a two-dimensional constrained optimisation problem. The grey area contains all of the feasible solutions; white areas are forbidden.

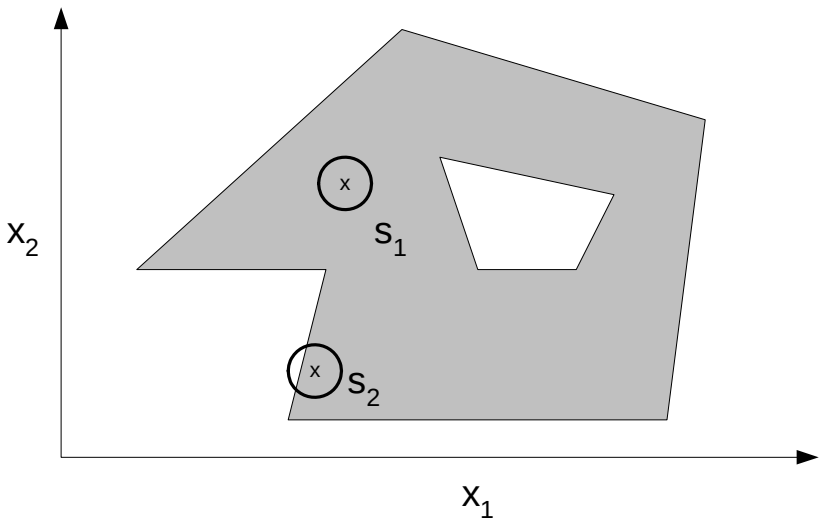


Figure 15 – Two different solutions in a constrained search space.

It can be seen that probing the areas with a radius ε around the solutions would lead to a penalisation for s_2 , because a section of the circle is in the forbidden zone. On the other hand, s_1 would not be subject to such a penalty, because none of the points within the neighbourhood $\leq \varepsilon$ would cause a violation of a constraint.

As shown above, both problems can be solved by using a measure of the quality of the solutions contained in the neighbourhood of a solution. For unconstrained optimisation, this penalty would change the shape of the effective fitness landscape so that narrow peaks are penalised. For constrained optimisation, it would change the shape of the fitness landscape so that points near to or on the border of a forbidden region would be lowered in fitness or, in other words, the edges would be 'rounded off' and a fitness barrier would be produced to protect the solutions from leaving the feasible space when realised through an engineering process
(Figure 16).

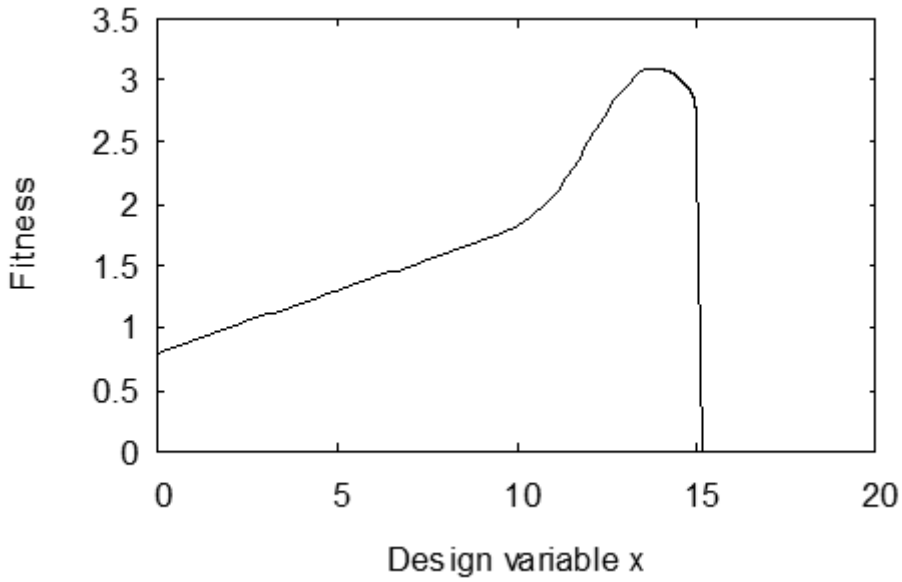


Figure 16 – Fitness barrier to protect solutions from dropping into the forbidden area.

To overcome this problem, a meta-heuristic was developed, which is explained in the next section.

4.3 A meta-heuristic for robust optimisation.

Figure 17 shows a flowchart of the meta-heuristic for engineering applications based on Baldwinian learning.

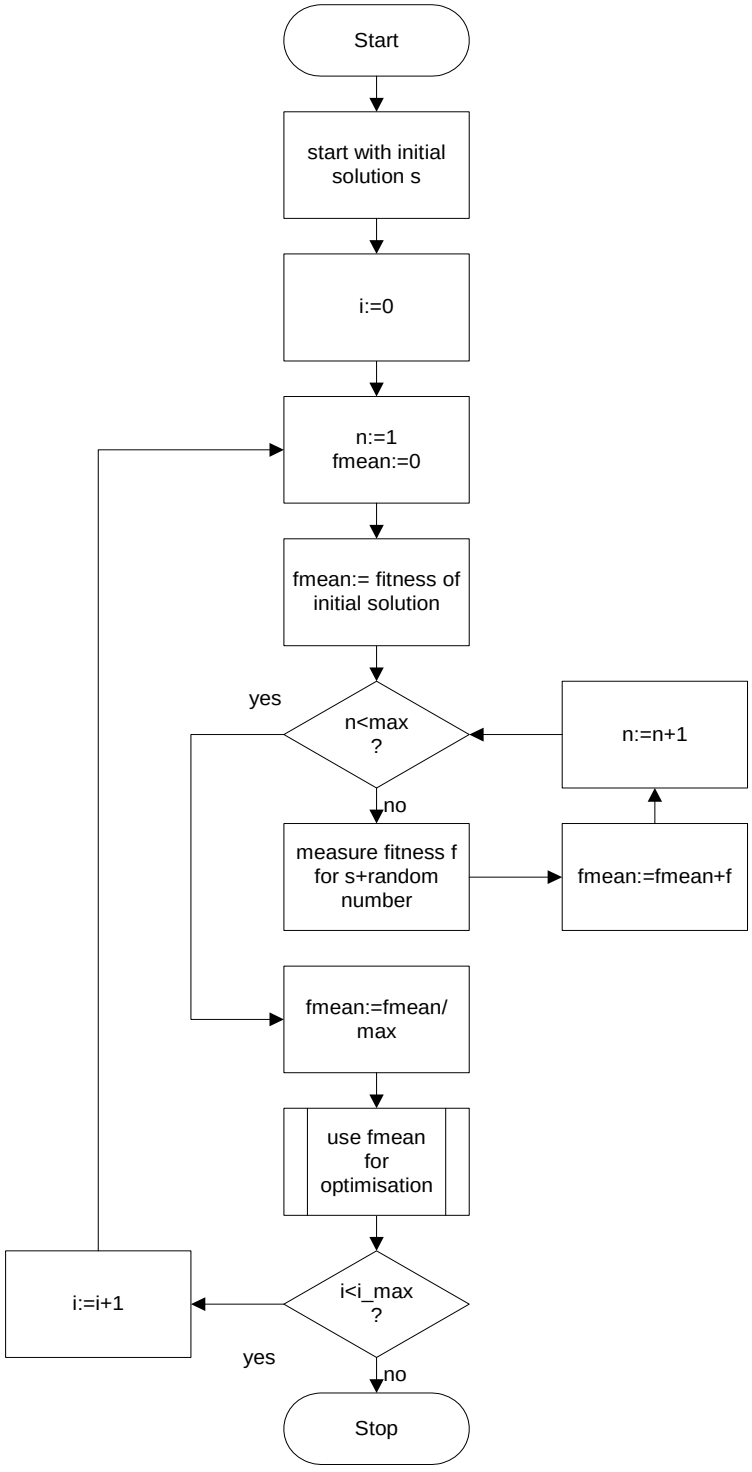


Figure 17 – Baldwinian-based meta-heuristic for engineering optimisation.

The search begins with an initial solution, respectively with an initial population of solutions. Each time a solution is due to be evaluated by the optimisation algorithm being used, its fitness is evaluated first and then the neighbourhood with the radius ε is sampled with random trials. The average fitness of all of the trials is calculated and used by the host optimisation algorithm instead of the fitness for the original solution. Figure 18 shows the optimisation loop for the Baldwinian-based meta-heuristic.

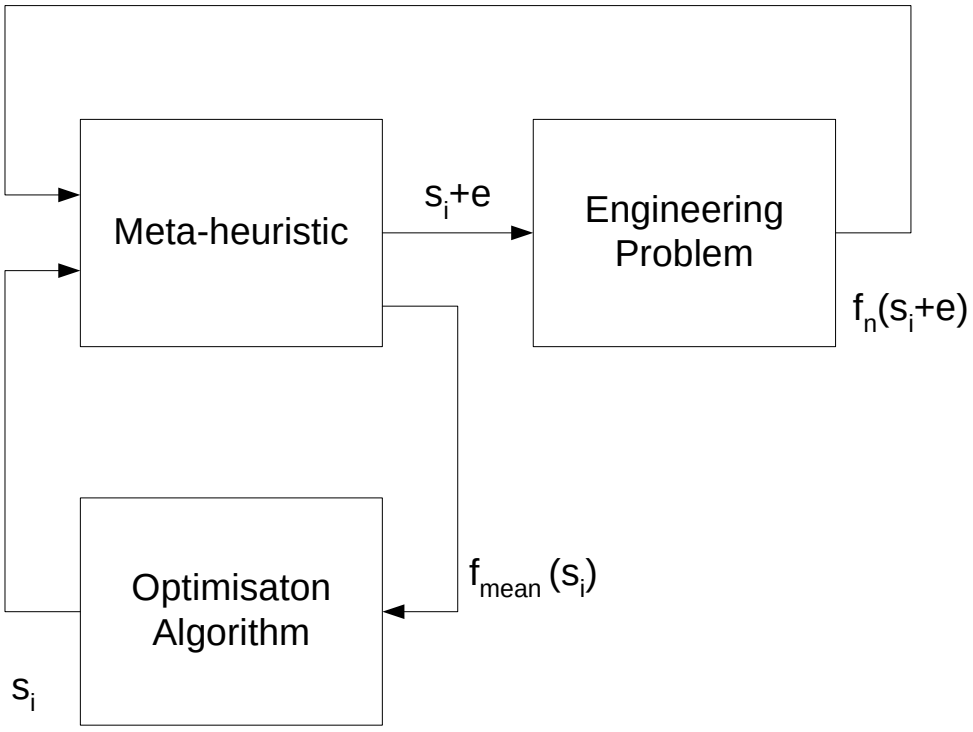


Figure 18 – Optimisation loop for Baldwinian-based meta-heuristic.

It can be seen that the optimisation algorithm, which could be of any type, does not receive a quality measure directly from the engineering problem. Instead, it sends its solution to the meta-heuristic, which in turn presents a number of slightly different versions of the solution to the problem and calculates the average fitness, including any penalties if applicable. This average fitness is then used by the optimisation algorithm for decision making. This meta-heuristic, referred to as Baldwinian-based meta-heuristic (BMH), was tried in combination with PSO and SASS. These two algorithms were selected because of their low number of control parameters

(see Chapter 2.3), which makes them suitable for engineers who are inexperienced in the field of computational optimisation. These two methods are introduced in the next sections.

4.4 Basic PSO

Fish, ants, and many other animals search in swarms for rich feeding grounds. Besides individual searches (cognitive part), each of them is orientated to the other swarm members in their direct environment (social part). A particle swarm optimisation algorithm (PSO) (Chen and Li, 2007; Yoshida *et al.*, 1999) mimics this social behaviour of e.g. ant colonies or fish schools, with a stochastic, population-based recursion procedure. This heuristic technique belongs to the family of swarm intelligence computational techniques and was first introduced by Kennedy and Eberhart in 1995. It quickly emerged that some unique features make PSO very efficient in solving optimisation problems related to science and engineering (Jordehi *et al*., 2015; Liao *et al.*, 2007; Yin *et al.*, 2009). Compared with classical approaches such as linear and non-linear programming, PSO algorithms obtain much higher efficiency and do not require continuity and differentiability of the objective function (Wu and Tsai, 2008). Also among other bio-inspired approaches like genetic algorithms, evolution strategies, or differential evolution, PSOs show interesting features that make them suitable and efficient to use (Abido, 2001). PSO, for example, has less parameters that need to be tuned, faster convergence rates, easier coding, higher accuracy, and less computational effort than the most classical and heuristic approaches.

The basic procedure of PSO is very simple and starts with random initialisation of a swarm of individuals in the n-dimensional search space. Each particle moves through the search space with adjustable velocity and keeps two possible values in its memory. Value one is the best experience of the individual itself or to be more precise, the best fitness value of the individual. Value two is the best experience of the entire swarm.

In detail, the procedure is as follows. Each search candidate is defined as a particle, which has the actual position $\overrightarrow{x_i}$ within the search space and the velocity $\overrightarrow{v_i}$, where $i = 1, \ldots, m$. Here, m is the number of decision variables.

In each iteration, the velocity of the particle is calculated as follows (Equation 9):

$$\vec{v}_i(t+1) = \omega \cdot \vec{v}_i(t) + \beta_1 \cdot \left(\vec{x}_i^{(local)}(t) - \vec{x}_i(t)\right) + \beta_2 \cdot \left(\vec{x}_i^{(global)}(t) - \vec{x}_i(t)\right) \quad (9)$$

The parameters β_1 and β_2 are random numbers. Their ranges are determined by two constants c_1 and c_2, which need to be selected by the user. The inertia weight ω also has to be adjusted to the problem. After the new velocity vector is computed, the position of the particle is updated using Equation 10.

$$\vec{x}_i(t+1) = \vec{x}_i(t) + \vec{v}_i(t) \quad (10)$$

$\vec{x}_i^{(local)}$ in Equation 9 denotes the local memory of each particle and it has the information about the best position within the search space, which the selected particle has identified so far (Equation 11).

$$\vec{x}_i^{(local)}(t) = \vec{x}_i\left(argmax_{u=1}^{t} f\left(\vec{x}_i(u)\right)\right) \quad (11)$$

$\vec{x}_i^{(global)}$ represents the global memory of the complete swarm, i.e. it holds the information about the best position within the search space where any of the particles have been so far, which means:

$$\vec{x}^{(global)}(t) = \vec{x}_j^{(local)}(t) \text{with } j = argmax_{i=1}^{m} f\left(\vec{x}_i^{(local)}(t)\right) \quad (12)$$

Figure 19 shows pseudo-code for the basic PSO algorithm.

```
Procedure PSO                                          //particle swarm optimization
For each particle i do Begin                           //initialize position of each
      choose random x_i ; v_i = 0;                     //particle
End                                                    //random in search space
Repeat                                                 //move swarm particles
      For each particle i do Begin                     //pass through all particles
            y := f(x_i);                               //compute function at position of
            If y ≥ f(x_i^(local)) then x_i^(local) := x_i;   //particle
            If y ≥ f(x_i^(global)) then x^(global) := x_i;
      End                                              //update local/global memory
      For each particle do Begin                       //pass through all particles
            v_i := α · v_i + β_1 · (x_i^(local) − x_i) + β_2 ·   //again
            (x^(global) − x_i);
            x_i := x_i + v_i;
      End                                              //update velocity and position of
Until stopping criterion is met;                       //each particle
```

Figure 19 – Pseudo-code for PSO.

4.5 Basic SASS

In many engineering design problems, a search based optimisation is used to maximise or minimise the system's vectors. In most cases, important information about the system, like transfer functions or a function's derivatives, is not available and the use of different heuristic computational optimisation algorithms such as GA (Matousek *et al.*, 2007; Nolle *et al.*, 1999), SA (Nolle *et al.*, 2002), ACO (Nolle *et al.*, 2008) becomes inevitable. A problem of using these algorithms in a practical and engineer-friendly manner is the lack of standard methodology for choosing a matching algorithm for a particular design problem. Furthermore, the process of choosing, tuning and applying proper optimisation techniques requires specialist knowledge and a large number of computational experiments. These circumstances and their lack of experience make it difficult for engineers to involve acceptable and novel algorithms in their optimisation problems.

To overcome these problems, more generally applicable, efficient, and effective algorithms, which require less effort to tune the control parameters, would be beneficial. An approach proposed by Nolle and Bland (2012) is a further development of a hill climbing algorithm (Hopgood, 2001), named self-adaptive step size search (SASS). It combines effectiveness, wide range applicability and efficiency with a minimum number of parameters

that need to be tuned. The only parameter that can be tuned in SASS is the number of particles. This results in an approach that is easy to use even for inexperienced engineers.

It was recently shown by Nolle (2004) that the definition of the neighbourhood and the chosen step size are important parameters for the success and performance of the algorithm. It was demonstrated that using a random step size limited by a maximum step size smax outperforms other methods using a fixed step size. In particular, when the search space was too large to mind direct neighbours of a candidate solution, the performance of the optimisation declined and an adaptive step size became inevitable . In SASS, the neighbourhood of a particle pi is defined by the space between the particle itself and a randomly chosen sample particle *si* of the population. Because the initial population is distributed over the entire search space, the space between *si* and *pi* is typically very large at the beginning of the search process. During the search process, every single particle is attracted to a local optimum and the population is accumulated around a set of optima. For the case that *pi* and *si* are located in varying clusters, pi can escape the local optimum to achieve higher fitness. At the end of the search process, most particles will be in the region of the global optimum and the distance between *pi* and *si* will be much smaller than in the initial population. The sequence of the different steps is shown in the following pseudo code (Figure 20).

```
Procedure selfAdaptiveStepSizeSearch

Begin

    initialise population of n particles
    While stopping criterion not met
    Begin
           For every particle p in population
           Begin
                  select random particle s ≠ p
                  For every component pi in particle p
                  Begin
                         smax ← | pi – si |
                         generate random value r Î [–smax; +smax]
                         p'i ← pi + r
                  End
                  If f(p') better than f(p) then p ← p'i
           End
      End
      Return best result
End
```

Figure 20 – Pseudo-code for SASS.

4.6 Comparison of BMH/PSO and BMH/SASS

Both, PSO and SASS were combined with the meta-heuristic presented in Section 4.3. Since standard test functions, like De Jong's functions or the Rosenbrock function (Schwefel, 1995), do not exhibit the desired properties, the resulting hybrids were tested on an artificial test function (Equation 13).

$$f(x,y) = 2^2 \cdot e^{\left(-\frac{x-5}{0.5}\right)^2} \cdot e + e^{\left(-\frac{y-5}{0.5}\right)^2} + 1.8^2 \cdot e^{\left(-\frac{x-7}{2}\right)^2} \cdot e^{\left(-\frac{y-7}{2}\right)^2} + 1.5^2 \cdot e^{\left(-\frac{x-8}{4}\right)^2} \cdot e^{\left(-\frac{y-13}{4}\right)^2} \qquad (13)$$

This multi-modal function, which is also depicted in Figure 13, was designed to contain a very narrow global optimum and two local ones that are wider. This represents a typical engineering design scenario, where the global optimum is not the most favourable for designers. It was also designed so that it could be visualised easily.

For both algorithms the number of particles was arbitrarily set to 50 and the number of iterations was set to 1000. The sample size for the meta-heuristic was set to 5 and the safety distance ε was set to 0.1.

Before PSO could be applied, three control parameters, omega, c1 and c1 had to be selected. To avoid extensive tuning, the constants c_1 and c_2 were both set to 2, which is a

common value for PSO (Bansal et al., 2011). The inertia weight ω cannot be set to a default value, because its setting has to be adjusted to the problem at hand.

To determine a suitable value for the inertia weight, it was changed from 1 to 0.0001. Each value was applied 100 times. Figure 21 shows the influence of ω on the average fitness and its standard variations.

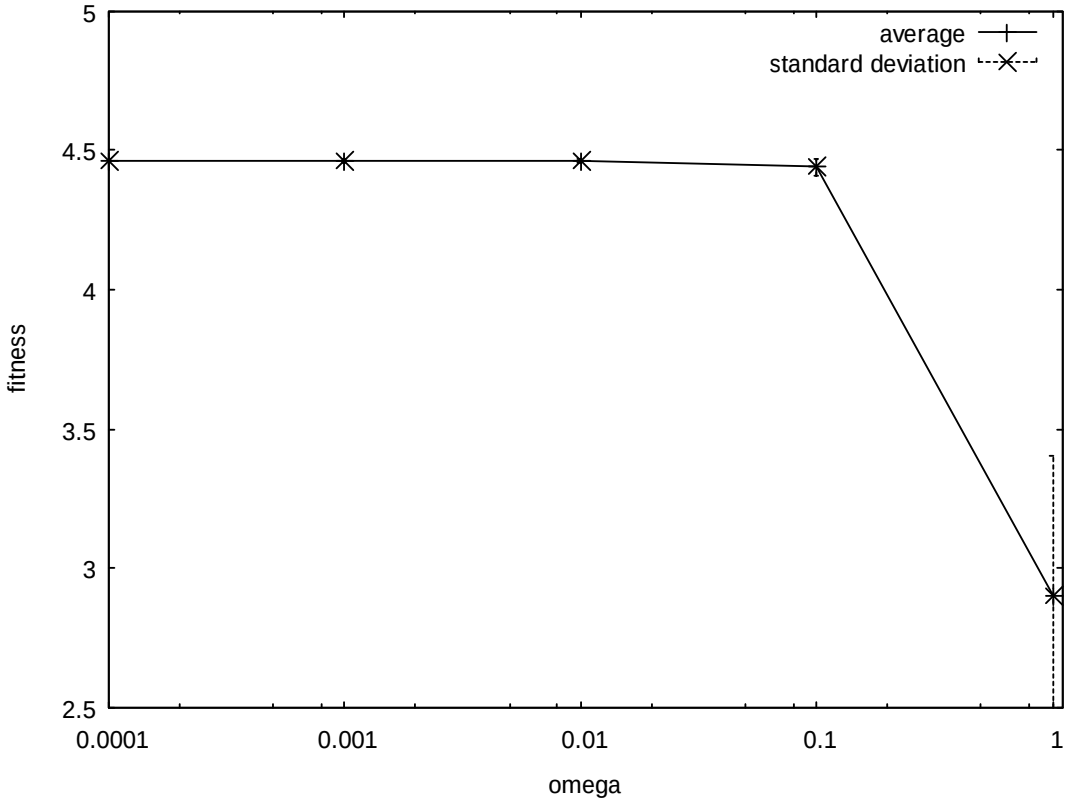


Figure 21 – Influence of omega on fitness for PSO.

It can be seen that for values above 0.01 the standard variation increases and that for values above 0.1 the average fitness drops dramatically. Therefore, a value of 0.001 was chosen for the experiments.

Figure 22 shows a plot of the locations of the solutions for BMH/PSO and PSO. Each algorithm was run 100 times. It can be seen that PSO found the global optimum at *(5,5)* very reliably. However, this is a very narrow global optimum, which should be avoided for robust

applications. BMH/PSO, on the other hand, found the second best optimum in most runs. This second best optimum is much wider and hence preferable to the global optimum.

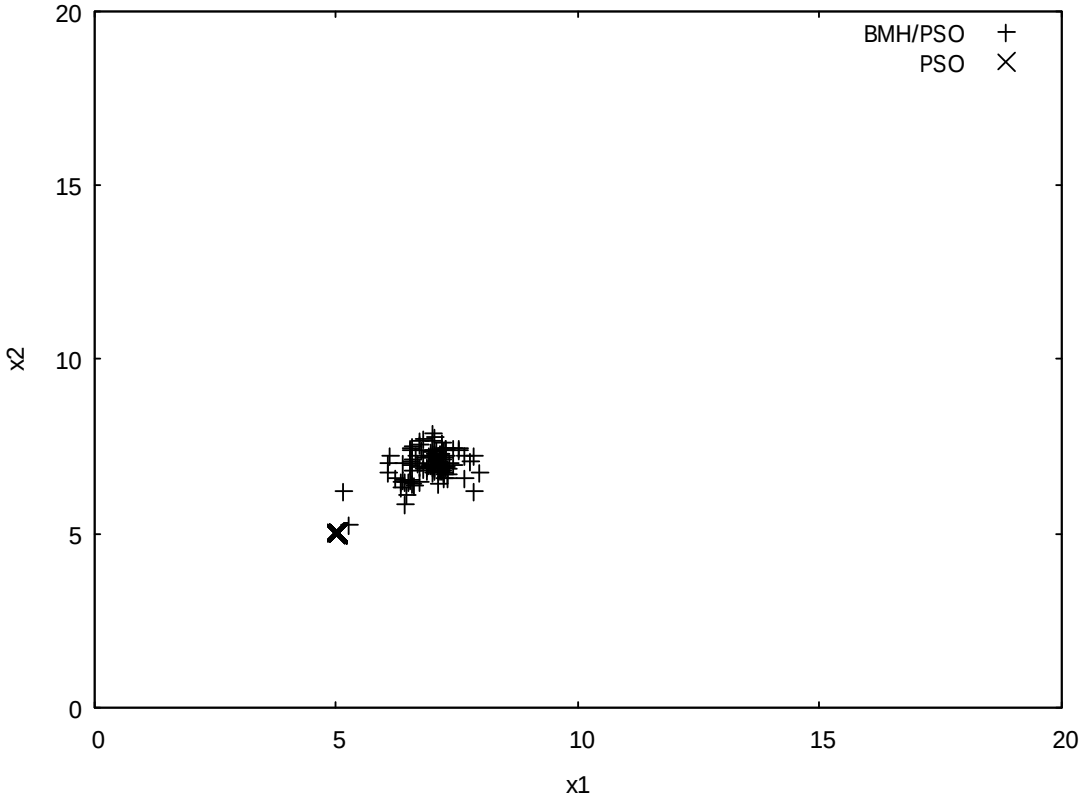


Figure 22 –Comparison of PSO and BMH/PSO.

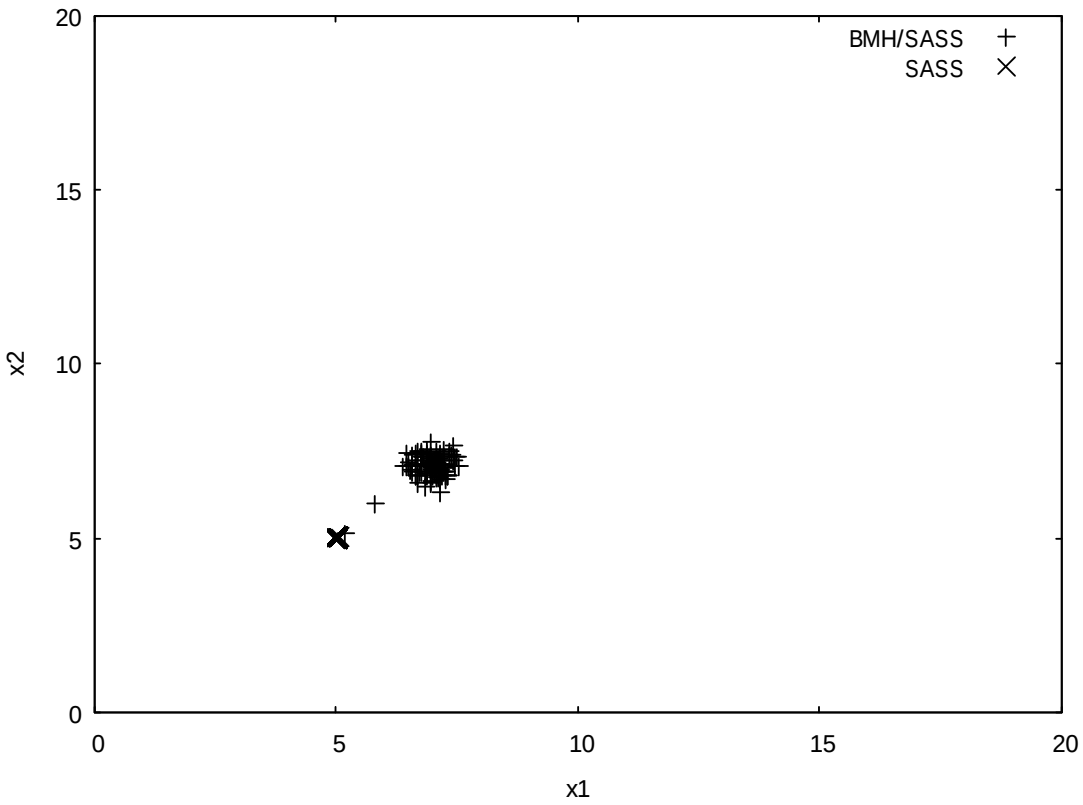


Figure 23 – Comparison of SASS and BMH/SASS.

Figure 23 shows a plot of the locations of the solutions found by BMH/SASS and SASS. Here, too, SASS found the global optimum at *(5,5)* very reliably whilst BMH/SASS converged towards the preferable optimum at *(7,7)*.

Comparing BMH/PSO and BMH/SASS it can be said that both hybrids found robust solutions in most of the experiments. The area in which the solutions are located is slightly smaller for BMH/SASS. This algorithm also has the advantage that no tuning is required prior to optimisation. In conclusion, it can be said that both hybrids found good robust solutions in most cases and hence are suitable for robust engineering optimisation applications.

4.7 Summary

This chapter described problems caused by inaccuracies in engineering processes, i.e. manufacturing processes, for real-world applications of computational optimisation techniques. The case study presented in Chapter 3 was used to demonstrate this problem. Based on an analysis of the problem, a novel meta-heuristic was proposed, based on Baldwinian learning, to

temporarily shape the fitness landscapes so that solutions are still fit for purpose even if the implemented solution deviates slightly from the theoretical one due to the nature of the manufacturing processes. The meta-heuristic was tested in combination with two standard optimisation algorithms on an artificial benchmark. These tests showed that the meta-heuristic is capable of reliably guiding direct search algorithms towards robust solutions. This meta-heuristic is used in the next chapter to find robust solutions to the highly constrained optimisation problem of pressure vessel design.

5 Experiments

In this chapter, the new meta-heuristic proposed in Chapter 4 was applied to the pressure vessel design. The results of the experiments are presented and discussed.

5.1 Experimental set-up

The meta-heuristic presented in the previous chapter can be combined with any direct search algorithm, i.e. any algorithm that uses an optimisation loop to adjust current solutions using the fitness currently observed (see Chapter 1). For the experiments, self-adaptive step size search (SASS) (Nolle, 2006; Nolle and Bland, 2012) was chosen because it has only one control parameter, which is the number of particles. Hence, the experiments would not be influenced by the meta-optimisation problem of tuning control parameters, for example genepool size, crossover probability etc. as in the case of genetic algorithms (Chapter 2.3). SASS has also proven to be effective and efficient for the pressure vessel problem (Nolle and Bland, 2012). Figure 24 shows the optimisation loop for the pressure vessel problem, using SASS as a direct search algorithm.

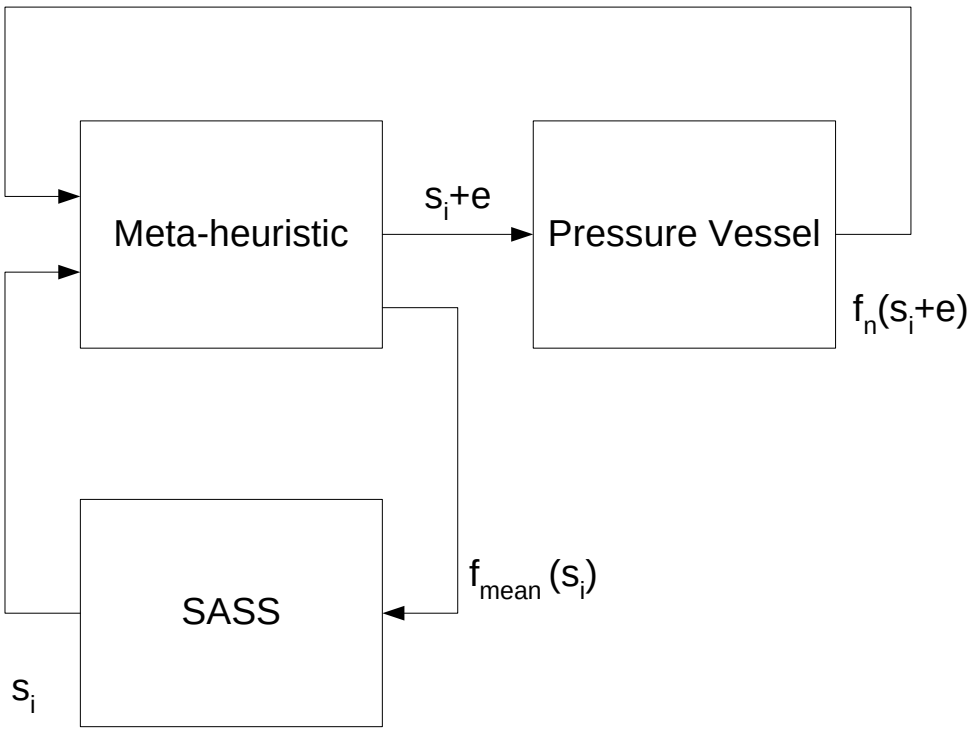


Figure 24 – Optimisation loop for the pressure vessel experiments.

For SASS, the same control parameters were used as reported by Nolle and Bland (2012) to allow a fair comparison. Hence, the number of particles was set to 16 and the maximum number of iterations was initially set to 25,000.

Before carrying out the experiments, it was decided to limit the number of places after the decimal; for the pressure vessel problem, the design parameters x3 and x4 are continuous. For practical applications, a solution with a large number of significant figures behind the decimal point cannot be implemented, because of the accuracy of the engineering processes involved, for example +/- 0.0002 inches for modern CNC machines (Pullarcot, 2002). Therefore, only four places behind the decimal point were used for the experiments.

The new method, referred to as Baldwinian-based meta-heuristic (BMH), has two degrees of freedom, which are the number of trials or the sample size, and the range of the samples around a solution, i.e. perturbation epsilon. These had to be determined empirically as described below.

5.2 Determining the control parameters

In this set of experiments, the influence of the sample size and epsilon range on the effectiveness of the BMH, i.e. the average fitness achieved, was studied. The outcome was used to determine the settings for the actual optimisation, which is discussed in the next section.

The sample size s is the number of trials that the BMH algorithm carries out in order to estimate the average fitness of a solution. The sample size was varied from one to 50 in steps of 10. This was repeated for epsilon=0.01%, 0.1% and 1.0% of the individual ranges for each dimension of the search space. For comparison, a sample size of one was included, which is equivalent to not using the meta-heuristic, because the first sample is always the original solution. Each experiment was repeated 50 times and the average fitness and standard deviation were calculated.

Figure 25 shows the sample size s versus the average fitness for epsilon = 1.0% of the search space dimension length. It can be seen that with a larger number of samples the average fitness values increase and converge towards around 6240.

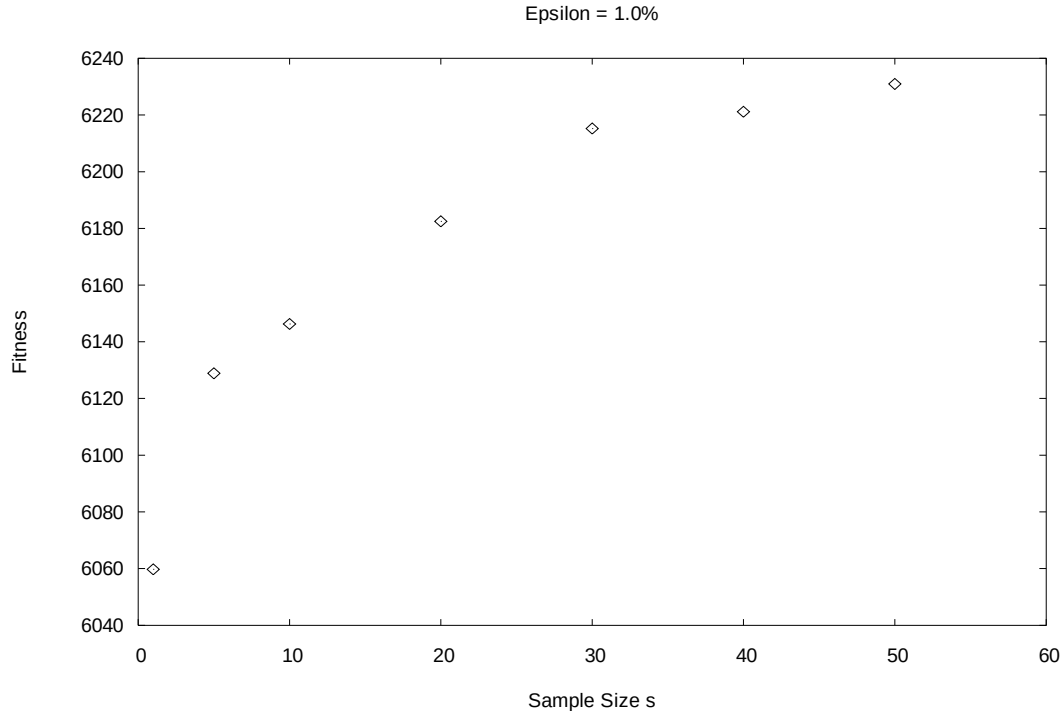


Figure 25 – Sample size versus fitness for epsilon = 1.0%.

Since the optimisation problem at hand is a minimisation problem, lower fitness values are related to better solutions.

In terms of the standard deviation of the fitness (Figure 26) it can be seen that larger sample sizes result in lower standard deviations from the mean values, apart from an outlier at s = 40. Therefore, larger sample sizes result in better reproducibility of results. The same behaviour can be observed in Figure 27 and Figure 28 for epsilon = 0.1% of the search space dimension length and in Figure 29 and Figure 30 for epsilon = 0.01% respectively. For epsilon = 0.1%, there is an outlier in standard deviation for s = 10. For epsilon = 0.01%, there are outliers for both the average fitness and the standard deviation for s=20.

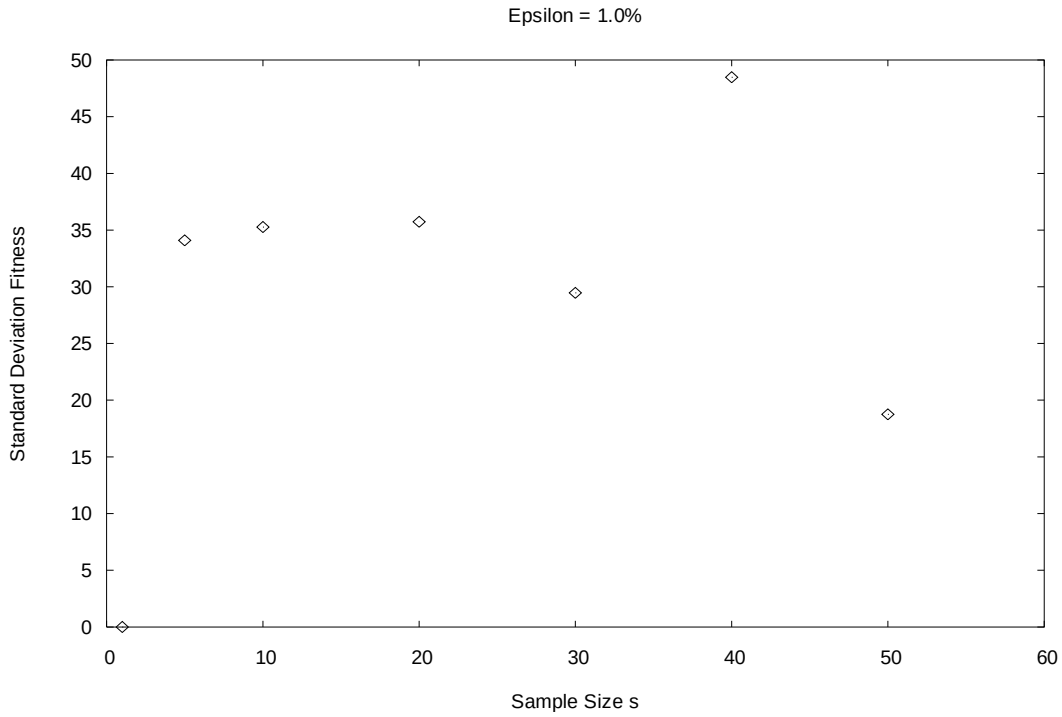


Figure 26 – Sample Size versus standard deviation of the fitness for epsilon = 1.0%.

Based on the results from this set of experiments, the sample size for the pressure vessel design problem was chosen to be 10 and epsilon was chosen to be 0.01% of the search space dimension range. These parameters were used for the optimisation and are described below.

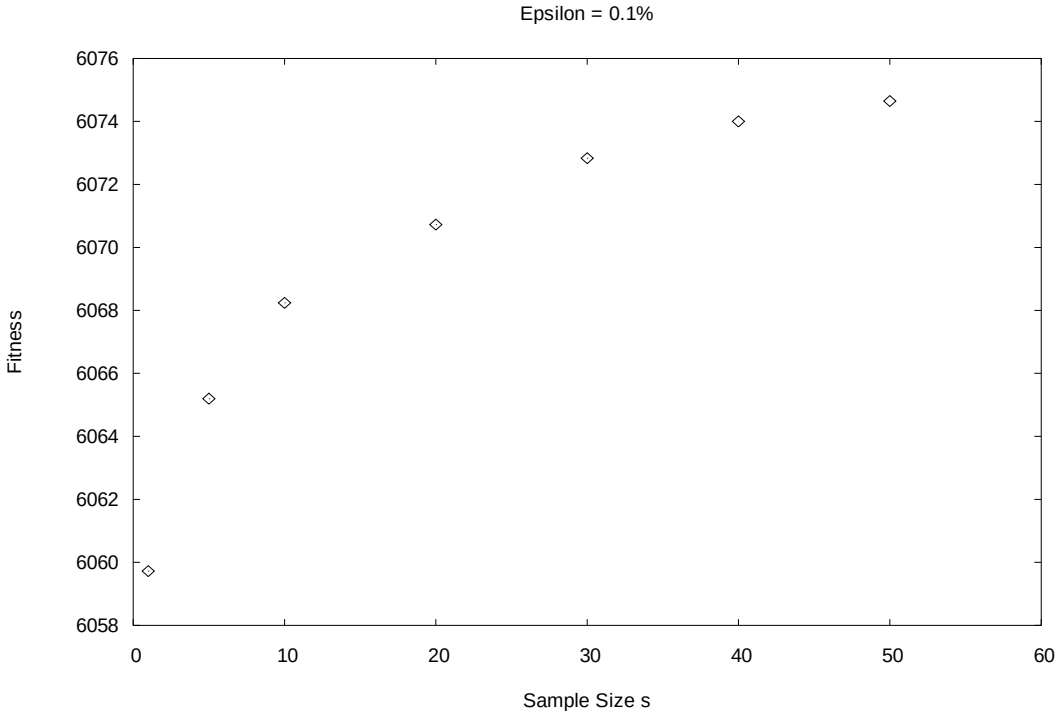


Figure 27 – Sample size versus fitness for epsilon = 0.1%.

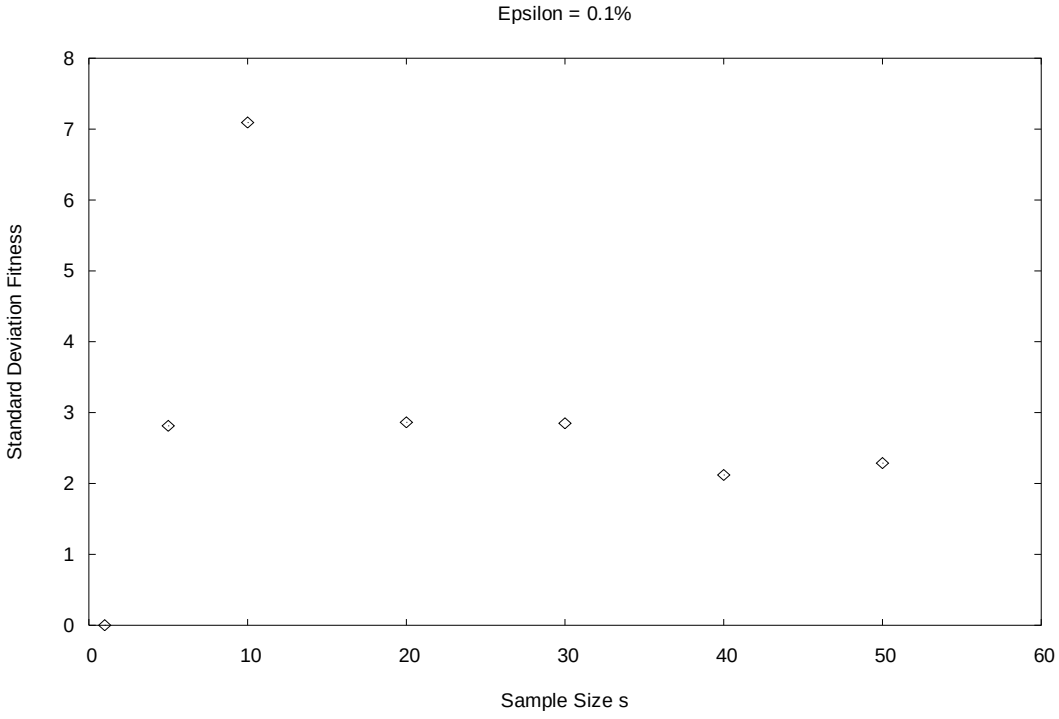


Figure 28 – Sample size versus standard deviation of fitness for epsilon = 0.1%.

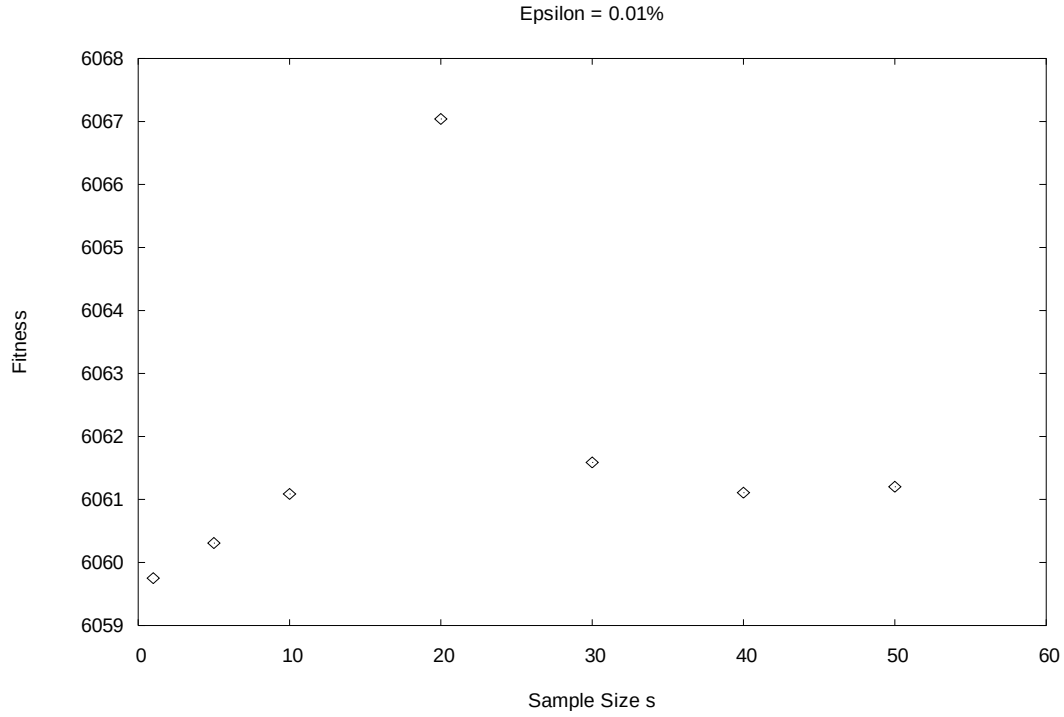


Figure 29 – Sample size versus fitness for epsilon = 0.01%.

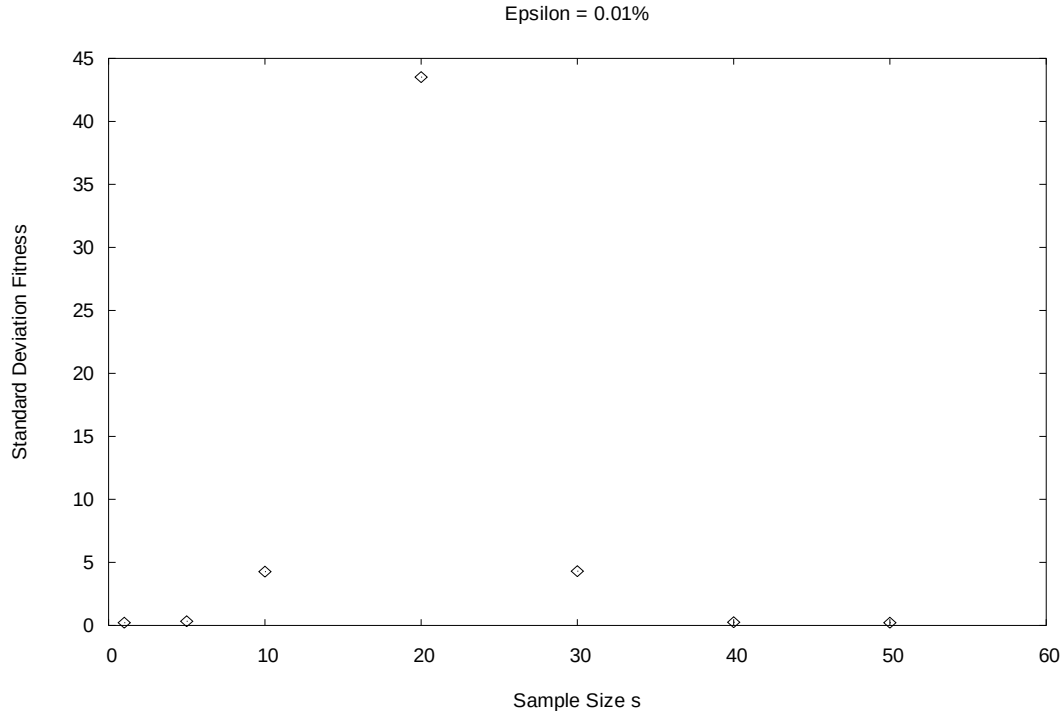


Figure 30 – Sample size versus standard deviation of fitness for epsilon = 0.01%.

It can clearly be seen that the best results were achieved using epsilon = 0.01% of the search space; for small values of s, the average fitness is almost as good as the fitness achieved without the meta-heuristic (sample size = 1). And even for larger values, the average fitness seems to have converged against a value of around 6061.

5.3 Experimental Results

During the previous experiments the number of iterations was kept at 25,000, as reported by Nolle and Bland (2012). Next, the maximum number of iterations necessary for BMH was determined. For this, five experiments were carried out with the maximum number of iterations ranging from 12,000 to 100,000. Each experiment was repeated 50 times and the findings are depicted in Figure 31.

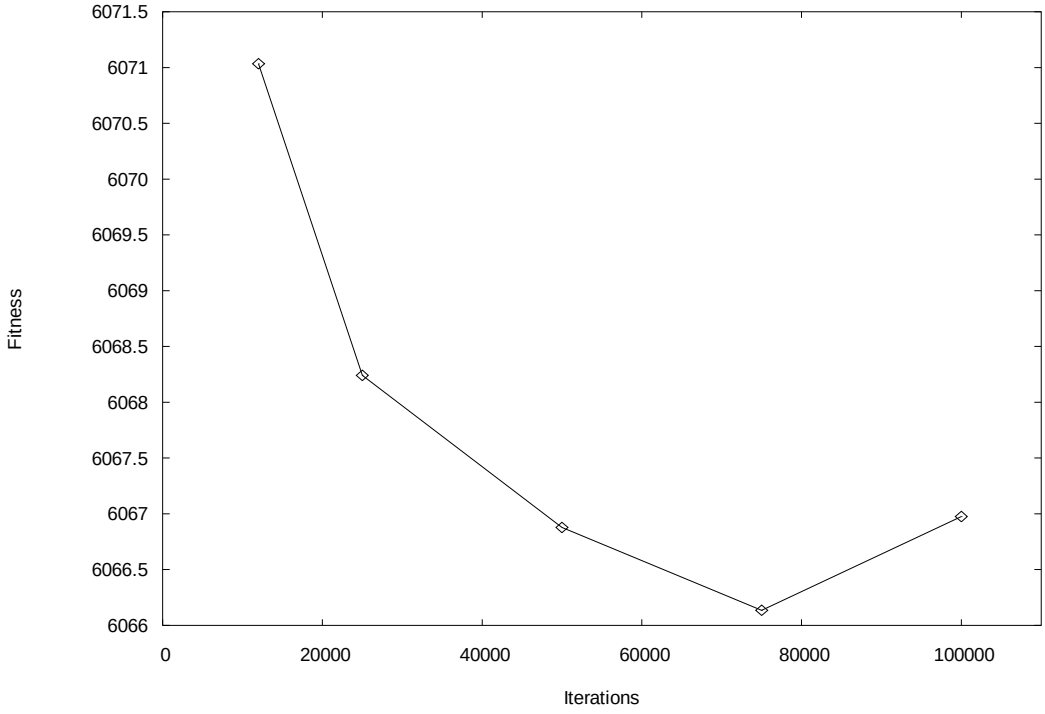


Figure 31 – Iterations versus average fitness.

It can be seen that the average fitness achieved drops to approximately 6066 after around 50,000 iterations. Figure 32 shows a correlation diagram for the maximum number of iterations versus the number of iterations needed on average to find the optimum solution. It can be seen that after around 50,000 iterations the algorithm found the best solution before the maximum

number of iterations was reached, whereas below 50,000 iterations the algorithm constantly improved the solution.

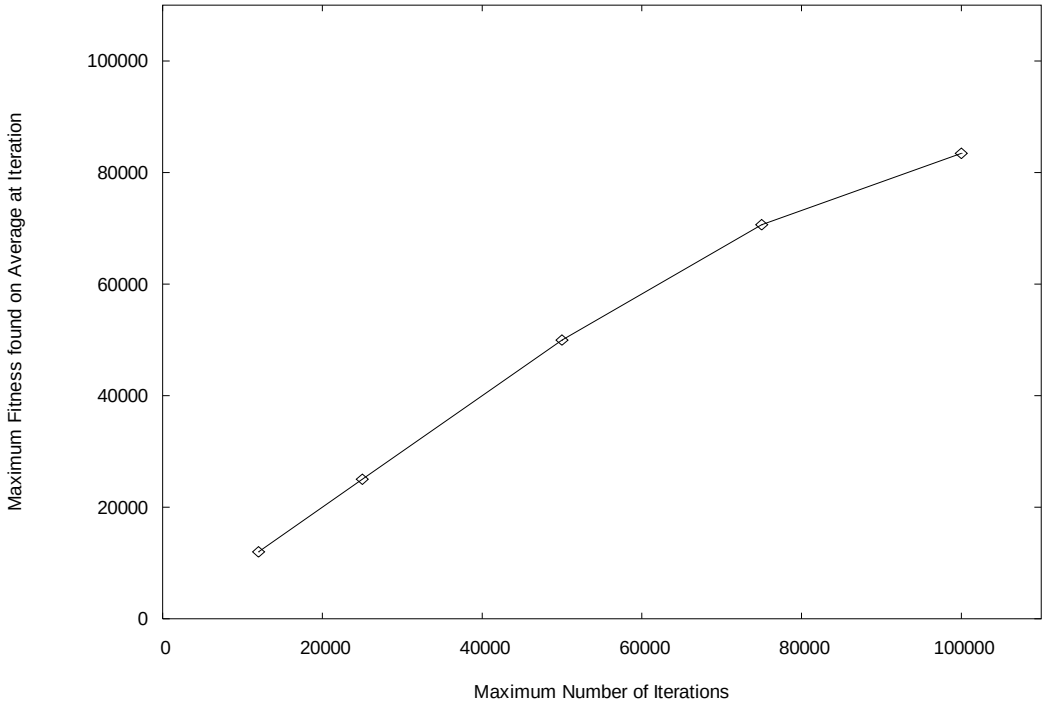


Figure 32 – Maximum number of iterations versus average iteration needed.

Therefore, the maximum number of iterations was chosen to be 75,000 for the experiments. After 50 runs of the algorithm, an average fitness of 6065.3908 was achieved with a standard deviation of 2.5375. Figure 33 shows a conversion plot of a typical run. It can be seen that the average fitness improves up to around 65,000 iterations, whereas the best fitness had already been found after approximately 9,000 iterations.

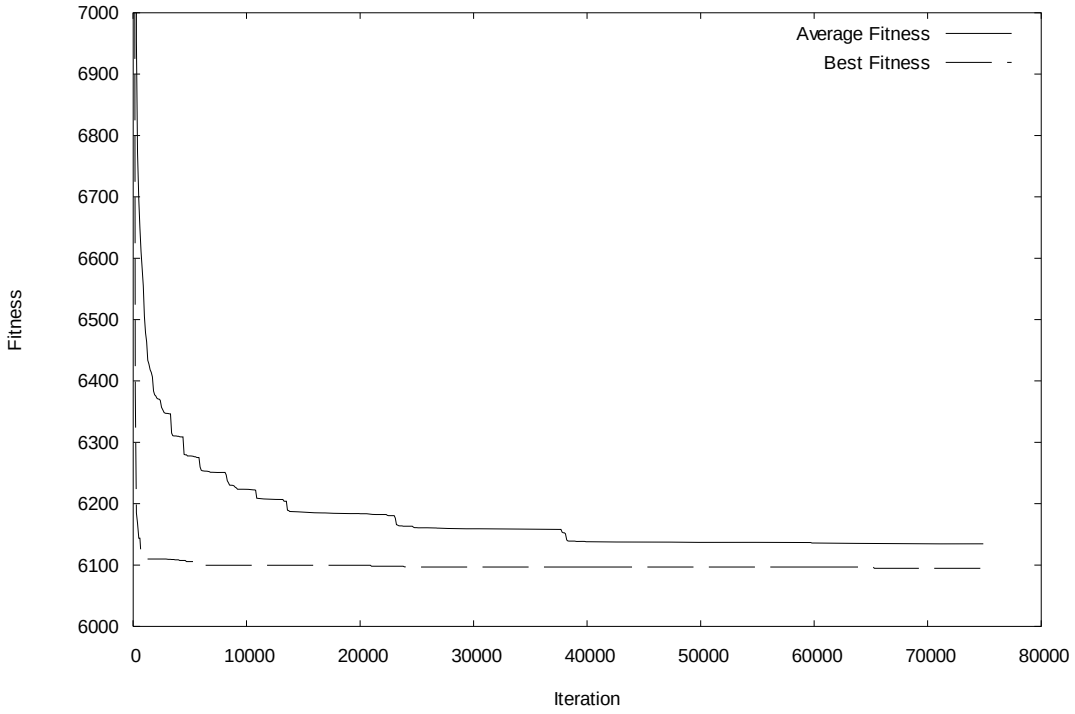


Figure 33 – Conversion plot for a typical run of BMH.

In the next section, the best solutions found during the 50 runs of the BMH/SASS algorithm are compared with the best solutions reported in the literature.

5.4 Comparison of results

In order to evaluate the solutions, a Monte Carlo simulation (Rubinstein, 1981) was developed; the manufacturing process was simulated by randomly changing the optimal solutions found by different optimisation algorithms. These algorithms were: Particle Swarm Optmisation (PSO) (He and Prempain, 2004), Hybrid Particle Swarm Branch-and-Bound (HPB) (Nema *at al.*, 2008), Self-Adaptive Stepsize Search (SASS) (Nolle and Bland, 2012) and the new Baldwinian-based meta-heuristic on top of SASS (BMH/SASS). Apart from the latter method, the best solutions reported in the literature were used and rounded to four decimal places. The perturbation epsilon was randomly chosen from the intervals presented in Table 6, which represent the technical limitations in pressure vessel manufacturing (Pullarcot, 2002). A uniform distribution of random numbers was used.

Table 6 – Upper and lower bounds of the perturbation epsilon.

Limits [inches]	x_1	x_2	x_3	x_4
Lower limit	-0.0118	-0.0118	-0.1969	-0.2756
Upper limit	0.0472	0.0472	0.1969	0.2756

The quality criterion here was the percentage of pieces produced that were rejected, i.e. that violated one or more constraints. The lower the number of rejected goods the better the solution.

The solutions produced by each algorithm were virtually implemented 100 times using the random perturbation as described above. Table 7 shows a comparison of the results achieved.

Table 7 – Comparison of results.

Limits [inches]	**PSO**	**HPB**	**SASS**	**BMH/SASS**
Passed [%]	42	40	40	53
Average Fitness	6298.09	6298.72	6318.39	6284.24

5.5 Discussion

As can be seen from Table 7, out of the 100 pieces virtually produced, only 40 passed the quality check in the case of SASS and HPB and 42 in case of PSO. SASS was outperformed by both PSO and HPB, in terms of pass rate and average fitness. However, when combined with BMH, the number of passes for SASS increased by 30% and the average fitness dropped below that of PSO and HPB. This clearly shows that BMH is capable of improving the effectiveness of generic direct search algorithms for engineering applications, i.e. for applications where the actual realisation of a solution differs slightly from the theoretical one because of the accuracy of the manufacturing processes involved in producing the goods in the physical world.

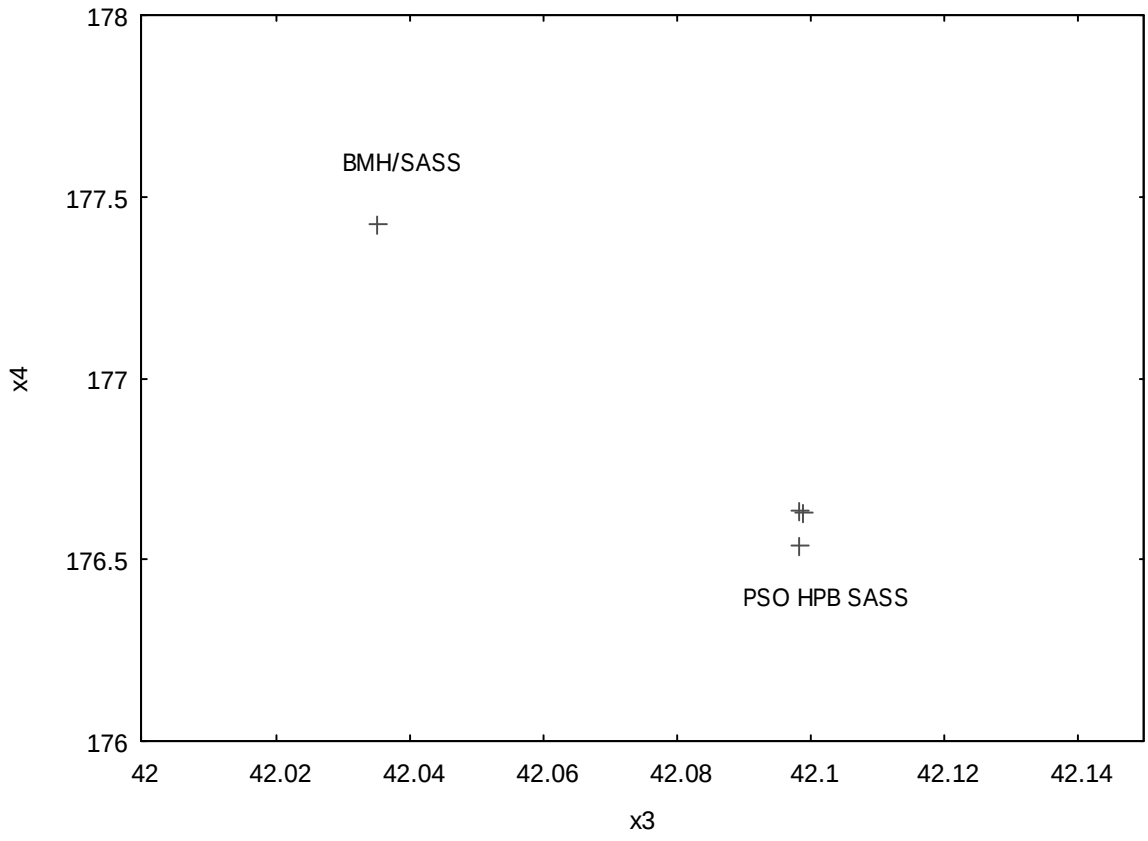


Figure 34 – Map showing how the positions have changed in relation to the other methods.

Figure 34 shows the design parameters x_3 and x_4 of the solutions found by PSO, HPB, SASS and BMH/SASS. Design parameters x_1 and x_2 are not used since they have the same values in all solutions. As can be seen, the solution found by SASS at *(42.0352,177.426)* is located in a different region of the design space, whereas the solutions found by the other methods are clustered in the right bottom corner. This shows that the meta-heuristic successfully guided SASS away from the global optimum, which is located too close to the constraint area, towards a more stable, i.e. robust solution.

5.6 Summary

In this chapter, the proposed Baldwinian-based meta-heuristic (BMH) was applied to the pressure vessel design problem. The influence of the perturbation and the number of trials was analysed and, based on the results, control parameters for the novel method were chosen. BMH was then combined with SASS and the resulting algorithm was employed to optimise the pressure vessel design. The solutions found by BMH/SASS, as well as the best solutions for PSO, HPB and SASS reported in the literature, were virtually manufactured using a Monte Carlo simulation. BMH improved the effectiveness of SASS and hence demonstrated its potential to improve generic iterative optimisation algorithms for engineering optimisation applications.

6 Conclusions and future work

This chapter revisits the research question presented in Chapter 1 and discusses the findings obtained by applying the novel Baldwinian-based meta-heuristic (Chapter 4) for generic iterative optimisation algorithms (Chapter 2) to one of the engineering case studies from Chapter 3. Following a discussion, it draws the thesis to a close by suggesting future work that could be undertaken in this area.

6.1 Conclusions

The aim of this research was to identify problems in engineering optimisation and then to develop novel solutions to the identified problems. First, principles of computational optimisation were studied and a literature review was conducted. It emerged that the latest research in the area of automated engineering design optimisation tends to combine different optimisation algorithms to improve effectiveness or efficiency. Three basic types of such hybrid configurations were identified: nested algorithms, sequential algorithms and meta-optimisers. Two problems were then identified that inexperienced practitioners encounter when trying to apply computational optimisation to real-world engineering problems. The first is the problem of parameter tuning and the second is the problem of finding robust solutions. A well-known engineering design problem, the pressure vessel problem, was selected as a case study to investigate these problems in an engineering context. The pressure vessel problem is used as a test-bed by researchers from the field of computational optimisation to evaluate new optimisation methods because it is a difficult problem to solve due to its relatively large number of constraints.

The problem of robust optimisation has two dimensions: the first is that for optimisation problems, the theoretical solutions have to be implemented in the physical world using manufacturing processes. As with everything in the physical world, these processes suffer from noise, i.e. inaccuracies that disturb the theoretical solutions. It is also well-known that for constrained problems, optimal solutions usually exist on the borders of the feasible solution

space. If the perturbation moves a solution around slightly it might enter an area of the solution space that is not allowed because one or more constraints would be violated.

The second dimension is related to the narrowness of global solutions; if a global solution is located on a very narrow peak in the multi-dimensional fitness landscape, slight deviations from that location will result in a dramatic drop in fitness. In the real-world this could have catastrophic consequences for practical engineering applications, such as designing a bridge. If a beam were designed so that it had maximum strength and minimum weight, but could not be manufactured with the required accuracy, then the strength might drop below a safety level and hence the bridge could collapse. Therefore, engineers usually incorporate safety factors into their designs (Yao, 1972), which means that they move away from optimum designs. Economically, this is not justifiable and it is in contrast to the aims of computational optimisation.

To overcome the problems mentioned above, a Baldwinian-based meta-heuristic (BMH) was proposed. As well as using the fitness of a solution, it also probes its neighbourhood in order to estimate the goodness of the region of the solution. This meta-heuristic can be combined with any arbitrary optimisation algorithm. This was demonstrated in Chapter 5, where a combination of BMH and SASS was applied to the pressure vessel design problem. SASS was chosen because it only has one control parameter to tune, which makes it most suitable to overcome the problems of parameter tuning.

It was shown that BMH/SASS was able to outperform standard SASS as well as Particle Swarm Optimisation (PSO) and Hybrid Particle Swarm Branch-and-Bound (HPB).

Another aspect of engineering optimisation is the number of decimal places. For example, a theoretical solution that relies on a design parameter to have eight figures behind the decimal point when measured in millimetres cannot be manufactured, even with modern CNC equipment. Therefore, a recommendation here is to use no more than four decimal places.

In conclusion, it can be said that the new method proposed in this work has the potential to find more robust solutions for engineering optimisation applications. However, there are still some open questions, which will be discussed in the next section.

6.2 Future work

One open question is the influence of the random number distribution used for epsilon. Here, only uniformly distributed random numbers were used. However, this distribution might not correctly reflect the distributions obtained from the manufacturing processes. Other distributions, for example a normal distribution, could lead to better results and hence should be investigated.

Finally, the combination BMH /SASS worked well for the pressure vessel design problem, but other combinations should be explored in order to prove the claim that BMH can be combined with any arbitrary direct search method.

References

Abido, M.A. (2001) Particle swarm optimization for multi-machine power system stabilizer design, in *Proceeding of Power and Energy Society Summer Meeting*, 15 Jul-19 Jul, Vancouver, Canada, pp 1346–1351.

Arnaud, R., Poirion, F. (2014) Stochastic annealing optimization of uncertain aeroelastic system, *Aerospace Science and Technology*, Vol. 39, pp 456-464.

Azad, S. K., Hasancebi, O. (2014a) An elitist self-adaptive step-size search for structural design optimization, *Applied Soft Computing*, Vol. 19, pp 226-235.

Azad, S. K., Hasancebi, O. (2014b) Elitist Self-Adaptive Step-Size Search in Optimum Sizing of Steel Structures, *International Journal of Advances in Computer Science & Its Applications*, Vol. 4, No. 4, pp 192-196.

Bach, H. (1969) On the downhill method, *Communications of the ACM*, Vol. 12, Issue 12, pp 675-677.

Bansal, J.C., Singh, P.K., Saraswat, M., Verma, A., Jadon, S.S., Abraham, A. (2011) Inertia Weight Strategies in Particle Swarm Optimization, *Third World Congress on Nature and Biologically Inspired Computing,* 19-21 Oct, Salamanca, Spain, pp 633-640.

Beyer, H.G., Sendhoff, B. (2007) Robust optimization – A comprehensive survey, *Computer Methods in Applied Mechanics and Engineering, 196* (2007) 3190–3218.

Cao, Y.J. and Wu, Q.H. (1999) A mixed variable evolutionary programming for optimization of mechanical design, *Engineering Intelligent Systems for Electrical Engineering and Communications*, Vol. 7, No. 2, pp 77-82.

Chen, X., Li, Y. (2007) A modified PSO structure resulting in high exploration ability with convergence guaranteed, *IEEE Transaction On Systems, Man, and cybernetics—part b: cybernetics*, Vol. 37, No. 5, pp 1271-1289.

Christensen, P. W., Klarbring, A. (2008) *An Introduction to Structural Optimization*, Springer.

Coello, C. A. C. (2002) Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art, *Computer Methods in Applied Mechanics and Engineering*, Vol. 191, Issue 11-12, pp1245-1287.

Coello, C.A.C., Montes, E.M. (2001) Use of dominance-based tournament selection to handle constraints in genetic algorithms, *Proc. ANNIE*, Vol.11, pp 177-182.

Congedo, P.M., Witteveen, J., Iaccarino, G. (2013) A simplex-based numerical framework for simple and efficient robust design optimization, *Computational Optimization and Applications*, Vol. 56, pp 231–251.

Costanzo, F. (2013) *Engineering Mechanics: Statics*, 2nd ed., McGraw-Hill Companies.

Deb, K. (2000) An efficient constraint handling method for genetic algorithms, *Computer Methods in Applied Mechanics and Engineering,* Vol. 186, Issue 2-4, pp 311-338.

Dias Junior, A., da Silva Junior, D.C. (2013) Using guiding heuristics to improve the dynamic checking of temporal properties in data dominated high-level designs, *Proceedings of 2013 IEEE Computer Society Annual Symposium on VLSI*, August 5-7, Natal, Brazil, pp 20-25.

Dimopoulos, G.G. (2007) Mixed-variable engineering optimization based on evolutionary and social metaphors, *Computer Methods in Applied Mechanics and Engineering*, Vol. 196, No. 4-6, pp 803-817.

Dorigo, M., Gambardella, L. (1997) Ant Colony System: A Cooperative Learning Approach to the Travelling Salesman Problem, *IEEE Transactions on Evolutionary Computation*, Vol. 1, No. 1, pp 53-66.

Du, X., Wang, Y., Chen, W. (2000) Methods for robust multidisciplinary design, *Tech. Rep. 2000-1785*, American Institute of Aeronautics and Astronautics, AIAA, 2000.

El-Mihoub, T. A, Hopgood, A. A., Nolle, L., Battersby, A. (2006) Hybrid Genetic Algorithms: A Review, *Engineering Letters*, Vol. 13, No. 2, pp 124-137.

Golub, B., Holmer, M, McKendall, R., Pohlman, Zenios, S.A. (1994) Stochastic Programming Models for Money Management, *European Journal of Operational Research*, Vol. 85, Issue 2, pp 282–296.

Guo, W.A., Li, W.Z., Zhang, Q., Wang, L., Wu, Q.D., Ren, H.L. (2015) Biogeography-based particle swarm optimization with fuzzy elitism and its applications to constrained engineering problems, *Engineering Optimization*, Vol. 46, No. 11, pp. 1465–1484.

He, S.; Prempain, E., Wu, Q.H. (2004) An improved particle swarm optimiser for mechanical design optimization problems, *Engineering Optimization*, Vol. 36, No. 5, pp 585-605.

Holland, J.H. (1975) *Adaptation in Natural and Artificial Systems*, University of Michigan Press.

Hopgood, A.A. (2001) Intelligent Systems for Engineers and Scientists. 2nd ed., CRC Press.

Jamshidi, R., Ghomi, S.M.T.F., Karimi, B. (2015) Flexible supply chain optimization with controllable lead time and shipping option, *Applied Soft Computing*, Vol. 30, pp. 26–35.

Jayaprakasam, S., Rahim, S.K.A., Leow, C.Y. (2015) PSOGSA-Explore: A new hybrid metaheuristic approach forbeampattern optimization in collaborative beamforming, *Applied Soft Computing*, Vol. 30, pp. 229-237.

Jeet, V., Kutanoglu, E. (2007) Lagrangian relaxation guided problem space search heuristics for geralized assignment problems, *European Journal of Operations Research*, Vol. 182, No. 3, pp. 1039-1056.

Jordehi, A.R., Jasni, J. (2015) Particle swarm optimisation for discrete optimization problems: a review, *Artificial Intelligence* Review, Vol. 43, Issue 2, pp 243-258.

Kanagaraj, G., Ponnambalam, S. G., Jawahar, N., Nilakantan, Mukund, J. (2015) An effective hybrid cuckoo search and genetic algorithm for constrained engineering design optimization, *Engineering Optimization*, Vol. 46, No. 10, pp 1331–1351.

Kennedy, J., Eberhart, R. (1995) Particle swarm optimization, Proceedings of the IEEE International Conference on Neural Networks, Nov. 27- Dec. 01, Perth, Australia, pp 1942-1948.

Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P. (1984) Optimization by Simulated Annealing: Quantitative Study, *Journal of Statistical Physics*, Vol.34, 1984, pp 975-986.

Kitayama, S., Arakawa, M., Yamazaki, K. (2006) Penalty function approach for the mixed discrete nonlinear problems by particle swarm optimization, *Structural and Multidisciplinary Optimization,* Vol. 32, No. 3, pp 191-202.

Liao, T., Socha, K., Montes de Oca, M. A., Stuetzle, T., Dorigo, M. (2014) Ant Colony Optimization for Mixed-Variable Optimization Problems, *IEEE Transactions on Evolutionary Computation*, Vol. 18, No. 4, pp 503-518.

Lee, K.H., Park, G.J. (2001) Robust optimization considering tolerances of design variables, *Computers and Structures 79* (2001) 77–86

Li, F., Sun, G., Huang, X., Rong J., Li, Q. (2015) Multiobjective robust optimization for crashworthiness design of foam filled thin-walled structures with random and interval uncertainties, *Engineering Structures*, Vol. 88, pp. 111-124.

Li, X., Zhang, G. (2013) Minimum penalty for constrained evolutionary optimization, *Computational Optimization and Applications*, Vol. 60, Issue 2, pp 513-544.

Li, Z., Li, Ze., Nguyen, T.T., Chen, S. (2015) Orthogonal chemical reaction optimization algorithm for global numerical optimization problems, *Expert Systems with Applications*, Vol. 42, pp. 3242-3252.

Liao, C.J., Tseng, C.T., Luarn, P. (2007) A discrete version of particle swarm optimization for flowshop scheduling problems, Computers & Operations Research, Vol. 34, No. 10, pp 3099–3111.

Lopez, R. H., Ritto, T. G., Sampaio, R., Souza de Cursi, J. E. (2014) A new algorithm for the robust optimization of rotor-bearing systems, *Engineering Optimization*, Vol. 46, No. 8, pp 1123-1138.

Mahia, M., Baykan, Ö.K., Kodaz H. (2015) A new hybrid method based on Particle Swarm Optimization, Ant Colony Optimization and 3-Opt algorithms for Traveling Salesman Problem, *Applied Soft Computing*, Vol. 30, pp. 484-490.

Martínez-Soto, R., Castillo, O., Aguilar, L.T., Rodriguez, A. (2015) A hybrid optimization method with PSO and GA to automatically design Type-1 and Type-2 fuzzy logic controllers*, International Journal of Machine Learning and Cybernetics,*Vol. 6, pp 175-196.

Murty, K. G. (1983) *Linear programming*, John Wiley & Sons.

Nelder, J. A., Mead, R. (1965) A Simplex-Method for Function Minimization, *Computer Journal*, Vol. 7, Issue 4, pp 308-313.

Nema, S., Goulermas, J., Sparrow, G., Cook, P. (2008) A Hybrid Particle Swarm Branch-and-Bound (HPB) Optimizer for Mixed Discrete Nonlinear Programming, *IEEE Transactions on System, Man and Cybernetics – Part A*, Vol. 38, No. 6, pp 1411-1424.

Nolle, L. (2004) On the effect of step with selection schemes on the performance of stochastic local search strategies, In: *Horton, G. networked simulations and simulated networks*, SCS, pp 149-153.

Nolle, L. (2006) On a Hill-Climbing Algorithm with Adaptive Step Size: Towards a Control Parameter-Less Black-box Optimisation Algorithm, *Advances in Soft Computing*, Vol. 38, pp 587-595.

Nolle, L. (2007) SASS applied to optimum work roll profile selection in the hot rolling of wide steel, *Knowledge-Based Systems*, Vol. 20, Issue 2, pp 203-208.

Nolle, L. (2008) On a novel ACO-estimator and its application to the target motion analysis problem, *Knowledge-Based Systems*, Vol. 21, No. 3, pp 225–231.

Nolle, L., Armstrong A., Hopgood, A., Ware, A. (1999) Optimum work roll profile selection in the hot rolling of wide steel strip using computational intelligence, *Lecture Notes in Computer Science*, Vol. 1625 , pp 435–452.

Nolle, L., Goodyear, A., Hopgood, A.A., Picton, P.D., Braithwaite, S.N. (2002) Automated control of an actively compensated Langmuir probe system using simulated annealing, *Knowledge-Based Systems*, Vol. 15, Issues 5–6 , pp 349–354.

Nolle, L., Bland, J.A. (2012) Self-adaptive stepsize search for automatic optimal design, *Knowledge-Based Systems*, Vol. 29, pp. 75–82.

Matousek, R., Nolle, L. (2007) GAHC: Improved GA with HC mutation, in: *Proceedings of World Congress on Engineering and Computer Science* WCECS 2007, San Francisco, USA, pp. 24–26.

Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A., Teller, E. (1953) Equation of State Calculations by Fast Computing Machines, *Journal of Chemical Physics*. Vol. 21, pp 1087-1092.

Mulvey, J., Vanderbei, R.J. (1994) Robust Optimization of Large-Scale Systems, *Operations Research*, Vol 43, No 2, 1995.

OED (2015) *Oxford English Dictionary*, Oxford University Press.

Pappas, M., Amba-Rao, C. L. (1971) A Direct Search Algorithm for Automated Optimum Structural Design, *The American Institute of Aeronautics and Astronautics Journal*, Vol. 9, No. 3, pp. 387-393.

Phadke, M.S. (1989) Quality engineering using robust design, *Englewood Cliffs: Prentice Hall*; 1989. p 97–229.

Pholdee, N., Park, W., Kim, D.K., Im, Y., Bureerat, S., Kwon, H., Chun, M. (2015) Efficient hybrid evolutionary algorithm for optimization of a strip coiling process, *Engineering Optimization*, Vol. 47, Issue 4, pp. 521-532.

Pullarcot, S. (2002) *Practical Guide to Pressure Vessel Manufacturing*, CRC Press.

Rao, S. S. (2009) *Engineering Optimization, Theory and Practice*, 4th ed., Wiley & Sons.

Rechenberg, I. (1973) *Evolutionsstrategie – Optimierung technischer Systeme nach Prinzipien derbiologischen Evolution*, Frommann-Holzboog.

Rubinstein, R. Y. (1981) *Simulation and the Monte Carlo Method*, John Wiley & Sons, New York.

Runarsson, T. P., Yao, X. (2000) Stochastic ranking for constrained evolutionary optimization, *IEEE Transactions on Evolutionary Computation*, Vol. 4, Issue: 3, pp 284-294.

Salimi, H. (2015) Stochastic Fractal Search: A powerful metaheuristic algorithm, *Knowledge-Based Systems*, Vol. 75, pp 1–18.

Sandgren, S. (1990) Nonlinear integer and discrete programming in mechanical design optimization, *Journal of Mechanical Design*, Vol. 112, pp 223-229.

Sayol, J., Nolle, L., Schaefer, G., Nakashima, T. (2008) Comparison of simulated annealing and SASS for parameter estimation of biochemical networks, *Proceedings of IEEE World Congress on Computational Intelligence*, 1-6 June, Hong Kong, China, pp 3568-3571.

Schwefel, H.-P. (1995) *Evolution and Optimum Seeking,* Wiley & Sons.

Shanley, F. R. (1949) Principles of Structural Design for Minimum Weight, *Journal of the Aeronautical Sciences*, Vol. 16, No. 3, pp. 133-149.

Standards Australia (1995) *Steel plates for pressure equipment*, AS 1548:1995.

Storn, R., Price, K. (1997) Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces, *Journal of Global Optimization*, Vol. 11, pp 341–359.

Sundaresan, S., Ishii, K., Houser, D.R. (1995) A robust optimization procedure with variations on design variables and constraints, *Engineering Optimization* 1995;24(2):101–17

Suri, R., Otto, K. (2001) Manufacturing system robustness through integrated modeling, *Journal of Mechanical Design* 123 (4) (2001) 630–636.

Templeman, A. B. (1970) Structural design or minimum cost using the method of geometric programming, *ICE Proceedings*, Vol. 46, Is. 4, pages 459 –472.

Wang, J., Yuan, W., Cheng, D. (2015) Hybrid genetic–particle swarm algorithm: An efficient method for fast optimization of atomic clusters, *Computational and Theoretical Chemistry*, Vol. 1059, pp. 12–17.

Wu, W.C., Tsai, M.S. (2008) Feeder reconfiguration using binary coding particle swarm optimization, *International Journal of Control, Automation and Systems* 6(4):488–494

Xu, R., Venayagamoorthy, G.K., Wunsch, D.C. (2007) Modeling of gene regulatory networks with hybrid differential evolution and particle swarm optimization, *Neural Networks*, Vol. 20, No.8, pp. 917-92.

Yang, H. Z. , Zhu, Y., Lu, Q. J., Zhang, J. (2015) Dynamic reliability based design optimization of the tripod sub-structure of offshore wind turbines, *Renewable Energy*, Vol. 78, pp 16-25.

Yao, J. T. P. (1972) Concept of Structural Control, *Journal of the Structural Division*, Vol. 98, No. 7, July 1972, pp. 1567-1574

Yin, S.A., Lu, C.N. (2009) Distribution feeder scheduling considering variable load profile and outage costs, *IEEE Transactions on Power Systems* , Vol. 24, No. 2, pp 652–660.

Yoshida, H., Kawata, K., Fukuyama, Y., Nakanishi, Y. (1999) A particle swarm optimization for reactive power and voltage control considering voltage stability, in *Proceeding of the International Conference on Intelligent Systems and Applied Power Systems*, Rio de Janeiro, Brazil, pp. 117–121.

Zhou, Y., Zhou, G., Zhang, J. (2015) A hybrid glow worm swarm optimization algorithm to solve constrained multimodal functions optimization, *Optimization: A Journal of Mathematical Programming and Operations Research*, Vol. 64, Issue 4, pp 1057-1080.

Appendix – Published paper

Krause, R., Nolle, L., Cant, R. J. (2015) Self-adaptive stepsize search applied to robust engineering design optimisation, to appear in: *Proceedings of the 29th European Conference on Modelling and Simulation*, 26th-29th May, Albena, Bulgaria.

SELF-ADAPTIVE STEPSIZE SEARCH APPLIED TO ROBUST ENGINEERING DESIGN OPTIMISATION

Ralph Krause
Siemens AG
Energy Management
EM MS S SO PR
Mozartstrasse 31c
91052 Erlangen, Germany
Email: ralph.krause@siemens.com

Lars Nolle
Jade University of Applied Science
Department of Engineering Science
WE Applied Computer Science
Friedrich-Paffrath-Straße 101
26389 Wilhelmshaven, Germany
Email: lars.nolle@jade-hs.de

Richard J. Cant
Nottingham Trent University
School of Science and Technology
Clifton Lane
Nottingham, NG11 8NS, UK
Email: richard.cant@ntu.ac.uk

KEYWORDS
Robust engineering design optimisation, Pressure vessel problem, Self-Adaptive Stepsize Search

ABSTRACT

In engineering it is usually necessary to design systems as cheap as possible whilst ensuring that certain constraints are satisfied. Computational optimisation methods can help to find optimal designs. However, it is demonstrated in this work that an optimal design is often not robust against variations caused by the manufacturing process, which would result in unsatisfactory product quality. In order to avoid this, a novel meta-method is introduced, which can guide arbitrary optimisation algorithms towards more robust solutions. This was demonstrated on a standard benchmark problem, the pressure vessel design problem, for which a robust design was found using the proposed method together with self-adaptive stepsize search, an optimisation algorithm with only one control parameter to tune. The drop-out rate of a simulated manufacturing process was reduced by 30% whilst maintaining near-minimal production costs, demonstrating the potential of the proposed method.

INTRODUCTION

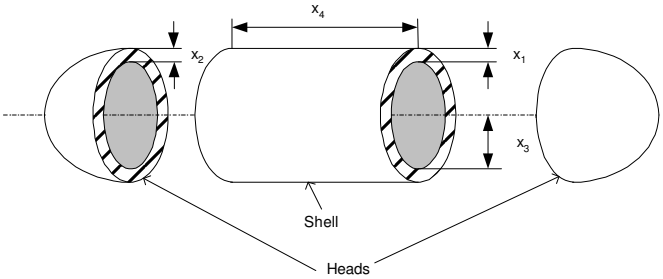
In engineering, designing a system that satisfies all the user requirements, without violating problem specific constraints, usually requires determining the system's design variable values in such a way that the resulting design is optimal in turns of a cost function. Usually, the design space is too vast to evaluate every possible design and hence only a subset of the design space can be considered. Here, a computer-based approach can be of great benefit to the engineering design process.

For this, a large number of computational optimisation algorithms are readily available, for example Genetic Algorithms (Holland 1975; Goldberg 1989), Simulated Annealing (Kirkpatrick 1984; Nolle et al 2001), Particle Swarm Optimisation (Kennedy and Eberhart 1995) or Self-Adaptive Stepsize Search (Nolle 2006; Kazemzadeh Azad and Hasanc 2014), to generate optimal designs automatically (Nolle and Bland 2012).

An interesting standard benchmark problem from the field of mechanical engineering, which was introduced by Sandgren (1990), is the pressure vessel problem. It was used as a standard problem by many researchers. Although the problem only consists of four design variables, it is not a trivial one, as it involves the optimisation of both discrete and continuous design variables under a relatively large number of constraints. The problem is used as a case study here and is therefore explained in more detail in the next section.

Pressure vessel problem

The pressure vessel consists of a simple cylinder (shell) with each end capped by a hemi-spherical shell (head), as illustrated in Figure 1. There are four design variables, which can be chosen by the engineer: the thickness x1 of the shell, the thickness x2 of the heads, the inner radius x3 and the length x4 of the shell.



Figures 1: Pressure Vessel Design Problem

The variables x1 and x2 are both discrete and vary in multiples of 0.0625 inches (please note: Sandgren used Imperial units), and variables x3 and x4 are both continuous (see Table 1).

Table 1: Design parameter types and ranges for the pressure vessel problem

Design variable	Definition	Variable Type	Thickness Increments
x1	Thickness of cylinder	discrete	0.0625 inch
x2	Thickness of sphere	discrete	0.0625 inch
x3	Inner radius of cylinder	continuous	N/A
x4	Length of cylinder	continuous	N/A

The design has to satisfy four constraint functions and an allowed range for each design variable. The design and constraint functions and the ranges of the design variables are given in the following equations and inequalities:

$$\begin{aligned} g_1(\mathbf{x}) &= 0.0193x_3 - x_1 \le 0 \\ g_2(\mathbf{x}) &= 0.00954x_3 - x_2 \le 0 \\ g_3(\mathbf{x}) &= 1{,}296.000 - \pi x_3^2 x_4 - 4/3\pi x_3^3 \le 0 \\ g_4(\mathbf{x}) &= x_4 - 240 \le 0 \\ 0.0625 &\le x_1 \le 6.1875 \\ 0.0625 &\le x_2 \le 6.1875 \\ 10 &\le x_3 \le 200 \\ 10 &\le x_4 \le 200 \end{aligned} \quad (1)$$

The aim is to minimise the total cost of the materials used and also the production costs, which consist of the costs of forming and welding the pressure vessel. Equation (2) provides the fitness function f(x), which was introduced by Sandgren (1990). It comprises the four design variables x = (x1 , x2 , x3 , x4) .

$$f(\mathbf{x}) = 0.6224\, x_1 x_3 x_4 + 1.7781\, x_2 x_3^2 + 3.1661\, x_1^2 x_4 + 19.84\, x_1^2 x_3 \quad (2)$$

Previous methods used are, amongst many others, Branch and Bound (BB) (Sandgren, 1990), Evolutionary Programming (EP) (Cao and Wu, 1999), Genetic Algorithm (Coello and Montes, 2001), Particle Swarm Optmisation (PSO) (He and Prempain, 2004), Hybrid Particle Swarm Branch-and-Bound (HPB) (Nema at al., 2008) and Self-Adaptive Stepsize Search (SASS) (Nolle and Bland, 2012). Table 2 shows the best designs, as found by the three best methods, and the associated fitness values as reported by Nolle and Bland (2012). As can be seen, the fitness (costs) could be reduced dramatically, compared to Sandgren's original value of 7982.5.

Table 2: Comparison of results

	PSO	HPB	SASS
Fitness	6,059.7143	6,059.6545	6,059.7143
x1	0.8125	0.8125	0.8125
x2	0.4375	0.4375	0.4375
x3	42.09845	42.09893	42.09845
x4	176.6366	176.6305	176.5366

This example clearly showed that computational optimisation methods are capable of finding very good solutions for engineering applications in terms of fitness function values. However, when implementing optimal designs physically, engineers are often confronted with a number of problems, which are outlined below.

Problems with optimal solutions

As can be seen from the example above, computational optimisation methods are capable of finding very good solutions for engineering applications in terms of fitness function values. However, solutions presented in the literature often use design parameter values that have many decimal places. This means, in reality, that those solutions cannot actually be manufactured because of the tolerances of both the materials involved and the manufacturing processes.

For the pressure vessel problem, for example, the solutions presented in the literature show up to five places after the decimal point. Since the manufacturing process cannot achieve the required accuracy, the parameter values are slightly different and hence result in a different fitness value. For example, using a modern CNC machine, tolerances of +/- 0.005 mm are common. Also, controlling the dimensions of the heads is especially difficult (Pullarcot 200). Other unavoidable causes of error are the thickness tolerances for plate rolled on a plate mill (Standards Australia, 1995). Table 3 shows the combined upper and lower tolerances for the pressure vessel design problem.

Table 3: Upper and lower tolerances for pressure vessel

Tolerance [inches]	x_1	x_2	x_3	x_4
Lower limit	-0.0118	-0.0118	-0.1969	-0.2756
Upper limit	0.0472	0.0472	0.1969	0.2756

The upper and lower tolerance values were added to the design solutions presented in Table 2. Table 4 shows how the fitness values changed when the upper and lower tolerances were taken into account.

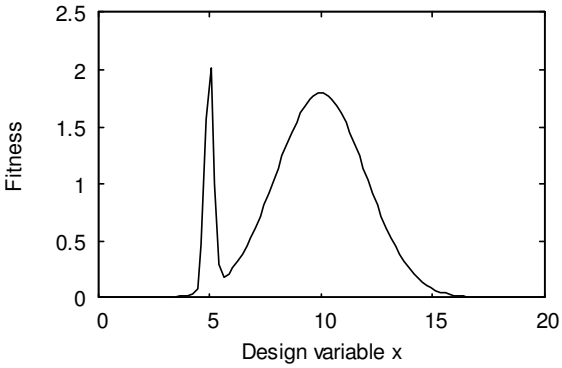
Table 4: Fitness after applying upper and lower tolerances to the original design solutions

Method	Original fitness	Fitness for lower tolerance values	Fitness for upper tolerance values
PSO	6,059.7143	N/A (g1 and g3 violated)	6579.68
HPB	6,059.6545	N/A (g1 and g3 violated)	6579.60
SASS	6,059.7143	N/A (g1 and g3 violated)	6579.678

As can be seen from Table 4, the fitness decreases dramatically for all of the designs using the upper tolerance values, with an error of 8.6%. Even worse, when using the lower tolerances, all of the designs are unfeasible because they violate constraints g_1 and g_3 and hence would not be fit for purpose!

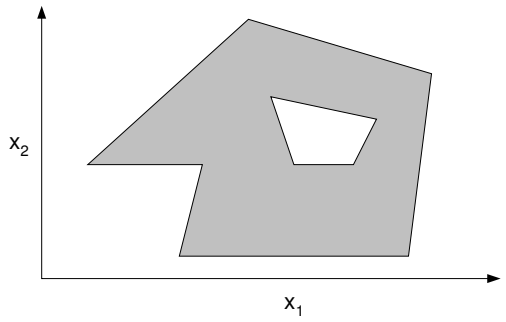
There are two aspects of robust optimisation. The first is related to the narrowness of the global optimum for unconstraint optimisation. Figure 2 shows an example of a fitness landscape for a one-dimensional optimisation problem. In the example, there are two maxima, one at x=5 and the other at x=10. The global optimum is located at x=5. However, the peak is too narrow, and slight deviations in implementing the

solution will cause a significant drop of the fitness value. The local optimum at x=10 does not offer the same fitness, but slight deviations in x do not cause a dramatic drop in fitness when implementing the solution. If the fitness were acceptable, this would be the preferred solution for an engineering application. For a practical application, it would be more desirable to find a robust solution rather than the global optimum.



Figures 2: Example of a fitness landscape for a one-dimensional optimisation problem

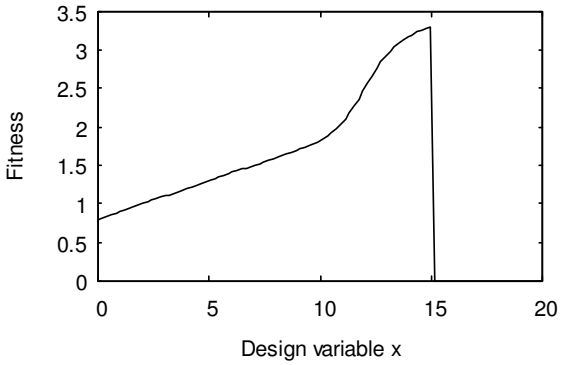
The other problem relates to constraint optimisation. Here, constraints define areas in the input space which do not contain feasible solutions. Figure 3 shows an example of an input space for a two-dimensional constraint optimisation problem.



Figures 3: Example of an input space for a two-dimensional constraint optimisation problem

In the example, there are two design variables, x1 and x2. However, only combinations of x1 and x2 that lie in the grey area are feasible, i.e. allowed, whereas combinations that lie in the white areas are not feasible. The white areas are defined by the constraints. The more constraints there are, the more complicated the shape of the allowed area(s) can be and hence the more difficult the optimisation problem.

It is a well-known fact that optimum solutions are normally located at the edges of the feasible space; for example, this forms the basis for the simplex algorithm for linear programming (Murty, 1983). Figure 4 shows an example of a fitness landscape for a one-dimensional constrained optimisation problem.



Figures 4: Example of a fitness landscape for a one-dimensional constrained optimisation problem

It can be seen that the global optimum is located at x=15. However, all of the points for values x>15 lie outside of the area of feasible solutions and hence, due to penalisation, result in a fitness value of zero. If an algorithm finds the global optimum correctly, but due to the engineering related problems mentioned above the implementation of the solution uses $x+\varepsilon$, all solutions with a positive ε will cause a constraint violation, i.e. lead to infeasible solutions.

Clearly, it would be of benefit if an algorithm could find a solution close to the global optimum but with a safety distance $d > \varepsilon$. In this case, if the realisation process causes a deviation up to ε, the resulting solution would a) still be close to the global optimum, and b) not violate any constraint.

Both problems mentioned above, i.e. narrow global optima and constraint optimisation using penalty functions, are equivalent, and hence it is possible to address both issues using a single method.

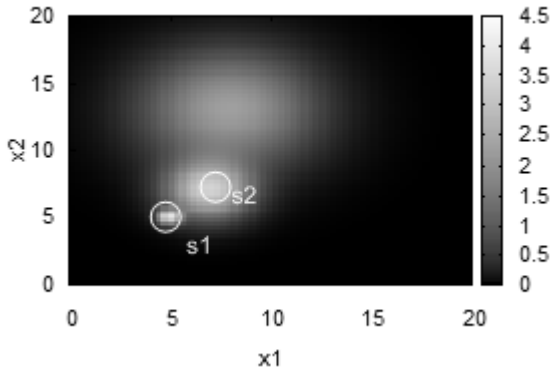
This analysis led to the design of a meta-method, which can be used in conjunction with any iterative optimisation algorithm. This method is presented in the next sections.

META-METHOD FOR ROBUST OPTIMISATION

Typical engineering optimisation problems deal with high-dimensional and multimodal design spaces. The goal of engineering design should be to find a good enough solution that is a) near-optimal in terms of its fitness criterion, and b) robust enough to maintain that fitness even if the manufacturing process causes slight deviations in the actual design parameter values.

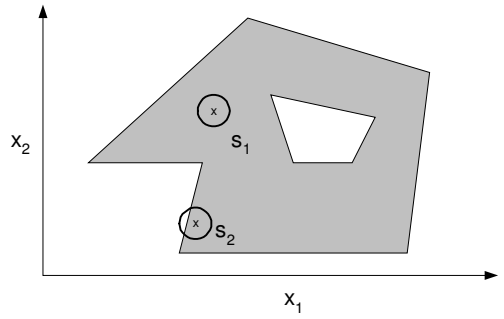
Computational optimisation methods have been proven to find near-optimal solutions in finite time and with a high degree of repeatability. These methods should be modified so as to learn from experience, i.e. guided away from dangerous areas during the search. However, none of the common standard computational optimisation algorithms has a facility to learn during a single iteration. The closest facility would be local search, a technique often used to improve the current solution by searching the neighbourhood of the current solution. If a better solution is found, the solution itself is changed (Lamarckian learning). Alternatively, for

genetic algorithms, the fitness value of the original current solution can be improved, which increases the individual's chance of being selected to generate offspring for the next generation (Baldwin learning) (El-Mihoub et al 2006).
Figure 5 shows as an example a contour plot of a fitness landscape for a two-dimensional unconstrained optimisation problem.



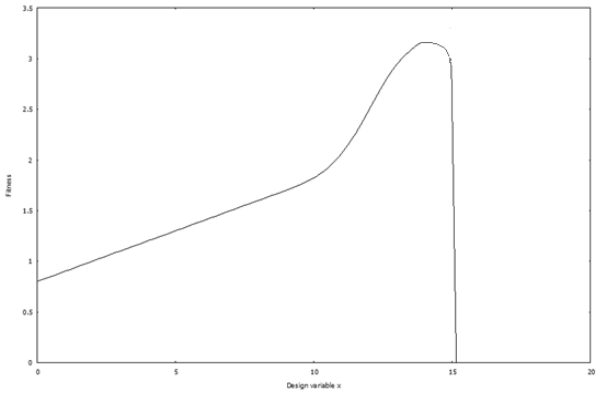
Figures 5: Contour plot of a multimodal fitness landscape

As can be seen, most deviations from s1 will cause the solution to drop dramatically in fitness whereas solution s2 is more robust in that respect. A local search here would not be of any help since the global optimum (s1) has already been found. However, if, through local probing, a measure of the deviation in fitness for the region around s1 could be established, it could be used to adjust the fitness accordingly. This would be similar to the Baldwin approach for genetic algorithms but could be used with any arbitrary optimisation algorithm.
Similarly, for constraint optimisation using penalty functions, this measure could be used to penalise solutions that are too close to the border with the forbidden areas. Figure 6 shows two solutions s1 and s2 for a two-dimensional constrained optimisation problem. The grey area contains all feasible solutions; white areas are forbidden.



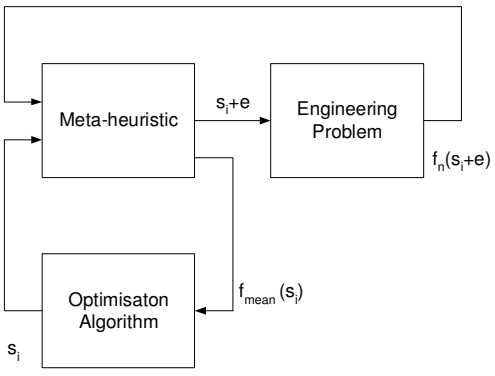
Figures 6: Two different solutions in a constrained search space

It can be seen that probing the areas with the radius ε around the solutions would lead to a penalisation for s2, because a section of the circle is in the forbidden zone. On the other hand, s1 would not be subject to such a penalty, because none of the points within the neighbourhood <=ε would cause a violation of a constraint.
As can be seen above, both problems can be solved by using a measure of the quality of the solutions contained in the neighbourhood of a solution. For unconstrained optimisation, this penalty would change the shape of the effective fitness landscape so that narrow peaks are penalised. For constrained optimisation, it would change the shape of the fitness landscape so that points near to or on the border of a forbidden region would be lowered in fitness or, in other words, the edges would be 'rounded off' (see Figure 4) and produce a fitness barrier to protect the solutions from leaving the feasible space when realised through an engineering process (Figure 7).



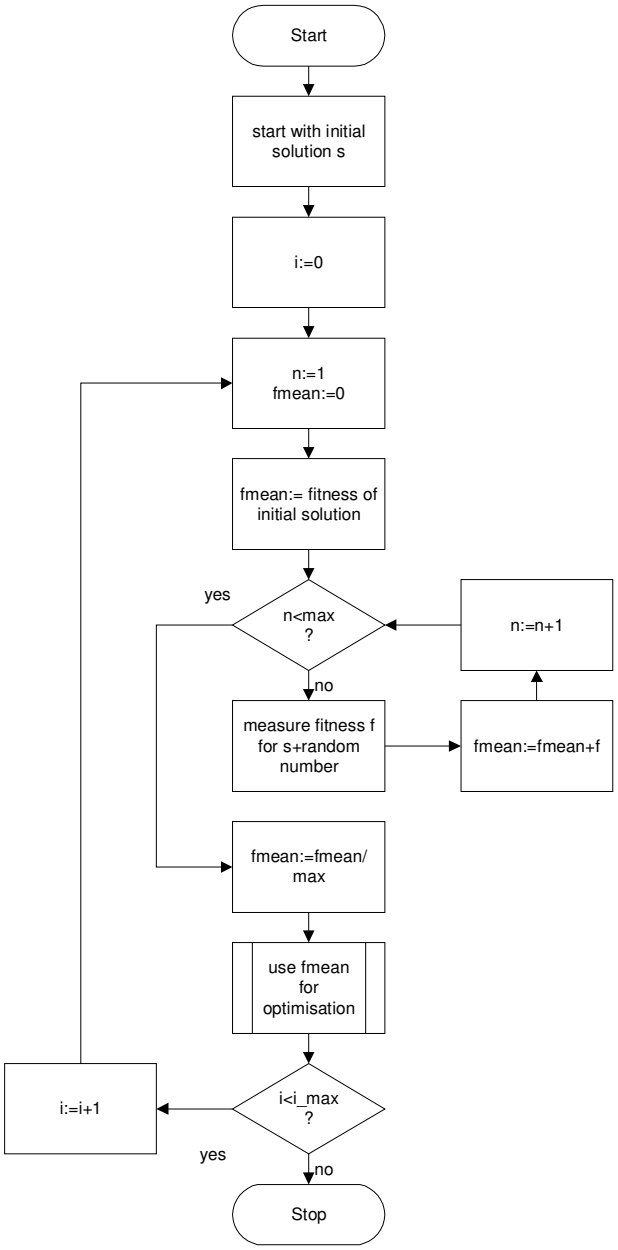
Figures 7: Fitness barrier to protect solutions from dropping into the forbidden area

This led to the development of the following meta-heuristic for engineering applications (Figure 9). The search begins with an initial solution, respectively with an initial population of solutions. Each time a solution is due to be evaluated by the optimisation algorithm used, its fitness is evaluated first and then the neighbourhood with the radius ε is sampled with random trials. The average fitness of all of the trials is calculated and used by the host optimisation algorithm instead of the fitness for the original solution. Figure 8 shows the optimisation loop for the Baldwinean-based meta-heuristic.



Figures 8: Optimisation loop for meta-heuristic

It can be seen that the optimisation algorithm, which could be of any type, does not receive a quality measure directly from the engineering problem. Instead, it sends its solution to the meta-heuristic, which in turn presents n slightly different versions of the solution to the problem and calculates the average fitness, including any penalties if applicable. This average fitness is then used by the optimisation algorithm for decision making.



Figures 9: Meta-heuristic for robust engineering

EXPERIMENTS

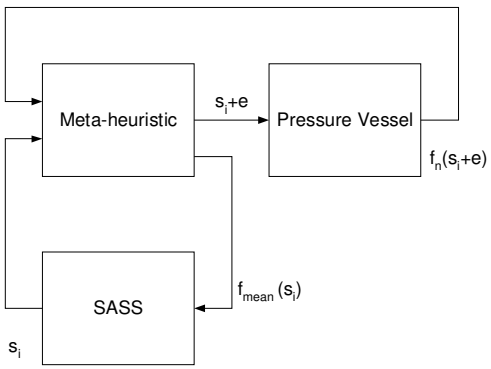
The meta-heuristic described above can be combined with any direct search algorithm, i.e. any algorithm, that uses an optimisation loop to adjust current solutions using the fitness currently observed.

For the experiments, self-adaptive stepsize search (SASS) (Nolle 2006) was used as the optimisation algorithm.

SASS is a population-based adaptation scheme for hill climbing with a self-adaptive step size, where the temporary neighbourhood of a particle p_i is determined by the distance between itself and a randomly selected sample particle s_i of the population in each iteration. When the search is progressing, each particle is attracted by a local optimum and hence the population is clustered around a number of optima. If both, p_i and s_i are located in different clusters, p_i has the chance to escape its local optimum if it samples from a region with a higher fitness, i.e. lower costs. Towards the end of the search, most particles have reached the region of the global optimum and hence their mean distance is much smaller than in the initial population. As a result, the maximum step size s_{max} is sufficiently small to yield the global optimum.

SASS was chosen because it has only one control parameter, which is the number of particles. This eliminates the need for fine-tuning control parameters before the algorithm can successfully applied to industrial problems (Nolle 2007; Dias Junior and da Silva Junior, 2013). Hence, the experiments would not be influenced by the meta-optimisation problem of tuning control parameters.

SASS was also proven to be effective and efficient for the pressure vessel problem (Nolle and Bland, 2012). Figure 10 shows the optimisation loop for the pressure vessel problem, using SASS as a direct search algorithm.



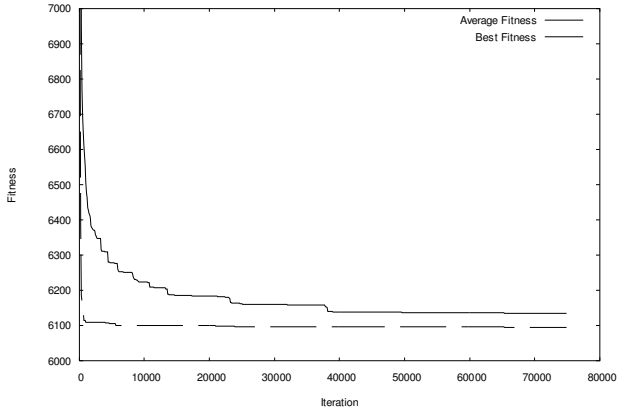
Figures 10: Optimisation loop for Baldwinean-based meta-heuristic

For SASS, the same control parameters were used as reported by Nolle and Bland (2012) to allow a fair comparison. The number of particles was hence set to 16.

Before carrying out the experiments, it was decided to limit the number of places after the decimal; for the pressure vessel problem, the design parameters x3 and x4 are continuous. For practical applications, a solution with a large number of decimal places behind the decimal point cannot be implemented, because of the accuracy of the engineering processes involved, for example +/- 0.0002 inches for modern CNC machines (Pullarcot, 200). Therefore, only four places behind the decimal point were used for the experiements.

The new method, refered to as Baldwinian-based Meta-Heuristic (BMH) has two degrees of freedom, which are the number of trials, or sample size, and the range of the samples around a solution, i.e. epsilon. These had to be determined empirically. Based on the results obtained from extensive experiments, the sample size was chosen to be 10 and epsilon was chosen to be 0.01% of the search space dimension range.

The maximum number of iterations was chosen to be 75,000 for the experiments. After 50 runs of the algorithm, an average fitness of 6065.3908 was achieved with a standard deviation of 2.5375. Figure 11 shows a conversion plot of a typical run. It can be seen that the average fitness improves up to around 65,000 iterations whereas the best fitness was already found after approximately 9,000 iterations.



Figures 11: Conversion Plot for a typical Run of BMH. The solid line depicts the average fitness and the dashed line represents the best fitness in the population.

The best solutions found during the 50 runs of the BMH/SASS algorithm are compared with the best solutions reported in the literature in the next section.

EVALUATION

In order to evaluate the solutions obtained, a Monte Carlo simulation (Rubinstein, 1981) was developed; the manufacturing process was simulated by randomly changing the optimal solutions found by different optimisation algorithms. These algorithms were: Particle Swarm Optmisation (PSO) (He and Prempain, 2004), Hybrid Particle Swarm Branch-and-Bound (HPB) (Nema at al., 2008), Self-Adaptive Stepsize Search (SASS) (Nolle and Bland, 2012) and the new Baldwinian-based meta-heuristic on top of SASS (BMH/SASS). Apart from the latter method, best solutions reported in the literature were used and rounded to four decimal places behind the decimal point. The perturbation was randomly chosen from the intervals presented in Table 3, which represent the technical limitations for pressure vessel manufacturing (Pullarcot, 2002). A uniform distribution or random numbers was used.

Table 5: Comparison of results

	PSO	HPB	SASS	BMH/SASS
Passed [%]	42	40	40	53
Average Fitness	6298.09	6298.72	6318.39	6284.24

As it can be seen from Table 5, out of the 100 pieces virtually produced, only 40 passed the quality check in the case of SASS and HPB and 42 in case of PSO. SASS was outperformed by both, PSO and HPB, in terms of pass rate and average fitness. However, when combined with BMH, the number of passes for SASS increased by 30% and the average fitness dropped below that of PSO and HPB. This clearly shows that BMH is capable of improving the effectiveness of generic direct search algorithms for engineering applications, i.e. for applications were the actual realisation of a solution differs slightly from the theoretical one because of the accuracy of the manufacturing processes involved in producing the goods in the physical world.

CONCLUSIONS

The aim of this research was to increase the robustness of engineering design solutions. Two major problems were identified; the first one is the problem of over-specifying solutions. This means, that for engineering optimisation problems, the theoretical solutions have to be implemented in the physical world using manufacturing processes. As everything in the physical world, these processes suffer from noise, i.e. inaccuracies that disturb the theoretical solutions. It is also well-known that for constrained problems, optimal solutions usually exist on the borders to the feasible solution space. If the perturbation moves a solution slightly around it might enter an area of the solution space that is not allowed because one or more constrains would be violated.

The second problem is related to the narrowness of global solutions; if a global solution is located on a very narrow peak in the multi-dimensional fitness landscape, slight deviations from that location will result in a dramatic drop of fitness. In the real-world this could have catastrophic consequences for practical engineering applications. For example, if a bridge has to be build using a beam that was designed so that it has maximum strength and minimum weight, but could not be manufactured with the required accuracy, then the

strength might drop below a safety level and hence the bridge could collapse. Therefore, engineers usually incorporate safety factors into their designs, which means that they move away from optimum designs. This is economically not justifiable and is in contrast to the aims of computational optimisation.

To overcome these problems, a Baldwinian-based meta-heuristic (BMH) was proposed. It uses not only the fitness of a solution alone, it also probes its neighbourhood in order to estimate the goodness of the region of the solution. This meta-heuristic can be combined with any arbitrary optimisation algorithm, which was demonstrated on the pressure vessel problem, where a combination of BMH and SASS was used. It was shown that BMH/SASS was able to outperform standard SASS as well as Particle Swarm Optimisation (PSO) and Hybrid Particle Swarm Branch-and-Bound (HPB).

Another aspect of engineering optimisation is the number of decimal places behind the decimal point. For example, a theoretical solution that relies on a design parameter to have eight decimal places behind the decimal point when measure in millimetres cannot be manufacture, even with modern CNC equipment. Therefore, a recommendation here is to use not more than four decimal places.

In conclusion, it can be said that the new method proposed in this work has the potential to find more robust solutions for engineering optimisation applications.

REFERENCES

Cao, Y.J. and Wu, Q.H. 1999 "A mixed variable evolutionary programming for optimization of mechanical design", *Engineering Intelligent Systems for Electrical Engineering and Communications*, Vol.7, No. 2, pp 77-82.

Coello, C.A.C. and Montes, E.M. 2001 "Use of dominance-based tournament selection to handle constraints in genetic algorithms", Proc. ANNIE, Vol.11, pp 177-182.

Dias Junior, A. and da Silva Junior, D.C. 2013. "Using Guiding Heuristics to Improve the Dynamic Checking of Temporal Properties in Data Dominated High-Level Designs". Proceedings of IEEE Computer Society Annual Symposium on VLSI, pp 20-25.

El-Mihoub, T.; Hopgood, A.A.; Nolle, L.; Battersby, A. 2006. "Hybrid Genetic Algorithms – a Review", *Engineering Letters*, Vol. 13, No. 2, pp 124-137.

Goldberg, D. E. 1989. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley.

Holland, J.H. 1975. *Adaptation in Natural and Artificial Systems*, University of Michigan Press.

He, S.; Prempain, E. and Wu, Q.H. 2004. "An improved particle swarm optimiser for mechanical design optimization problems", *Engineering Optimization*, Vol. 36, No. 5, pp 585-605.

Kazemzadeh Azad, S. and O. Hasanc¸ O. 2014. An elitist self-adaptive step-size search for structural design optimization. *Applied Soft Computing*, Vol. 19, pp 226-235.

Kennedy, J. and Eberhart, R. 1995. "Particle swarm optimization". Proceedings of IEEE International Conference on Neural Networks, Vol.4, pp 1942-1948.

Kirkpatrick, S.; Gelatt, C. D. and Vecchi M. P. 1984. "Optimization by Simulated Annealing: Quantitative Study", *Journal of Statistical Physics*, Vol.34, 1984, pp 975-986.

Murty, K. G. 1983. *Linear programming*, John Wiley & Sons.

Nema, S.; Goulermas, J.; Sparrow, G. and Cook, P. 2008. "A Hybrid Particle Swarm Branch-and-Bound (HPB) Optimizer for Mixed Discrete Nonlinear Programming". *IEEE Transactions on System, Man, And Cybernetics, Part A*, Vol. 38, No. 6, pp 1411-1424.

Nolle, L.; Goodyear, A.; Hopgood, A.A.; Picton, P.D. and Braithwaite, N.St.J. 2001. "Automated Control of an Actively Compensated Langmuir Probe System Using Simulated Annealing", in: Macintosh, A.; Moulton, M.; Preece, A. (Ed) *Applications and Innovations in Intelligent Systems*, Vol. IX, Springer, pp 115-128.

Nolle, L. 2006. "On a Hill-Climbing Algorithm with Adaptive Step Size: Towards a Control Parameter-Less Black-box Optimisation Algorithm". in: Reusch, B. (Ed) *Computational Intelligence, Theory and Applications*, *Advances in Soft Computing*, Vol. 38, Springer, 2006, pp 587-595.

Nolle, L. 2007. "SASS Applied to Optimum Work Roll Profile Selection in the Hot Rolling of Wide Steel". *Knowledge-Based Systems*, Vol. 20, Issue 2, pp 203-208.

Nolle, L. and Bland, J.A. 2012. "Self-adaptive stepsize search for automatic optimal design", *Knowledge-Based Systems*, Vol. 29, pp. 75-82.

Pullarcot, S. 2002 *Practical Guide to Pressure Vessel Manufacturing*, CRC Press.

Rubinstein, R. Y. 1981. *Simulation and the Monte Carlo Method*. John Wiley & Sons, New York.

Sandgren, S. 1990. "Nonlinear integer and discrete programming in mechanical design optimization". *Journal of Mechanical Design*, Vol. 112, pp 223-229.

Standards Australia 1995. *Steel plates for pressure equipment*, AS 1548:1995.

AUTHOR BIOGRAPHIES

RALPH KRAUSE was born in Weimar, Germany, and studied power engineering at the University of Applied Sciences in Magdeburg. He has been working with Siemens since 1998, where he held different positions in commissioning, project management, quality management, business development and is now head of department for process, tools and training governance in medium voltage and systems.

LARS NOLLE graduated from the University of Applied Science and Arts in Hanover, Germany, with a degree in Computer Science and Electronics. He obtained a PgD in Software and Systems Security and an MSc in Software Engineering from the University of Oxford as well as an MSc in Computing and a PhD in Applied Computational Intelligence from The Open University. He worked in the software industry before joining The Open University as a Research Fellow. He later became a Senior Lecturer in Computing at Nottingham Trent University and is now a Professor of Applied Computer Science at Jade

University of Applied Sciences. His main research interests are computational optimisation methods for real-world scientific and engineering applications.

RICHARD CANT started life as a theoretical physicist, then moved into industry, where he spent nine years as a system designer working on computer generated imaging for military training systems. Dr Cant is a Senior Lecturer in Computing with the Nottingham Trent University. His areas of research interest include: Intelligent simulation and modelling, Computer graphics, Artificial intelligence, Software engineering, Integrated hardware/software design and Physical simulation.