INTERNATIONAL CONFERENCE ON ENGINEERING DESIGN
ICED 03 STOCKHOLM, AUGUST 19-21, 2003

A MACHINE LEARNING-BASED APPROACH TO ACCELERATING COMPUTATIONAL DESIGN SYNTHESIS

Christopher A. W. Vale and Kristina Shea

Abstract

A modification-based framework for computational design synthesis augmented by machine learning is presented. The framework allows a wide range of engineering design problems to be addressed via a machine learning based search algorithm with minimal required adaptation of the search heuristics. Search is accomplished via two agents, a 'data modeller' and a 'modification advisor', that work together to guide a generate-and-test-oriented search with suggested actions based on past observation of the search. A proposed implementation of the search algorithm is discussed and the results of its application to two design examples are presented. Implications of the method for engineering design are discussed.

Keywords: Computational design synthesis, machine learning, knowledge re-use.

1. Introduction

Engineering design is increasingly being influenced by the proliferation of computer tools in the design environment. CAD systems that previously supported human efforts through design representation and visualisation are being supplemented with optimisation algorithms that automate various routine design tasks [1]. In some fields, notably electronics, systems are emerging of such complexity that computer-aided design synthesis is essential for success [2]. With advances in these areas yielding clear benefits for practitioners, computational design synthesis is finding application in more diverse areas, such as civil engineering [3] and mechanical engineering [4]. The primary aim of computational design synthesis is to achieve the most beneficial and innovative design outcomes with the lowest possible time investment. Over the years, a number of procedures have emerged as suitable candidates for solving difficult design problems but their successful application is often dependent on the nature of the design problem and its objectives. The growth of the field is therefore limited, to some extent, by the ability of developers to adapt existing heuristic search algorithms, such as simulated annealing (SA) and evolutionary strategies, such as genetic algorithms (GAs) to diverse practical problems. Of the large variety of approaches for computational design synthesis, we choose to focus on techniques that follow a bottom-up approach, forgoing the use of expert knowledge to favour the possible generation of novel, unbiased design solutions. Although techniques such as SA and evolutionary strategies support such ideals when applied in conjunction with a suitable design representation, their application to the range of practical problems often encountered in industry is not always easy and often computationally expensive. Practical scenarios, for example, often require the investigation of multiple design objectives to determine design trade-offs [5] requiring adaptations of SA- [6] and GA-based techniques [7]. A further complication arises from the nature of the design constraints and the way in which the search algorithm is able to search the design space. Standard GAs, for example, are not well suited to problems where the number of design variables change throughout the synthesis process. As a result of these practical concerns, much research that goes into computational design synthesis revolves around how the researcher managed to adapt a known heuristic method to a particular problem through development of a suitable design space model and appropriate modification of the chosen heuristic method.

This research attempts to address a wide range of practical design scenarios by defining a modification-based design framework where design solutions develop through the successive application of modification operators to the existing design. This formalism encompasses a large number of potential practical scenarios while requiring only minimal effort to put into effect. To operate within the modification-based framework, a general method for design synthesis with the aid of machine learning (ML) is presented. This is intended to guide the solution search using past experience that is self-learnt by on-going observation of the synthesis process. The search framework, summarised in Figure 1, comprises two agents. The 'data modeller' accumulates data about the synthesis problem through observation and learns relevant relationships between the design objectives, constraints and modification operators. The 'modification advisor', uses the knowledge elucidated by the data modeller to advise appropriate modifications to the current design taking into account various potential factors, including the current objective values, the recent history of modifications made and the recent progress made in the search. Design modifications are thereby favoured that are more likely to benefit synthesis. With increased experience, the data modeller can extend its knowledge base and allow the modification advisor to make a greater contribution to the search.

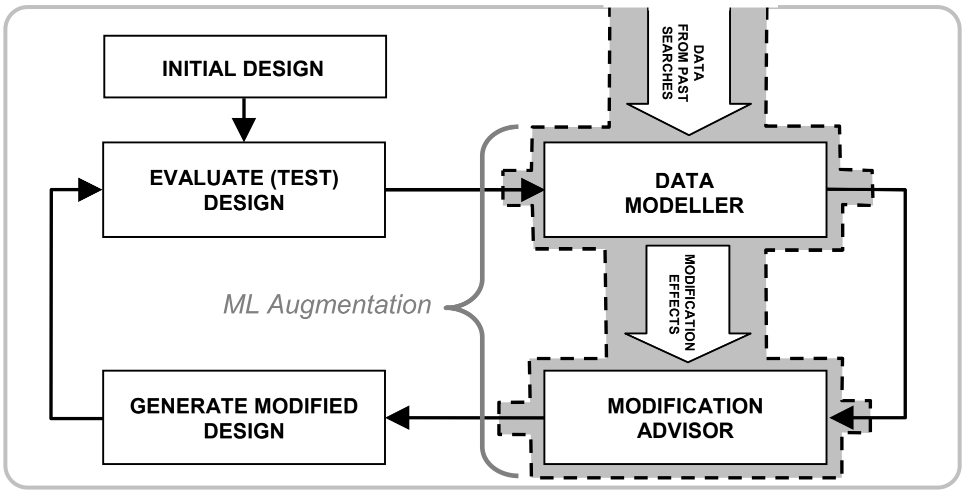


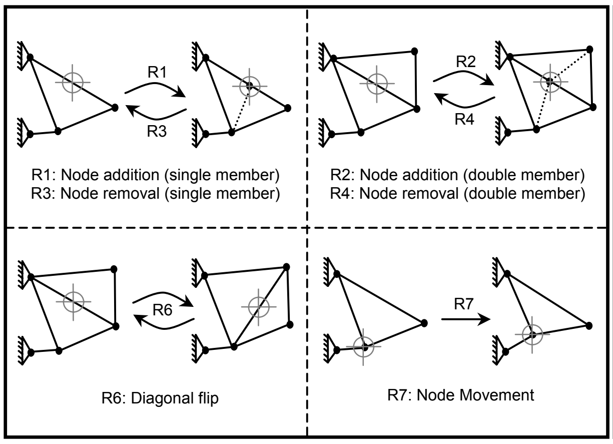
Figure 1. Iterative generate-and-test search augmented by machine learning (ML).

2. The modification-based design synthesis framework

Modification-based design describes a process by which one or more design solutions are gradually developed from an initial design through successive applications of modification operators. Typically, the design synthesis algorithm seeks to implement modifications that lead to improved designs, in terms of one or more objectives, after a number of modifications have been made. Because implementation of the modification operators and evaluation of the resulting new designs demand computational resources, it is desirable to minimise the number of iterations in the search whilst maintaining solution quality. This framework not only accounts for simple optimisation tasks but also more complex design formalisms where the modifications employed not only modify but also introduce and remove design variables. An example of some of the modification operators constituting the formalism used in the design of pin joint planar truss structures is summarised by Figure 2 [8]. A set of modification schemas is shown such that the modification operators, termed 'shape grammar rules' in this case, can be applied in succession to generate a large and diverse number of planar truss topologies. Figure 3 shows the generation of a typical loaded planar truss through repeated application of the rules. Note that with each application of the modification rules, design objectives, such as truss mass and number of nodes can change. Truss design will be revisited later as an example. It should be noted that the modification-based method here supports a bottom-up synthesis model, which does not bias synthesis by prescribing a design process, thereby allowing the generation of novel design solutions. Moving beyond parametric optimisation, synthesis is generally more naturally formulated as modification-based.

Each modification type is assigned an integer label from 1 to N_R, with N_R the total number of modification types. Interestingly, the modification operators in Figure 2 are capable of introducing (R1, R2), removing (R2, R4) and modifying design variables (R6, R7). In addition to a basic modification rule type, as shown in Figure 2, modifications may be further qualified by application variables that provide a more complete description of the applied modification. Such a parameter may be the point of application of the modification operator in Cartesian co-ordinates in the design (Figure 2). In combination, the modification type label and the application variables provide a detailed description of each rule application. However, it has been observed that is it preferable to keep the number of rule types and application variables low as a higher level of specification requires more data samples to support statistical inference through observation.

An additional consideration is the issue of *assured* versus *context sensitive* modifications. When modifications to a design are assured, it is possible to specify an entire *viable* design as an arbitrary sequence of modifications applied to any initial design. However, in many practical scenarios, constraints result in modifications being context sensitive in that the applicability of a modification at any stage of the design process depends on the state of the current design and is not always guaranteed. Looking at the example of the truss grammar, it is clear that repeated application of rule type '3' on any truss cannot continue endlessly. It is of some use to be able to predict the likelihood that a modification will be applicable at some point in the future. This may be accomplished by using prior knowledge of the modification set or by observing the history of successful and unsuccessful attempts at applying the modifications to help build a statistical model of the 'probability of application' for a particular modification under various circumstances.


R1
R3
R1: Node addition (single member)
R3: Node removal (single member)
R2
R4
R2: Node addition (double member)
R4: Node removal (double member)
R6
R6: Diagonal flip
R7
R7: Node Movement
Figure 2. The truss modification rules. (rule application locations are defined by crosshair indicators).

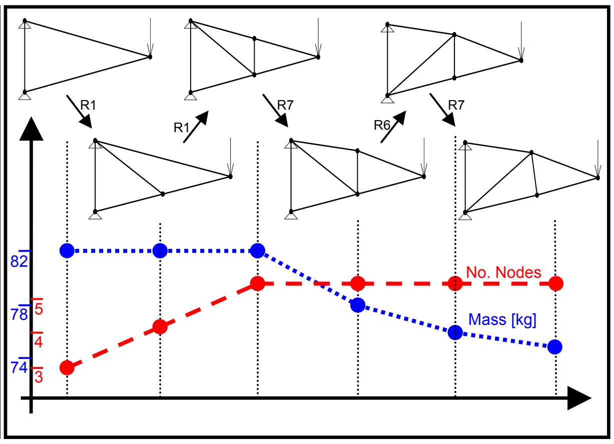


Figure 3. Development of a truss through application of grammar rules.

3. ML-assisted search within the modification-based framework

There are a variety of implementation possibilities for the data modeller and modification advisor from the completely trivial to the highly complex. The very simplest implementation is a modeller that does nothing and a modification advisor that suggests random modifications. This corresponds to the well-known, stochastic generate-and-test search process. Additional heuristics incorporated in search may include a periodic return-to-base and archiving of non-dominated solutions[1]. Enhancement of this approach might include a modeller that observes which modifications result in more drastic search progress and a modification advisor that weights modification selection accordingly. This approach could be further extended to make informed decisions based on more thorough observations of the past search history and, potentially, the histories of previous searches. With this more advanced approach, the system would pursue intelligent design strategies, planning the development of the design solution based on changing circumstances. The use of observation to aid decision-making has been widely used in such applications as robotic control [9] and design [10], though here knowledge is acquired through experimentation not observation of a third party expert. As the knowledge base increases, design strategies that prove to be reliable are favoured and previously observed approaches that are unproductive are avoided. Important factors in the development of such techniques are flexibility, multiobjective search capabilities, robustness and the ability to deal with a wide range of diverse design tasks.

3.1 Identifying design strategies

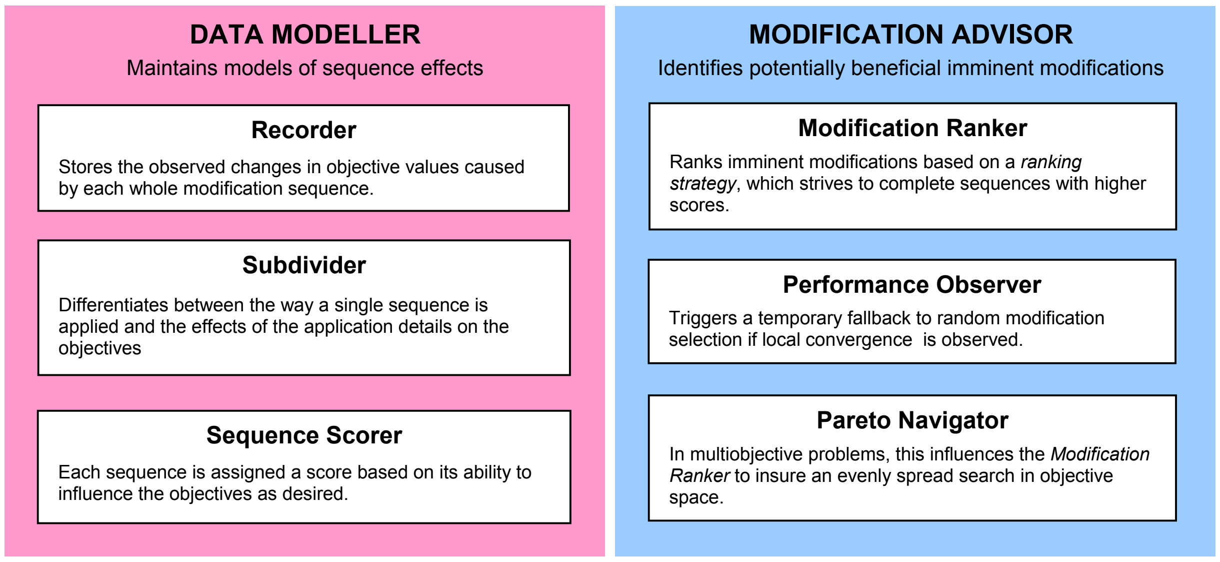
With the aim of introducing how design strategies might be identified within the modification-based framework, we use a simple example of a child's building block game. Modification types might be defined as *select block, lift block, drop block, move block,* etc. Evaluation of the design might involve one or more design objectives, such as calculation of the height or the base footprint of the tower being built or the number of blocks being used. It is quite obvious that repeated lifting and dropping of a block is unproductive, as is repeated selection of blocks without any other action, whereas lifting, then moving, then dropping of a block is more reliably productive. As suggested above, design strategies could be embodied by partial sequences of modifications. Longer partial sequences would typically capture more useful design process information but would be observed with far lower frequency than shorter sequences, which could be expected to capture more simplistic processes, and therefore may be less useful.

3.2. Algorithm overview

Having observed how good and bad design practice might be captured in sequences of modifications, we can propose an ML-based search algorithm that exploits this. The algorithm outlined here is but one of a number of potential implementations. Details of its implementation and that of other potential approaches can be found in [11] and [12]. The algorithm takes a statistical observation-based approach using a data modeller that observes and analyses the effects of sequences of modifications on the design objectives and rates them accordingly with an associated figure of merit. The modification advisor then ranks potential imminent modifications such that highly ranked modifications facilitate high scoring sequences. Parallels can be drawn with reinforcement learning techniques [13] where the figure of merit is associated with the standard penalty/reward system.

---

[1] In the context of Pareto optimality, a non-dominated solution implies one that is unbeaten in at least one of its objectives, by all other solutions found [5].


DATA MODELLER
Maintains models of sequence effects
Recorder
Stores the observed changes in objective values caused by each whole modification sequence.
Subdivider
Differentiates between the way a single sequence is applied and the effects of the application details on the objectives
Sequence Scorer
Each sequence is assigned a score based on its ability to influence the objectives as desired.
MODIFICATION ADVISOR
Identifies potentially beneficial imminent modifications
Modification Ranker
Ranks imminent modifications based on a *ranking strategy*, which strives to complete sequences with higher scores.
Performance Observer
Triggers a temporary fallback to random modification selection if local convergence is observed.
Pareto Navigator
In multiobjective problems, this influences the *Modification Ranker* to insure an evenly spread search in objective space.
Figure 4. The components of the data modeller and the modification advisor adapted for this implementation.

3.3. The data modeller

The data modeller records and analyses the effects of sequences of modifications on the design objectives, assigning a figure of merit to each fixed-length sequence observed.

3.3.1. The recorder

A sample of the progress of a typical search in a multiobjective design problem is captured in Table 1. Referring to the table, if the current iteration is 193, then the search has just applied modification type 1 to the outcome of iteration 192. In so doing, it has completed a single modification sequence, <1>, a double modification sequence, <3-1>, and a triple modification sequence, <1-3-1>. Each sequence is completed subject to the application variables listed in the last column. By looking even further back into the history of modification applications, it is possible to observe the effects of arbitrarily long sequences of modifications. Currently, the algorithm models at fixed pre-defined sequence lengths, though further research will develop modellers capable of dynamic variation of the modelled sequence length as the number of observations increase. Assuming, for the purposes of explanation, that the algorithm is modelling at the level of two modifications per sequence, it will then concern itself only with the just completed <3-1> sequence. The algorithm therefore calculates the changes in objective values according to eqn (1) and stores the result for further analysis.

$$\Delta\mathbf{O}_{<3-1>|\mathbf{v}} = \mathbf{O}_{193}^{After} - \mathbf{O}_{192}^{Before} \tag{1}$$

Table 1. Sample output of a search procedure

Iterate	Obj. Values Before	Obj. Values After	Rule Applied $\in[1,N_R]$	Application Variables $[v_1; v_2; v..]$
…	…	…	…	…
191	[1.0; 3.0]	[0.2; 6.0]	1	[0.2; 1.0]
192	[0.2; 6.0]	[0.1; 8.0]	3	[0.1; 0.1]
193	[0.1; 8.0]	[0.4; 1.0]	1	[1.0; 3.0]
…	…	…	…	…

3.3.2. The subdivider

An additional task of the data modeller is to group observed sequence effects according to the application variables so that, for any particular sequence, each subdivision exhibits differing effects on the design objectives when applied within a particular range of application variables. Figure 5 plots the observed change in truss mass for a truss design problem over the sequence of modifications <1-3>. This data is accumulated by the recorder as a function of a single application variable defined here as the geometric distance separating the application of the first and second modifications in the sequence. It can be seen that the observed changes in the mass are generally much wider spread out in the middle ranges of geometric separation. When the separation distance was zero all the observed changes in mass were zero. This is to be expected for the truss grammar, as rules 1 and 3 are a rule/anti-rule pair. Sequence <1-3> applied with zero distance separating them amounts to addition and immediate removal of a node, leaving the truss unchanged. Every sequence can similarly be resolved into discrete segments chosen so that the difference in the mean and variance of the observed changes in objective values of neighbouring segments is maximized. In Figure 5, the segments that result from subdivision of <1-3> are indicated by vertical lines. In multiobjective problems, subdivision is performed independently for each objective.

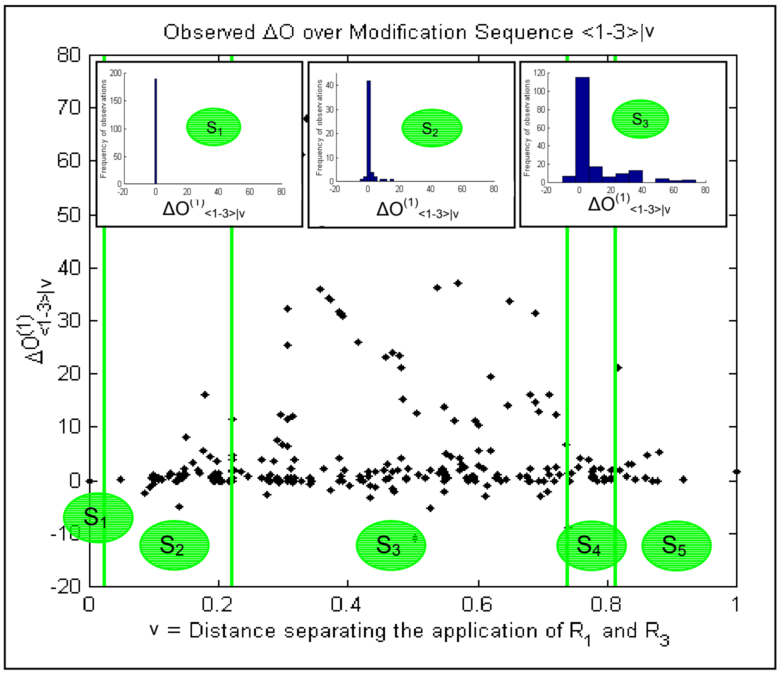


Figure 5. Changes in objective due to <1-3> as a function of the application variable together with histogram plots of the distributions of the observed data in the subdivided segments.

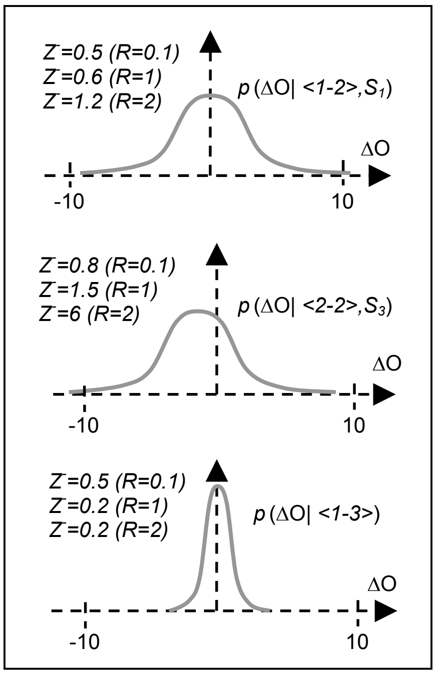


Figure 6. Sample ΔO distributions and their associated scores for different values of *R*.

3.3.3. The sequence scorer

Each sequence or segment thereof, if application variables are used, is assigned two scores, Z^+ and Z^-, for each design objective. The scoring mechanism favours distributions that promise a high probability of large positive or negative changes in the objective values respectively. Essentially the score corresponds to the maximum possible value of $improvement^R$ × *probability of improvement* for a particular distribution as in Figure 5. *Improvement* is defined as the positive or negative change in objective respectively. The *probability of improvement* is inferred from the frequency distributions as in Figure 5. *Risk*, *R*, is typically set at unity, but can be set higher or lower to favour high or low risk/benefit returns from the algorithm. Figure 6 shows typical Z^- scores for a few sample distributions.

3.4. The modification advisor

The purpose of the modification advisor is to provide an ordered list of appropriate modifications to make on the next iteration. It is comprised of three components.

3.4.1. The performance observer

When the complexity of escaping local optima exceeds the scope of the sequence models, the procedure can trap itself by strictly following its preferential treatment of ‘good’ sequences. The role of the performance observer is to analyse the convergence of the archive of solutions and instruct completely random modifications if local convergence is observed. As soon as an advance is made, the modification advisor switches back to its standard practice. Under this regime the search exhibits rapid archive development and high efficiency until it reaches a point where a higher level of creativity is required.

3.4.2. The pareto navigator

Relevant only in multiobjective problems, this component provides an indication of the most desirable change in objective values to best explore the range of non-dominated solutions given the past search history and the current location in objective space. It steers the search away from areas that have been thoroughly searched with little success, as well as objective values excessively far away from the developing archive of non-dominated solutions. The pareto navigator outputs a metric, M_w, for each objective, w, in the interval [-1,1], with –1 or 1 indicating a strong need to decrease or increase the corresponding objective respectively.

3.4.3. The modification ranker

Imminent modifications are ranked via a weighted roulette wheel with weights equal to utility values assigned to each modification. Calculation of the utility is accomplished by analysing the contribution of the imminent modification to high-scoring *sequences*. A simple way of doing this would be to assign these utilities to be equal to the scores of the sequences that each imminent modification would complete, thus ensuring that good sequences are always completed at the next iterate. However, blindly employing such a greedy policy is short-sighted, as is does not allow the algorithm to invest in short term ‘bad moves’ to capitalise later by completing a very good sequence. Strict adherence to a long term-oriented policy, however, never allows the scheme to capitalise on past investments. The technique used here combines some of the principles described by Humphrys [14] for robot planning and more formal techniques described by Bernardo [15]. The algorithm effectively does a search to a depth of N_S of the decision tree. At each node of the tree it combines the score of the sequence just completed with the future utility of that node. The future utility reflects the contribution of the node to sequences completed in the future and is a weighted sum of the utilities of the N_R possible future nodes, with weighting equal to the probability that the corresponding future path will be taken. In MO problems, a combined utility is calculated by summing weighted utilities for each objective with weights equal to the magnitudes of M_w (see 3.4.2).

4. Applications to multiobjective truss design

The first example is the design of an end-loaded symmetrical cantilever truss structure. Figure 8 shows the typical output of a basic search algorithm, which simply applies rules at random, archiving any new non-dominated solutions discovered in the process. Also shown in the figure are some of the corresponding design topologies. Note that the solutions approach the known optimal Prager topology for the cantilever design [16]. The trade-off between the number of nodes and the truss mass is visible in the plot of archived objective values. It is worth noting that since the node movement rule (R7), moves a node a predefined discrete distance, the node locations are effectively constrained to a grid that could be made more fine. Figure 9 plots the average convergence for the developing archive when augmenting simple random generate-and-test synthesis with the technique presented here. Pareto 'convergence' is measured by calculating the hyper-volume, e.g. shaded area in Figure 8, of objective space that is dominated by the archive of non-dominated solutions. For the first 3% of the search, convergence tends to be unchanged due to the initial exploration stage used by the augmented method to increase its knowledge base. Figure 9 includes samples of sequence scores obtained in the search, reflecting the recognised utility of the various sequences, with the sequences we would expect to perform well, from inspection of the truss grammar, indeed showing higher scores. For example, the application of node movement after node addition appears to be a generally good strategy, whereas removal of a node just after placing it is worthless. As expected, similar sets of scores were found for other simply loaded truss problems.

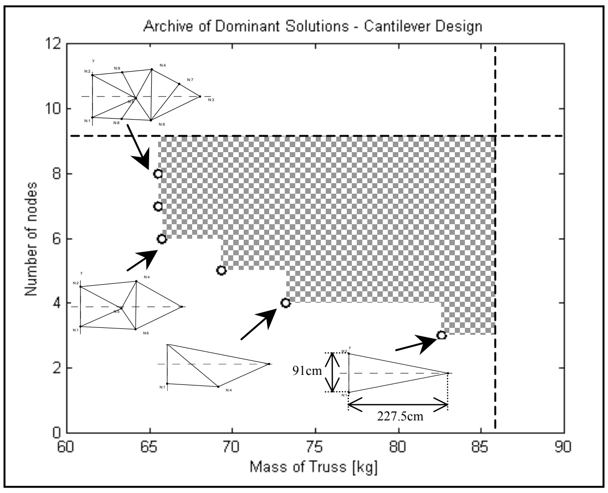
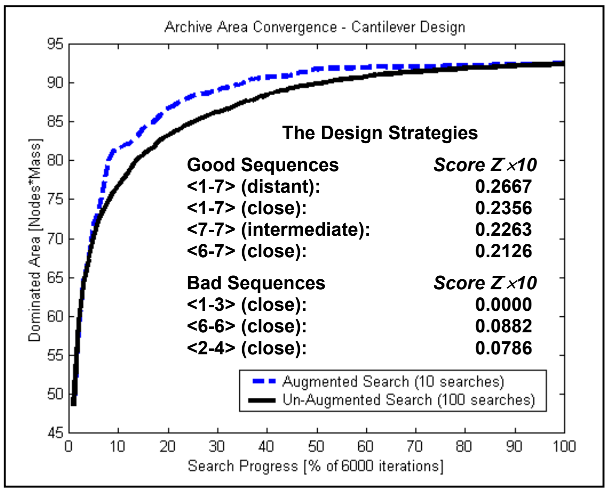


Figure 8. The cantilever truss solution archive



The Design Strategies

Good Sequences	*Score Z×10*
<1-7> (distant):	0.2667
<1-7> (close):	0.2356
<7-7> (intermediate):	0.2263
<6-7> (close):	0.2126
Bad Sequences	***Score Z×10***
<1-3> (close):	0.0000
<6-6> (close):	0.0882
<2-4> (close):	0.0786

Figure 9. Multiobjective convergence.

5. Application to a multiobjective sliding tile puzzle

The second synthesis task considered revolves around the well-known puzzle of sliding tiles, also sometimes called the 8-puzzle. The classic version of this puzzle consists of a number of tiles arranged in a frame in a grid-like arrangement with one tile missing, as shown in white in Figure 10b. When arranged correctly, the tiles typically form an image or sequence of numbers. The challenge of the puzzle is to reconstruct the proper arrangement of tiles from an initial jumbled up arrangement by successively sliding tiles into the free space. Here, we consider multiobjective design of dual-colour bitmap images. The initial bitmap image is a jumbled combination of the two sets of coloured tiles needed for the final images. Each objective to be minimised is the difference between the current bitmap and a unique 'goal' bitmap. Expressed as a multiobjective problem, this promises to reflect the trade-offs between conflicting goals that might lead to the development of 'morphed' bitmaps of varying degree, with the goal bitmaps obtained at the extremes of the front of non-dominated solutions. The modifications defined for the purposes of the data modeller consisted of 4 rule types, moving the free space up, down, left and right, with no application variables. The sequence length modelled was $N_S = 2$. Figure 10a shows two typical examples of final archives that emerge from the augmented and un-augmented searches. With the search limited to 12000 iterations, the higher efficiency of the augmented search translates to a greater capability in exploring and advancing the pareto front. Figure 10c shows the archive convergence averaged over a number of separate searches for both the augmented and un-augmented searches.

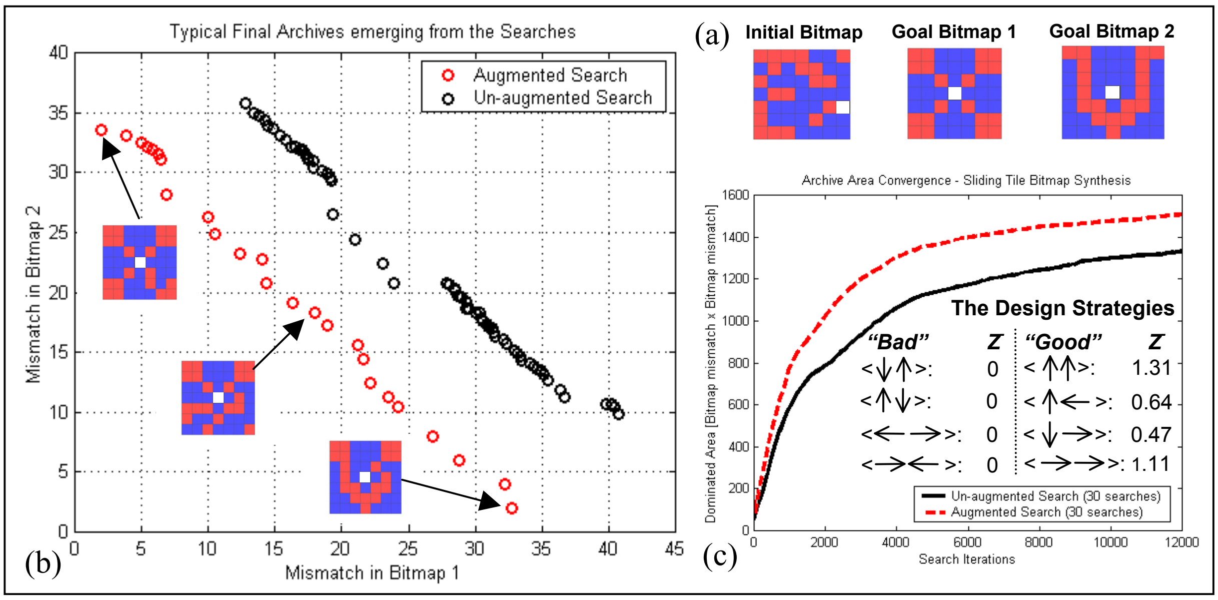


Figure 10. a) The archive of non-dominated solutions, b) the initial and goal bitmaps and c) archive convergence.

6. Implications for general engineering design synthesis

The examples above highlight the ability of the ML-based synthesis technique to speed up search convergence based on observations of good and bad design strategies. What is, however, more important, is that despite the major differences in the natures of the design tasks, no part of the design algorithm had to be adapted to be applied the various problems. The problems simply had to be expressed within the modification-based design framework. In principle the ML-based design algorithm could function in the same way with the same expected results for any problem as long as it fulfils the requirements of the modification-based framework. The design strategies obtained as a result of the ML-based search implementation are also of value as it allows the user to see exactly how modifications might be changed, or perhaps even dispensed with to further improve the search and the resulting designs generated. Since the modification operators themselves are being improved, all subsequent searches with the same modification set benefit from past observations. Another possibility would be to allow suitable modification sequences to gradually become "meta-rules", i.e. lumped rules in their own right, by defining the sequence as a self-contained rule type with its own numeric label. Meta-rules would be applied as single rules forcing only one evaluation by the evaluator and further increasing search efficiency. This more pro-active application of the rule set will lead to creation of a technique for refining modification sets through learning, a principle similar to that proposed by Gero et al. [17]. Implications for general engineering design are the ability to easily develop, improve and learn from design synthesis tasks formulated in the way described here and to cultivate a growing body of knowledge that would improve search efficiency.

7. Conclusions

A method for exploiting machine-learning-based techniques in engineering design within a modification-based framework has been discussed. A specific implementation of the methodology has been presented in greater detail and successfully applied to two diverse design tasks. Advantages of this approach include flexibility, re-usability of acquired knowledge and insight into the design synthesis problem. The framework facilitates easy extension to a range of design synthesis tasks with minimal effort from the user. The specific implementation described holds the additional advantage that it does not at all rely on design state information, making it suitable for synthesis formulations that rely on modification rules. In addition to the algorithm presented here, more advanced ML-based approaches have been implemented [11]. Research currently underway is directed at developing new ML-based design synthesis algorithms to provide even faster convergence within the modification-based framework and applying them to design synthesis problems in more diverse fields, such as electronics and mechanism design, as well as highly constrained structural design.

8. References

[1] Morelle, P., "Optimisation strategies for industrial multidisciplinary problems," in Proc. *4th ASMO-UK/ISSMO Conf. Eng. Design Opt.,* Newcastle, UK, July 4-5 2002, pp 9-19.

[2] Rubin, S.M., *Computer aids for VLSI design,* Addison-Wesley Longman, Boston, MA, 1987. (See also http://www.rulabinsky.com/cavd/)

[3] Grierson, D.E., Khajehpour, S., "Method for Conceptual Design Applied to Office Buildings," *Journal of Computing in Civil Engineering,* April 2002, pp 83-103.

[4] Shea, K., Starling, A.C., "From Discrete Structures to Mechanical Systems: A Framework for Creating Performance-Based Parametric Synthesis Tools", Proceedings of the AAAI'03 Spring Symposium, 'Computational Synthesis', Stanford, CA, 2003.

[5] Borkowski, A., Jendo, S., *Structural Optimization,* Volume 2, "Mathematical Programming," (eds. Save, M., Prager, W.), Plenium Press, New York, pp. 311-341 1990.

[6] Suppapitnarm, A., Seffen, K.A., Parks, G.T., Clarkson, P.J. "A simulated annealing algorithm for multiobjective optimisation," in *Engineering Optimization*, 33 (1), 2000, pp. 59-85.

[7] Fonseca, C.M., Fleming, P.J., "An Overview of Evolutionary Algorithms in Multiobjective Optimization," *Evol. Comp.* 3, pp. 1-16, 1995.

[8] Shea, K., Cagan J., Fenves, S.J. "A Shape Annealing Approach to Optimal Truss Design With Dynamic Grouping of Members," *Trans. of the ASME,* Vol. 119, Sept. 1997, pp. 388-394.

[9] Bresina, J., Drummond, M., "Integrating planning and reaction: A preliminary report," in J. Hendler, editor, *SRC TR 90-45*. Systems Research Center, University of Maryland, 1990.

[10] Wang, X. "Learning planning operators by observation and practice," Proc. *2nd Intl Conf. AI Planning Systems*, 1994.

[11] Vale, C.A.W., Shea, K., "Learning Intelligent Modification Strategies in Design Synthesis", Proceedings of the AAAI'03 Spring Symposium 'Computational Synthesis', Stanford, CA, 2003.

[12] Vale, C.A.W., *Multiobjective Dynamic Synthesis via Machine Learning,* M.Phil Thesis, University of Cambridge, UK, 2002.

[13] Mitchell, T.M., *Machine Learning,* McGraw-Hill, Singapore, 1997

[14] Humphrys, M., "Action selection methods using reinforcement learning," in Maes, P., Mataric, M., Meyer, J.-A., Pollack, J., and Wilson, S. W., editors, *From Animals to Animats 4: Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior*, Cambridge, MA, MIT Press, Bradford Books, pp 135-144, 1996.

[15] Bernardo, J.M., Smith, A.F.M., "Sequential Decision Problems," in *Bayesian Theory,* John Wiley & Sons, pp. 13-104, 1998.

[16] Prager, W. 'Optimal layout of Cantilever Trusses,' *Journal of Optimization Theory and Applications,* Vol. 23, no. 1, Sept. 1977, pp. 111-117

[17] Gero, J.S., Louis, S.J., Kundu, S., "Evolutionary learning of novel grammars for design improvement,".AIEDAM, 8(3), 1994, pp 83-94.

For more information please contact:

Christopher Vale   Cambridge University, Dept. Engineering, CB2 1PZ, Cambridge, United Kingdom
Tel: Int +44 1223 766961   Email: cawv2@cam.ac.uk