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

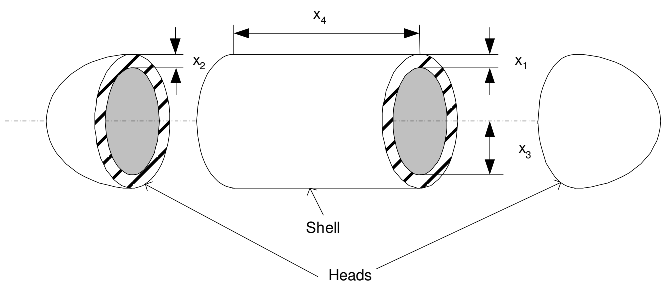
In engineering, designing a system that satisfies all the user requirements, without violating problem specific constraints, usually requires determining the system's design variable values in such a way that the resulting design is optimal in turns of a cost function. Usually, the design space is too 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:

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

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) .

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

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

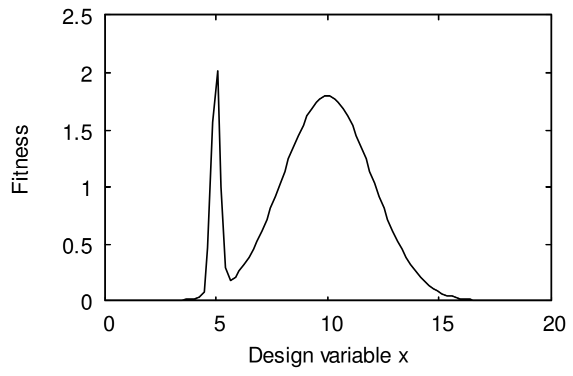
The upper and lower tolerance values were added to the design solutions presented in Table 2. Table 4 shows how the fitness values changed when the upper and lower tolerances were taken into account.

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

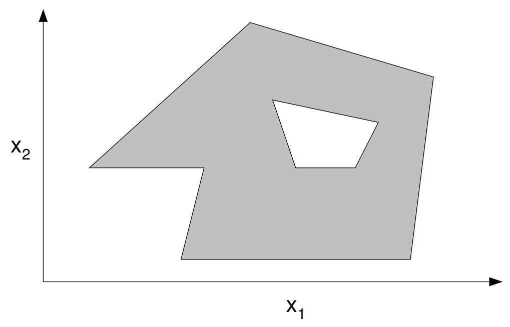
As can be seen from Table 4, the fitness decreases dramatically for all of the designs using the upper tolerance values, with an error of 8.6%. Even worse, when using the lower tolerances, all of the designs are unfeasible because they violate constraints g_1 and g_3 and hence would not be fit for purpose!

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

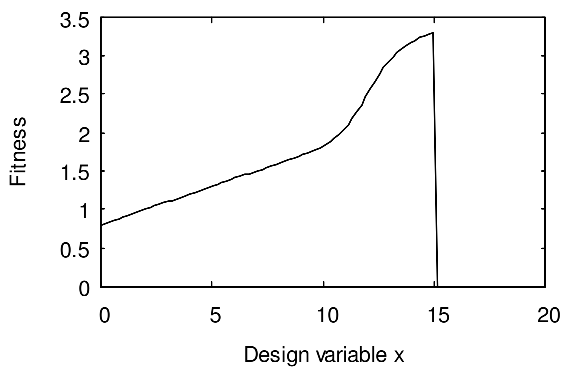
The other problem relates to constraint optimisation. Here, constraints define areas in the input space which do not contain feasible solutions. Figure 3 shows an example of an input space for a two-dimensional constraint optimisation problem.



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+ε, 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 > ε. 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.

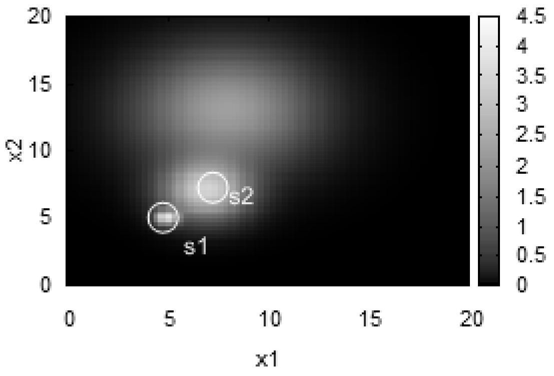
This analysis led to the design of a meta-method, which can be used in conjunction with any iterative optimisation algorithm. This method is presented in the next sections.

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.

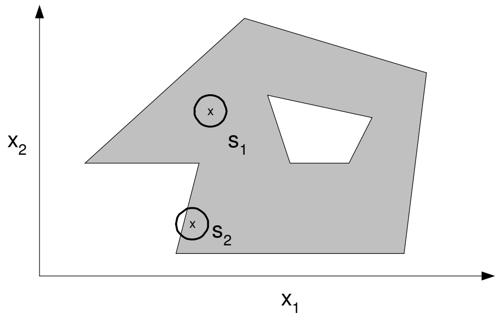
Computational optimisation methods have been proven to find near-optimal solutions in finite time and with a high degree of repeatability. These methods should be modified so as to learn from experience, i.e. guided away from dangerous areas during the search. However, none of the common standard computational 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.



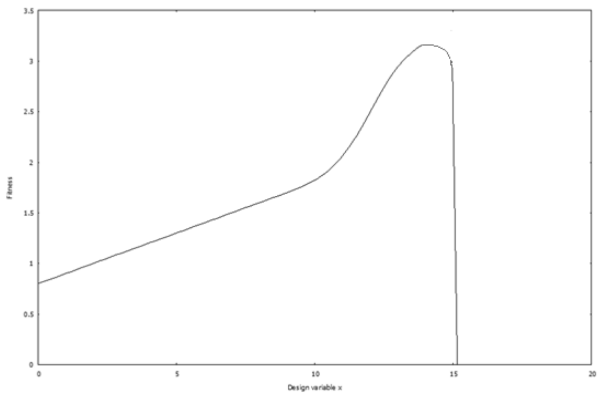
Figures 5: Contour plot of a multimodal fitness landscape

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.



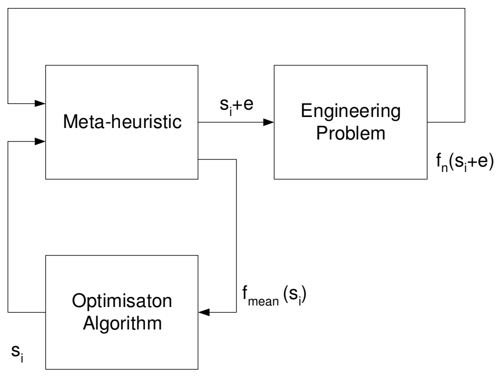
Figures 6: Two different solutions in a constrained search space

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).



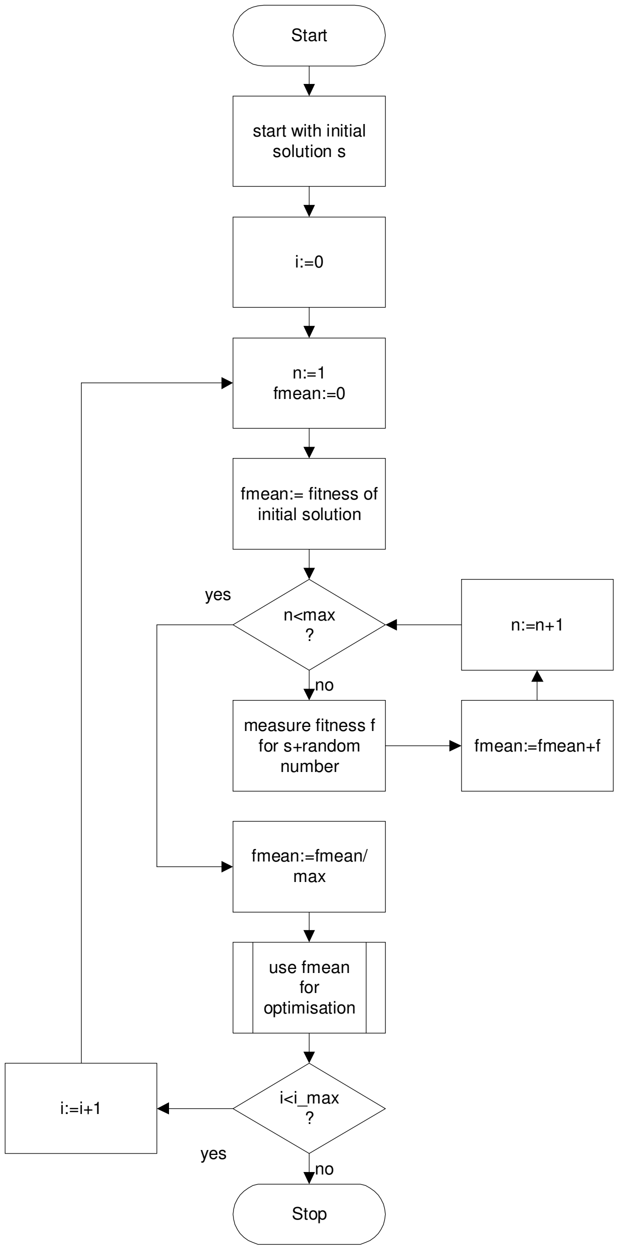
Figures 7: Fitness barrier to protect solutions from dropping into the forbidden area

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.


Start
start with initial solution s
i:=0
n:=1
fmean:=0
fmean:= fitness of initial solution
yes
n<max ?
n:=n+1
no
measure fitness f for s+random number
fmean:=fmean+f
fmean:=fmean/ max
use fmean for optimisation
i:=i+1
i<i_max ?
yes
no
Stop

Figures 9: Meta-heuristic for robust engineering

EXPERIMENTS

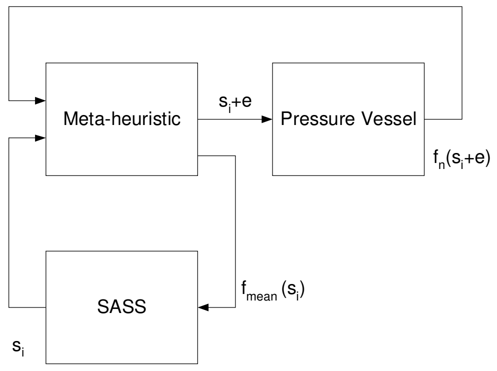
The meta-heuristic described above can be combined with any direct search algorithm, i.e. any algorithm, that uses an optimisation loop to adjust current solutions using the fitness currently observed.

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.

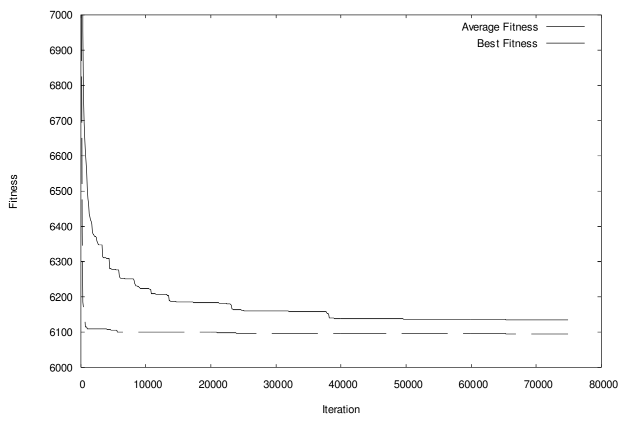
SASS was also proven to be effective and efficient for the pressure vessel problem (Nolle and Bland, 2012). Figure 10 shows the optimisation loop for the pressure vessel problem, using SASS as a direct search algorithm.



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.

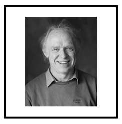
Standards Australia 1995. *Steel plates for pressure equipment*, AS 1548:1995.

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.