Artificial Intelligence for Engineering Design, Analysis and Manufacturing (2007), *21*, 351–366. Printed in the USA.

DOI: 10.1017/S0890060407000327

Inductive machine learning of optimal modular structures: Estimating solutions using support vector machines

SEAN HANNA
University College London, London, United Kingdom

(Received July 2006; Accepted May 2007)

Abstract

Structural optimization is usually handled by iterative methods requiring repeated samples of a physics-based model, but this process can be computationally demanding. Given a set of previously optimized structures of the same topology, this paper uses inductive learning to replace this optimization process entirely by deriving a function that directly maps any given load to an optimal geometry. A support vector machine is trained to determine the optimal geometry of individual modules of a space frame structure given a specified load condition. Structures produced by learning are compared against those found by a standard gradient descent optimization, both as individual modules and then as a composite structure. The primary motivation for this is speed, and results show the process is highly efficient for cases in which similar optimizations must be performed repeatedly. The function learned by the algorithm can approximate the result of optimization very closely after sufficient training, and has also been found effective at generalizing the underlying optima to produce structures that perform better than those found by standard iterative methods.

Keywords: Machine Learning; Optimization; Structures; Support Vector Machines

1. INTRODUCTION

Nature builds by trial and error via the very effective but slow and costly process of evolution. Similarly, models of iterative analysis and trial form the basis of structural optimization methods in which many solutions are evaluated in a computational loop. Many large or complex problems present a problem, however, because of long computation times of many members or difficulty in modeling fine details explicitly. This paper presents an alternative strategy. Although for several centuries the mathematical tools for explicit analysis have been dominant, the vast majority of design decisions throughout history have been based on experience of precedents: a practiced builder would know what would stand or fall without having to test it. Just as this intuitive knowledge is essential for design, the strategy can also aid in optimization. This work uses inductive learning to approximate a model that is as good an estimate of the structure's behavior, but far more efficient than precisely computing the structure's behavior. It demonstrates that nearly optimal solutions to a well-defined structural design problem can be found by training a machine learning algorithm on examples of other solutions found by a traditional optimization procedure.

Once trained, the advantage of such a machine is the same advantage that the human builder's training and experience provide: the ability to build quickly and without failed attempts. A structural problem is chosen that involves the repeated optimization of many interconnected modules, and thus takes full advantage of this increase in speed. It is also a problem of sufficient complexity that the solution cannot be calculated directly, but must be found by iterative simulation and testing. An algorithm capable of arriving at a general solution by inductive learning on presented examples is thus highly beneficial.

Optimization algorithms including gradient descent (GD) and genetic algorithms (GAs) make repeated evaluations of the strength or stiffness of different structures to design an effective shape to counter the given load. This process is time consuming, however, requiring repeated iteration for each new design, and in the case of GD is subject to inferior solutions because of local optima in the search space. This paper uses inductive learning to eliminate the need for this iterative step once sufficient examples have been generated to save processing time and achieve more constant fitness of solutions. If the optimization is repeated many times for many sets of loading conditions, the optimal shape of the structure can be imputed as a function of the load. This paper investigates the use of a support vector machine (SVM) to learn this function of optimal structures given the tensile or compressive loads in each axis, replacing the iterative analysis and optimization entirely (Fig. 1).

Reprint requests to: Sean Hanna, Bartlett School of Graduate Studies, 1-19 Torrington Place, London WC1E 7HB, UK. E-mail: s.hanna@ucl.ac.uk

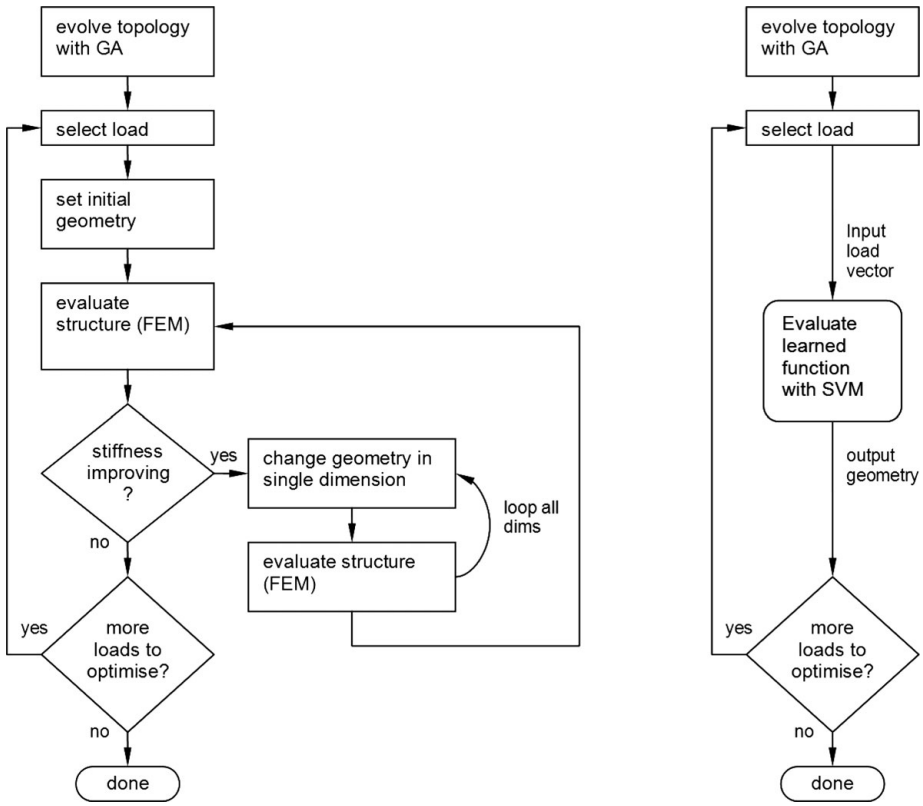

Fig. 1. The (left) iterative optimization process is replaced entirely by the function learned by the (right) support vector machine.

In addition to speed, learning offers another, perhaps greater benefit: the model may incorporate a complexity beyond what can be modeled explicitly. Analysis by the finite element method (FEM) is an approximation of a structure's actual behavior: material properties such as elasticity are set for each element as though the material is homogeneous even though these may actually represent the effect of finer scale microstructure geometry. Nevertheless, such finite element approximations can be made to simulate such complex microstructures as those of wood (Astley et al., 1997) and bone. Digital fabrication methods, too, may introduce added complexity into a structure that does not exist in the initial model—a slight thickening of members at the nodes because of material viscosity or raster surface irregularities, for example. Modeling the geometry of these irregularities would normally require a drastic increase in mesh resolution and corresponding computation time. This paper proposes that these complexities caused by the manufacturing process can, in fact, be incorporated into a model not by explicit simulation, but by inductive learning based on the behavior of samples. In this manner the changes of behavior can be taken into account even when the precise geometrical or material cause is not known explicitly. In the work presented, all experiments are entirely virtual, using a finite element simulation altered based on preliminary material testing, but the method is amenable to the use of data derived from the performance of real sample structures.

1.1. The structural problem

Space frame structures, a set of linear members oriented in any direction in three-dimensional space and connected at nodal points either by rigid or flexible connections, are investigated in this work. The specific problem addressed is that of small-scale space frames, an example of which is shown in Figure 2. The overall dimensions of this object as fabricated are $1 \times 1 \times 2$ cm, and the individual struts within it are less than 1 mm in length.

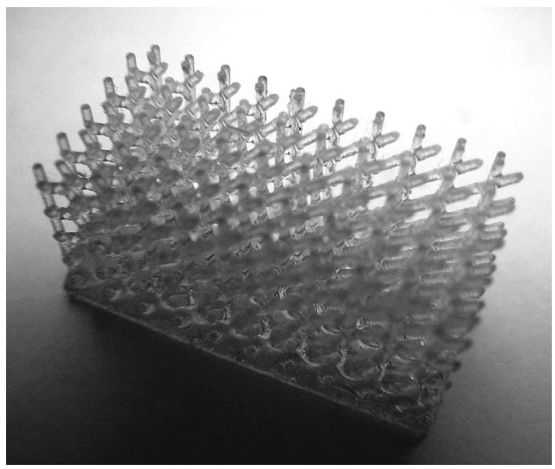

Fig. 2. A modular structure fabricated by stereolithography.

https://doi.org/10.1017/S0890060407000327 Published online by Cambridge University Press

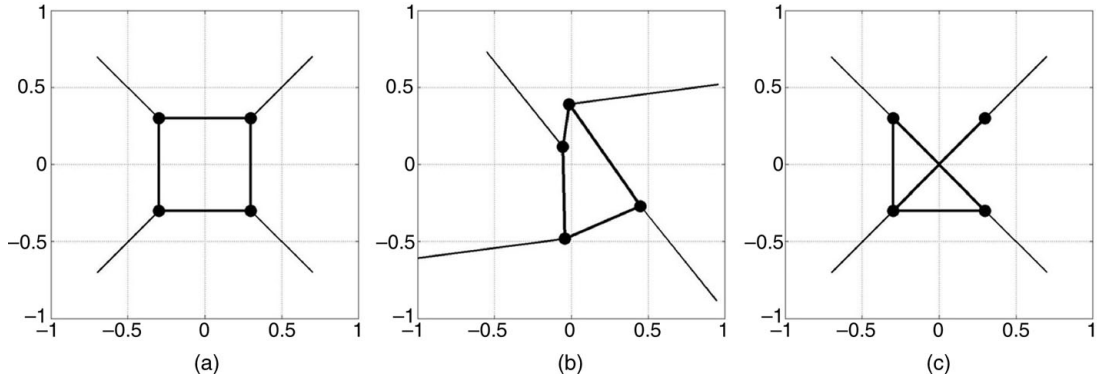

Fig. 3. An illustration of a change in the geometry and topology of a structure.

To define a particular space frame one must specify both the topology (connections between members) and the geometry (locations of the nodes in three-dimensional space) that determine the length and orientation of each member. Figure 3 illustrates this schematically. The center diagram displays a change in geometry from the left diagram by altering the positions of the nodes, and the right diagram displays a change in topology with struts connecting the same nodes differently.

The overall structure is based on a modular topology, so that it can be scaled to volumes of any size. Its volume is subdivided into a grid of cubes, which will be referred to as "unit cubes," each containing a portion of structure with identical topology such that each is connected to its neighbors to form a continuous structure (Fig. 4). In the related problem of optimization with freely parameterized materials (Bendsøe & Sigmund, 2003) the result is a vector field of principle stresses that varies continuously at each point in the volume. Although intended primarily as an exercise for attaining upper bounds on performance with real materials or as a framework for composite design, by optimizing the geometry of each unit cube to best resist the local stress of such a field, such optimal continuous structures can be designed. Bendsøe and Sigmund split the problem into the two coupled subproblems of local anisotropy and material distribution, which can be addressed in the problem representation used here by altering the direction and thickness of the struts, respectively.

Previous work by Hanna and Haroun Mahdavi (2004) has resulted in a method for rapid optimization of large and complex structures using this modular "unit cube" approach. By confining optimization to local unit areas, this prior method finds fit structures with a member count too high to manage using a standard global optimization, at greatly improved speed. An object under a complex loading condition exhibits differing stresses at various points in its volume. The vector of stresses in the three axes (x, y, and z) is sampled at the location of one of the unit cubes, and used to optimize the local module of structure at that location. Although the estimation of stresses is based on a simple FEM loading of an isotropic material, the free parameterization of Bendsøe and Sigmund could also be used as an initial starting point.

The ideal result is a modular structure as displayed in Figure 4 (bottom), with gradual changes in the geometry of the structure as the stresses change continuously across the volume of the object. With material concentrated in high stress zones and internal struts aligned to counter the changing direction of the stress vectors, these can achieve a stiffness of approximately 70 times that of a uniform structure. Geometry is optimized separately from topology, so that the repeated use of a single topology allows gradual geometrical changes to occur without weak points at the connection between differing units.

1.2. Manufacturing

The structures considered are designed to be fabricated by a digitally controlled process, the main advantage of which is the low cost of complexity. Such techniques are increasingly used in such large-scale manufacturing as automobiles and architecture (Sischka et al., 2004), but the development of smaller scale rapid prototyping technology allows manufacture at scales less than 1 mm and the fabrication of intricate internal structures. This has not yet become commercially viable for mass production, but several manufacturers of small-run and customized items such as lampshades and hearing aids use the technology. Researchers are preparing for the increasing accuracy and decreasing cost of the technology in the future.

Stereolithography is the method considered here. It begins with a tank of liquid photopolymer that is sensitive to ultraviolet light. An ultraviolet laser "paints" the object as a series of horizontal layers, exposing the liquid in the tank and hardening it. Once completed, the object is rinsed with a solvent and then exposed in an ultraviolet oven that thoroughly cures the result. Current machines are capable of creating very fine structures and build to a resolution of 0.05 mm.

The horizontal stratification inherent in the process adds a degree of complexity to the problem of optimization, as members built at different angles to this horizontal plane have varying degrees of stiffness (Haroun Mahdavi & Hanna, 2004). These were measured (Fig. 5; Haroun Mahdavi & Hanna, 2003) and factored into the optimization of examples presented to the machine for learning by modifying the stiffness of each element as calculated in the finite element model by a function of its angle.

https://doi.org/10.1017/S0890060407000327 Published online by Cambridge University Press

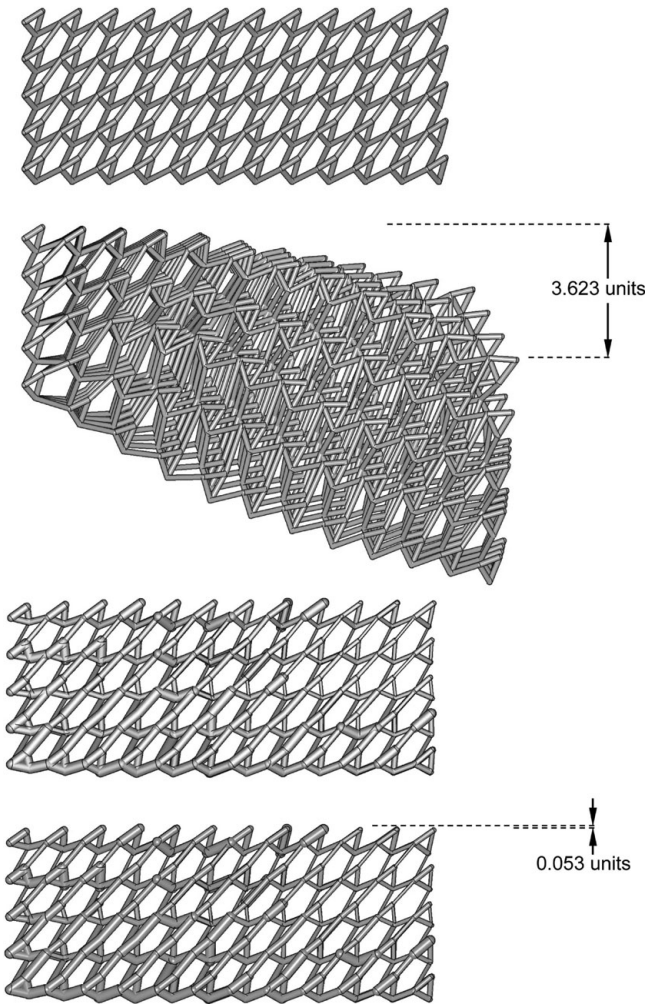

Fig. 4. A modular space frame forming a cantilever beam. Both have the same overall mass and topology, but (top) identical modules deflect far more under loading than do the (bottom) individually optimized ones.

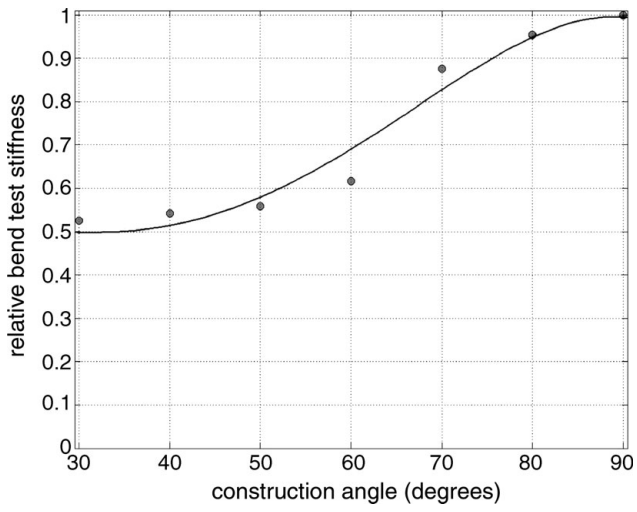

Fig. 5. The relative stiffness of members constructed at varying angles from the horizontal plane.

This step incorporates into the function to be learned a level of detail beyond the FEM model. The actual cause of this change in stiffness occurs between individual layers of resin, at a resolution far less than a single element, and is taken not from a simulation but from empirical observation. Similarly, the result of resin pooling at nodes and other fine details may be measured without ever being explicitly modeled. The effects of varying build resolutions and other manufacturing methods have not yet been investigated, but these could be incorporated in the same way. In each case, any additional factors affecting the training data become integrated into the function that replaces the FEM and optimization model.

2. RELATED RESEARCH

2.1. Optimization of structures

Initial data to be used in training any learning algorithm can typically come from several sources, including experts, previously published historical or experimental data, and simulation (Reich, 1997). Because of the repetitive nature of the problem and the well-defined behavior of structures, simulation by the FEM is both the most efficient and accurate. In the design task under consideration here it is a set of optimal solutions that is required.

Several techniques have been devised for generating the topology of continuous solids analyzed by the FEM. Many deterministic approaches stem from an iterative process of material distribution first proposed by Bendsøe and Kikuchi (1988), including the shifting of nodal points in the FEM representation toward high stress zones (Chen, 2002). Alternatively, stochastic approaches such as GAs have been used to specify a configuration of holes and solid using Voronoï diagrams or a list of hole shapes (Schoenhauer, 1996).

Discrete element structures (e.g., trusses, space frames) of the kind considered here involve both the design of the topology of connections as well as their position and size. Much early research in this area (e.g., Adeli & Cheng, 1993) has been in refining only the shape or member sizes, rather than the topology (in terms of members connecting the nodal points of the structure) to optimize the weight of space trusses by determining the width of each member in a given structure. More recent research has concentrated on topological optimization, or both topology and shape together. Steel space frames and bracing topologies for tall buildings have been designed by GA (Pezeshk & Camp, 2002), either by encoding the possible member connections within each structural bay in the genome (Murawski et al., 2000; Kicinger et al., 2005) or by beginning with an acceptable unoptimized solution and refining the topology by removing connections (Ping, 1996). These typically begin with set node positions, but generative, rule-based systems including shape annealing (Shea & Cagan, 1998) in which transformations are selected depending on design criteria, and those evolved by GA (Kicinger et al., 2005), provide more open-ended approaches.

https://doi.org/10.1017/S0890060407000327 Published online by Cambridge University Press

The problems of topology and geometry can also be separated and solved by nesting one within the other. In von Buelow (2002), a two-stage algorithm was used in which an outer GA evolved a topology for a space frame structure, whereas another GA found the geometry for each member of the population. The present work adopts a similar approach. Previous work by Haroun Mahdavi and Hanna (2003) has also used GA for topology, but it has been found that GD is more efficient for shape optimization. This method, extended here, uses a single topology previously found by a nested GA/GD process for each unit of the overall structure, but is concerned specifically with the process of geometry optimization.

Versions of the above geometry and topology optimizations have been applied to modular structures, particularly to realize desired properties in material microstructures (Bendsøe & Sigmund, 1999). These are typically arrayed as identical modules to form an isotropic structure. Material distribution methods have also been extended to find solutions requiring materials of free density and elasticity parameters (Bendsøe & Sigmund, 2003, pp. 190–204), usually approximated by composites of discrete materials in fabrication. It is for such situations that the present method is proposed to allow parameters to change continuously between modules.

2.2. Machine learning for behavior and structure

Machine learning has long been applied to structures and in the domain of civil engineering, most commonly as an enhancement of the optimization process. A recurring bottleneck in optimization is the simulation of a design's behavior, which can either be time consuming because of the complexity of the model, or simply incorrect because of incomplete knowledge. This can be addressed by "shallow modeling" a system's observed behavior with inductive learning (Arciszewski & Ziarko, 1990). Discrete, symbolic learning methods have been used to construct rule-based systems that draw relationships between design parameters that predict the performance of systems from individual beams (Arciszewski & Ziarko, 1990) to the steel skeletons of entire buildings (Szczepanik et al., 1996). Subsymbolic inductive methods such as artificial neural networks have been used also to predict structural and material performance (Reich & Barai, 1999) and the behavior of mechanical systems such as propeller blades (Neocleous & Schizas, 1995; Reich & Barai, 1999).

Some of the most recent and complex problems involve structural prediction in the field of bioinformatics, in which the molecular composition of proteins can be too computationally expensive to simulate fully. One stream of research is in the prediction of the secondary and tertiary structure of proteins by machine learning, where the inputs are the actual DNA string and the outputs are the predicted three-dimensional structure of the protein. Various learning algorithms have been used, including artificial neural networks (Meiler & Baker, 2003) and SVMs (Wang et al., 2004).

Regardless of the method used in simulation, the repeated iteration of generating and evaluating solutions is the other major hurdle in optimization. Inductive learning has been found useful to improve the speed and quality of this loop by reusing knowledge of previous designs or iterations. Murdoch and Ball (1996) have used a Kohonen feature map to cluster bridge designs in an evaluation space, and Schwabacher et al. (1998) have used a symbolic learning algorithm, C4.5 (Quinlan, 1993), to select appropriate starting prototypes and search space formulations for a parametric optimization of yacht hull and aircraft designs. Both allow a rapid reevaluation of previous work that improves the optimization when run again to new specifications or fitness criteria.

It is the aim of the present work to use a learning algorithm to replace the optimization process entirely, both the simulation and evaluation loops (Fig. 1). Although much previous research has concentrated on inferring rules to guide a design (Arciszewski & Ziarko, 1990; Neocleous & Schizas 1995; Szczepanik et al., 1996; Reich & Barai, 1999) or on suggesting a starting point on which to improve (Murdoch & Ball, 1996; Schwabacher et al., 1998), we use induction to derive a function that directly maps a given load condition to an optimal solution.

2.2.1. Algorithm selection

The choice of algorithm is dependent on the learning problem at hand, including the form and availability of data, and the goal of learning (Reich, 1997). The goal, in this case, is induction: to derive a generalization based on previous evidence of optimal structural solutions. Duffy (1997) lists six major machine learning techniques, of which three potentially apply.

1. *Analogical* or case-based reasoning techniques explicitly represent past examples in such a way that they can be retrieved and adapted to suit new problems.
2. What Duffy terms *induction*, specifically symbolic induction, allows a general rule or pattern to be generated to fit the data. Symbolic algorithms with discrete output, such as rough sets (Arciszewski & Ziarko, 1990) and C4.5 (Quinlan, 1993), yield explicit classification or parameter ranges and have therefore been used to estimate behavior or recommend design decisions in symbolic or labelled form.
3. *Artificial neural networks* are part of a class of subsymbolic algorithms (including, more recently, SVMs) that can result in a continuous output, and therefore interpolate exact output values to a finer degree than is specified by the input set. These also perform induction in the form of a continuous function.

The data form of this structural optimization problem is most suited to the third category. The solution to the structural shape is naturally a continuous function, and it has been noted that discretization is detrimental to optimization performance (in tests by the authors), or can lead to large learning error rates (Reich, 1997). As the problem is real valued overall and output is of higher dimensionality than input, it is this subsymbolic class of algorithms that is appropriate.

3. LEARNING METHODOLOGY

3.1. The optimization model

The proposed method builds upon the optimization procedure introduced in Section 1.1, which uses a FEM analysis at two stages: first to estimate the global distribution of stress across the entire loaded object, and second to guide the incremental optimization of each single module.

This paper addresses primarily the second analysis, in the optimization of the individual structural modules. In the final object, each of these will be connected to a number of slightly varying neighboring modules through which the loads will be distributed, but for the sake of efficiency and because the final structure is not known, the optimization deals with each unit in isolation. The struts internal to one unit of structure and all struts that connect it to its adjoining neighbors are used in the model, which, because of the grid arrangement, entails a duplication of each of the struts shared by one unit and its neighbors. These pairs of struts determine the boundary conditions: each component of the applied load (in the x, y, or z axis) is divided equally between all struts that connect to neighboring modules in the positive direction of that axis, whereas the corresponding degree of freedom is constrained in the negative direction. If three struts connect to units above a module (positive z axis), for example, there will be three corresponding struts connecting to units below. For a total applied load of -0.6 units in the z axis, -0.2 would be applied to each of the upper struts at their extreme node, whereas the lower nodes of the lower struts are constrained to the x–y plane. The same is true of the other axes. The extreme node of a single unloaded strut is fixed in all three axes to restrain the whole system. Each strut is modeled by a 12 degree of freedom beam element with one node at each end. All joints between struts are fixed in both bending and rotation.

All optimization optima are tested relative only to each other; therefore, where possible, no real-world units are used for dimension, force, and so forth. Actual material properties vary between resins used in stereolithography; but they are kept at constant default values to simulate a standard isotropic solid with a Young's modulus of 7000 kPa, a Poisson's ratio of 0.3, and a density of 2500 $\mathrm{kg/m^3}$. Adjustments made to simulate the effect of the manufacturing process on struts of varying angles (Section 1.2) were made after FEM analysis by dividing the calculated deflections by the measurements taken in physical testing. The member sections are treated as cylindrical with identical moments of inertia in each axis and torsional stiffness assumed from the polar moment of inertia for a cylinder $J = I_1 + I_2$.

3.1.1. Optimization by GD

The optimization addressed here with inductive learning is GD on node positions to minimize the total deflection in a structure under the specified load, as applied to a unit cube. Although GD is known to provide only local optima, previous investigations of this type of structure (Haroun Mahdavi & Hanna, 2004) have demonstrated that the search space is predominantly convex, and thus appropriate for GD search. Irregularities are introduced when the effects of the manufacturing process on materials (Section 1.2) are simulated, but these do not substantially affect the quality of the solutions found. In particular, the performance of deterministic GD has been shown superior to a stochastic GA both in computation time and in fitness of solutions found. Dependence on initial starting solutions is low so the optimization is only run once for each unit of structure, but all runs are begun from the same initial solution taken from a prior optimization for an equal tensile load in all three axes.

Simultaneous optimization of member section diameters and nodes was found to be more difficult for gradient methods. Instead, the nodes are first determined by optimization as above with a constant member section, and final member sections chosen with an area proportional to the stress in the member. Simulation of stress and deflection is performed using the FEM, using OpenFEM, an open-source finite element toolbox accessed via MATLAB.

3.1.2. Approximations in the model

Although it is clear that several approximations are made in the above model, particularly in the abstraction of a single unit of structure in isolation, these approximations have been shown effective in practice. Refining these is the focus of further research, beyond the scope of this paper. The work presented here will focus specifically on the ability of the machine learning algorithm to approximate the output of any optimization model, and as such, is concerned more with internal consistency. The SVMs can be trained to learn several slightly different versions of the above description, and it is assumed that its output would only improve if the optimization model can be made more accurate.

3.2. The learning algorithm

SVMs (Vapnik, 1995) were chosen to perform the learning described in this paper. They can be generally described as a type of linear classifier that uses a nonlinear kernel function to map input data to a sufficiently high dimension such that it can be separated by a hyperplane (Duda et al., 2001). The transform resulting from this kernel function ensures this hyperplane is nonlinear in the original input space. Thus, the SVM can just as easily be used in regression to a nonlinear function as in classification. They will be used in this capacity to learn the function of optimal structures.

Given a data set D, consisting of an input vector $\mathbf{x}$ and a response vector $\mathbf{y}$, the function to be learned,

$$\mathbf{y} = f(\mathbf{x}), \tag{1}$$

is approximated by the SVM by building a model $f'(\mathbf{x})$ based on D, which enables the estimation

$$\mathbf{y}' = f'(\mathbf{x}). \tag{2}$$

https://doi.org/10.1017/S0890060407000327 Published online by Cambridge University Press

The type of SVM used in this paper is a least squares SVM (LS-SVM) in which the solution follows from solving a set of linear equations instead of quadratic programming for classical SVMs (Suykens et al., 2002). The kernel is the Gaussian radial basis function.

3.2.1. Learning objective

The design objective to be learned is to find the best structural geometry for a single modular unit given the input of its external load. The task is the following: for each set of loads, find the set of nodal points that represent the optimal structure (Fig. 6). The input **x** is the three-dimensional vector of external loads corresponding to the stress (in either tension or compression) in the three axes of a given unit cube. This is represented by the components in the directions of the x, y, and z axes:

$$\mathbf{x} = (x_{(x)}, x_{(y)}, x_{(z)}). \tag{3}$$

The output structure **y** consists of the nodal point positions for the optimal structure as found by prior optimization. This is the set of (x, y, z) coordinates for each of the nodal points y_i:

$$\mathbf{y} = (y_{1(x)}, y_{1(y)}, y_{1(z)}, y_{2(x)}, y_{2(y)}, y_{2(z)}, \ldots, y_{n(x)}, y_{n(y)}, y_{n(z)}). \tag{4}$$

The nodes are also located in three-dimensional space, so for a topology of n points the output **y** is a $3n$-dimensional vector. The inclusion of section sizes and other optimization parameters such as material properties would be a simple extension of the dimensionality of **y**, but in this paper only the nodes are used.

3.2.2. Output complexities

The output vector **y** to be learned is found by GD (Section 3.1.1), and takes into account not just the behavior of an ideal, homogeneous space frame, but also the complexities caused by the fabrication process (Section 1.2). The function of node positions to the resulting stiffness of a standard finite element space frame model,

$$\mathbf{s} = g(\mathbf{y}), \tag{5}$$

is illustrated in part in Figure 7 (left), where the horizontal plane represents a two-dimensional slice through the $3n$-dimensional space of nodal points **y** and the vertical axis represents the stiffness **s**. When changes in member strength because of fabrication angle are simulated, a revised stiffness function

$$\mathbf{s}' = g'(\mathbf{y}) \tag{6}$$

reveals the more complex behavior illustrated in Figure 7 (right). It is this function on which GD is performed to optimize stiffness. Although the function $g'(\cdot)$ used here is derived from iterative gradient on the modified finite element model, it is this function that could also incorporate samples taken from physical testing.

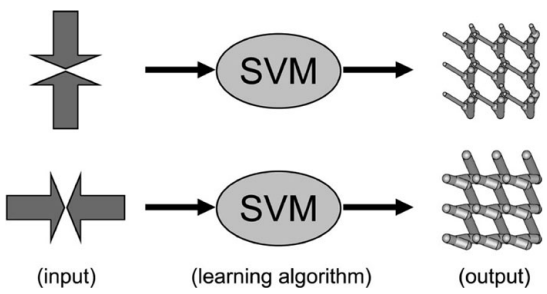

Fig. 6. Different force inputs result in ideal geometry outputs.

3.3. The data sets

Learning results of two different structural models based on the same unit module were compared. Model A is more constrained, having all unloaded boundary nodes fixed with pin joints and identical material properties for all elements. In model B, boundary nodes are fixed only in the axes in which they connect to surrounding unit cubes (Section 3.1) but are free to slide in the corresponding plane to more accurately simulate the flexing of the surrounding structure. This results in the behavior of model B more closely resembling reality, but makes the search space more complex. For practical reasons, optimization of model B was terminated after 110 iterations, regardless of whether an optimum was found. Although GD optimization of model A always resulted in a near global optimum, the variance of solutions found for model B was greater, amounting to considerable increased noise in the function to be learned.

A single topology was used consisting of four nodes per unit cube, resulting in a 12-dimensional output **y**. For models A and B, the data set D was created not to uniformly sample the entire space of possible solutions, but for a normally distributed range of forces and the associated optimal solutions.

Each training sample is created by generating a random input vector **x** from the normal distribution with mean of 0 and standard deviation (σ) of 1, resulting in a range of approximately [−3:3] units of force in each of the three axes. The actual distributions of each of the components of **x** are plotted in Figure 8. The nodal point outputs **y** are found by the GD method described in Section 3.1.1 and result in asymmetrical distributions of nodal positions throughout the space of the unit cube. The distributions of each of the four nodal points in the three axes of space are shown in Figure 9. Although the positions of nodes are not constrained by the optimization algorithm, the repeated nature of the structural modules implies a maximum bound on the search space of one unit for each of the components of **y**. The variance in the data set for each of the points is 0.72, 0.49, 0.53, and 0.56 units in this space, indicating a large portion of the space was sampled in D.

https://doi.org/10.1017/S0890060407000327 Published online by Cambridge University Press

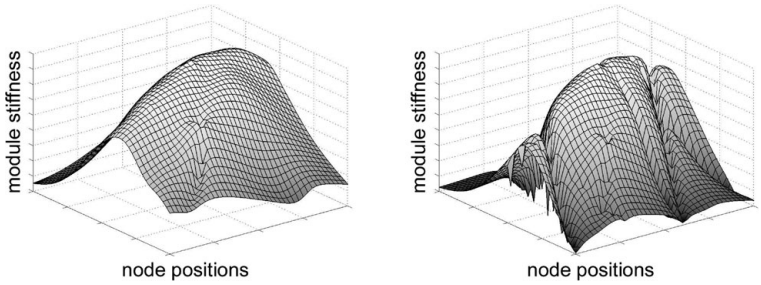

Fig. 7. Samples of the stiffness function of a given topology for (left) homogeneous struts and (right) struts that vary in stiffness as a function of the angle. In these plots increased stiffness is upward: the gradient is ascended in optimization.

3.4. Training

The learning objective set for the SVM is to predict the optimal geometries of a structure **y** given different load conditions **x**. Each training example is an optimal solution found by the iterated GD algorithm in which each sample is a finite element analysis of the structure. The greatest computational cost is therefore in generating this training data, and so the proposed learning method is online, with a gradually increasing data set rather than as a batch process.

Training of the SVM was performed with the gradually increasing set of stress vectors **x** and nodal points **y** until the accuracy of the learned function no longer increased. A radial basis function kernel with variance $\sigma^2 = 0.2$ was used to map all dimensions in the SVM.

3.4.1. Error estimation

Methods of error estimation have been systematically evaluated in Reich and Barai (1999). The method used, holdout, is the most conservative, in that it maintains a pessimistic bias toward the results.

The data D are divided at random into two sets: a training set T and a separate validation set V. The SVM is trained on T, and then evaluated on V, the errors in V indicating the generalization error. For D of size n, the size of T is ideally $0.6n$ to $0.8n$ and V is the remaining $0.2n$ to $0.4n$. Although there are no general bounds for regression, the data D of size $n > 1000$ produce results with confidence more than 0.95 in classification problems (Reich & Barai, 1999).

The following tests conform to these recommendations for accuracy. The performance of the SVM was evaluated for training sets of varying size, to a maximum size $n = 1300$. For all tests, the validation set V was a randomly selected set of size 300. The size of D for which the SVM will be considered in the tests to be fully trained occurs at $n = 1000$, which is the recommended size for 0.95 confidence, and errors for even smaller training sets have the most pessimistic bias of any estimation method. The results therefore display the worst-case estimation of errors, and the true accuracy of the algorithm is likely to be no less than is reported in the following sections.

4. THE TRAINED ALGORITHM: RESULTS AND ANALYSIS

The structures produced by the SVM are compared in this section to those found by a standard benchmark of GD for the same problem definition. The stiffness and stress of individual modules is examined first, followed by the performance of a composite structure of many units. The two optimization methods are also compared in terms of speed.

To evaluate the results of learning, an SVM was trained on an increasing set T of samples (from 1 to 1000) while being tested against a separate validation set V of 300 samples. In the following plots, this performance is evaluated both in terms of how similar the solutions given by the SVM are to the ideal solutions on which it was trained, and how well those solutions actually perform when tested under given loads. Under both criteria learning was seen to improve steadily with an increasing training set until performance plateaued at a very high level.

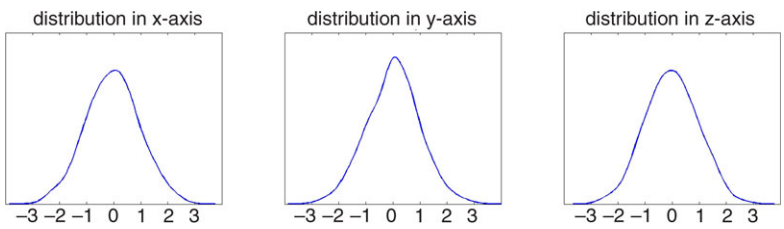

Fig. 8. Probability distributions of the x-axis, y-axis, and z-axis components of input force vector **x** are based on a normal distribution with mean 0. [A color version of this figure can be viewed online at www.journals.cambridge.org]

https://doi.org/10.1017/S0890060407000327 Published online by Cambridge University Press

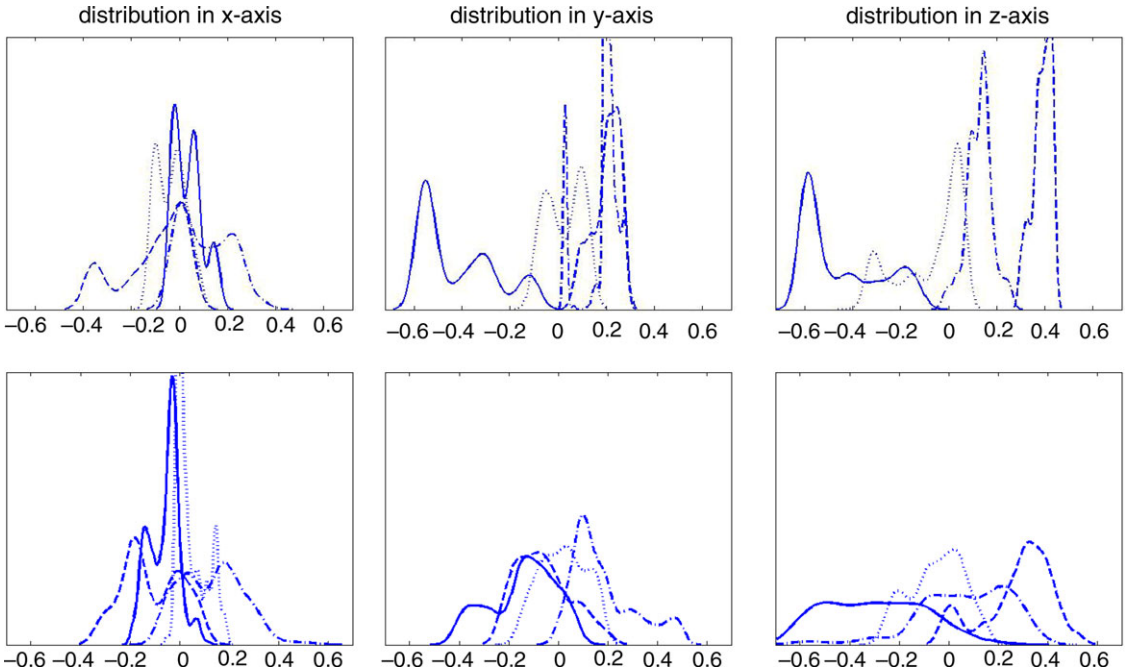

Fig. 9. Probability distributions of the *x*-axis, *y*-axis, and *z*-axis components of the nodal points **y** are asymmetrical in physical space, the result of optimization; (top) model A and (bottom) model B. [A color version of this figure can be viewed online at www.journals.cambridge.org]

4.1. Accuracy of the learned function

The performance, or error θ, of the algorithm trained with output **y** consisting of a single component y is often measured as a square loss function

$$\theta = 1/n\left[\sum_{i=1:n}(y_i - f'(\mathbf{x}_i))^2\right], \tag{7}$$

where n is the number of samples in the validation set V (Reich & Barai, 1999). Because the output vector **y** is 12-dimensional, this can be generalized to

$$\theta = 1/n\left[\sum_{i=1:n}\left(\sum_{j=1:d}|y_{ij} - f'(\mathbf{x}_i)|^k\right)^{1/k}\right], \tag{8}$$

where d is the dimensionality of the output vector **y** and k is the exponent of the metric. The choice of $k = 2$ (squared Euclidian distance) is appropriate for measurement of error in physical space, or $k = 1$ (the Manhattan or city block metric) is suited to independent parameters. Because the data in **y** are a combination of both—independent points in physical 3-space—the Euclidian metric of $k = 2$ has been used.

This error θ then is simply the mean distance between the nodes in each of the ideal samples **y** and the nodes in the corresponding solution $\mathbf{y}' = f'(\mathbf{x})$ output by the SVM. Distance here is measured as the sum of squared differences in each dimension of the 12-dimensional output vectors. The graphs in Figure 10 display the accuracy of the predicted nodes during training with an increasing set T of examples and a separate validation set V of 300 examples from model A and model B. Both indicate a steadily decreasing error for training sets T up to a point (indicated by ○), after which point there is little further perceptible change.

The number of training examples required (650 and 275) appear to be a result of the particular data sets, rather than inherent in the algorithm, and it is likely the required size of training set T would vary for different structural topologies. There is negligible variance in the resulting error θ when a different randomly selected set T is used in the SVM, or in the order of samples presented in training. Although the observed plateau beginning after this point in training does not coincide with an error θ of zero, it should be noted that both the generalization of the model $f'(\mathbf{x})$ and the pessimistic bias of holdout estimation will ensure a lower limit on the error. Training set sizes of 650 and 275 are likely simply to be the limits of learning for this problem.

The average accuracy of the functions at this point is approximately 0.025 units to the validation set, equal to the smallest possible manufacturing tolerance for a unit cube of 2 mm. At this stage the function of optimal geometries as provided by GD can be considered, for all practical purposes, sufficiently learned.

4.2. Performance of the predicted geometries compared to GD

Although the previous plots indicate the standard method of evaluating the accuracy of the function in terms of node distances, it is more relevant to our purposes to know how well the structures perform under their respective stresses. This can be determined for a given structure in the validation set

https://doi.org/10.1017/S0890060407000327 Published online by Cambridge University Press

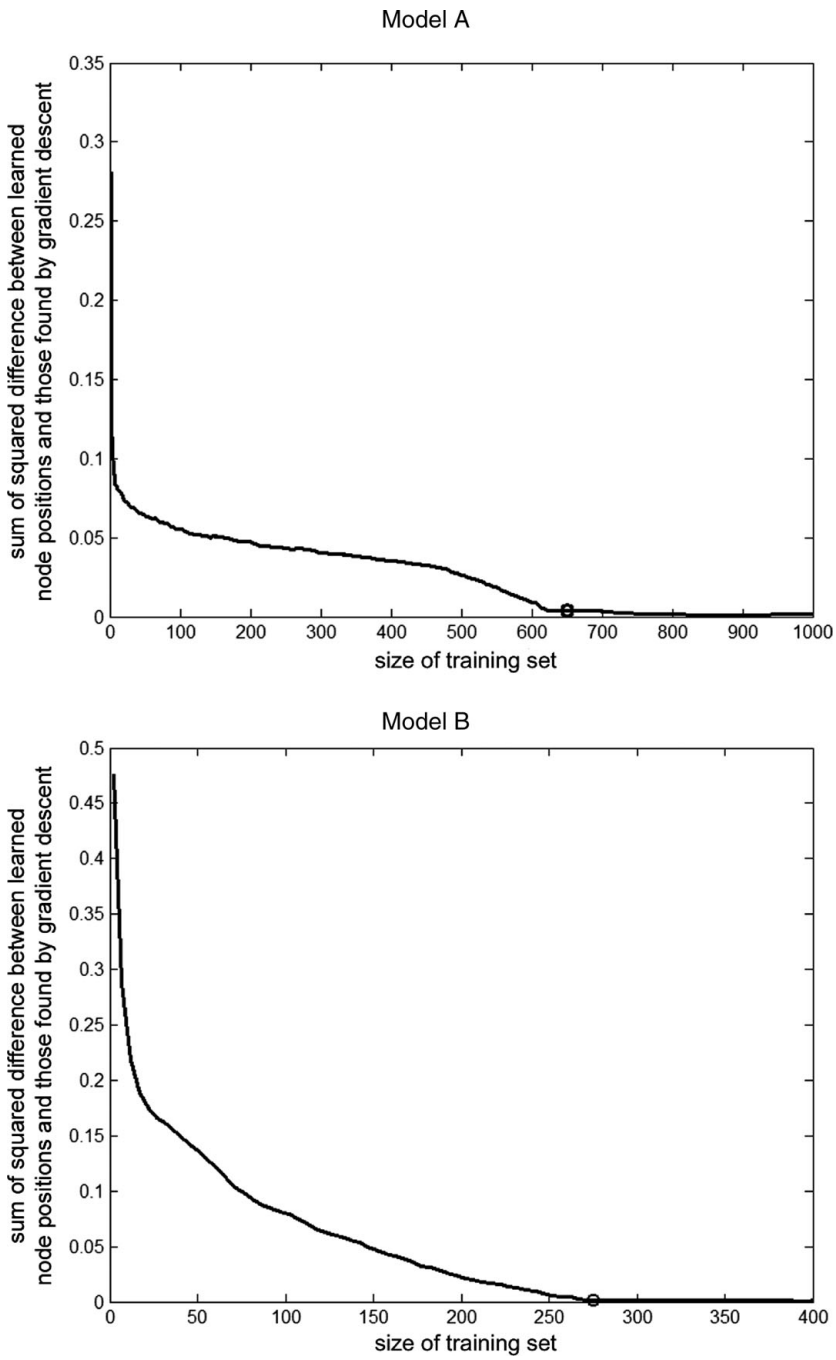

Fig. 10. The accuracy of learning increases with increased training.

by performing a finite element analysis on both the geometry $\mathbf{y}$ found by GD and the predicted geometry $\mathbf{y}' = f'(\mathbf{x})$ as found by the SVM. Both are loaded with the same input vector of stresses, and their stiffness under this load condition is measured as a total displacement of nodes when the load is applied.

This displacement between the original nodal points $\mathbf{y}$ and the resulting positions $\hat{\mathbf{y}}$ under simulated load is thus given by

$$\text{disp}(\mathbf{y}, \hat{\mathbf{y}}) = \sum_{i=1:m} [(y_i(x) - \hat{y}_i(x))^2 + (y_i(y) - \hat{y}_i(y))^2 + (y_i(z) - \hat{y}_i(z))^2)]^{1/2}, \quad (9)$$

where m is the number of nodal points and the performance (δ) of the predicted structures $\mathbf{y}'$ is estimated as the average ratio of displacements (disp),

$$\delta = 1/n \left[\sum_{i=1:n} (\text{disp}(\mathbf{y}, \hat{\mathbf{y}})/\text{disp}(\mathbf{y}', \hat{\mathbf{y}}')) \right], \quad (10)$$

where n is the number of samples in the validation set V.

Figures 11 and 12 (top) plot this performance δ of the predicted structures $\mathbf{y}'$ against the same validation set as in Figure 10. A ratio of 1.0 would indicate the predicted

https://doi.org/10.1017/S0890060407000327 Published online by Cambridge University Press

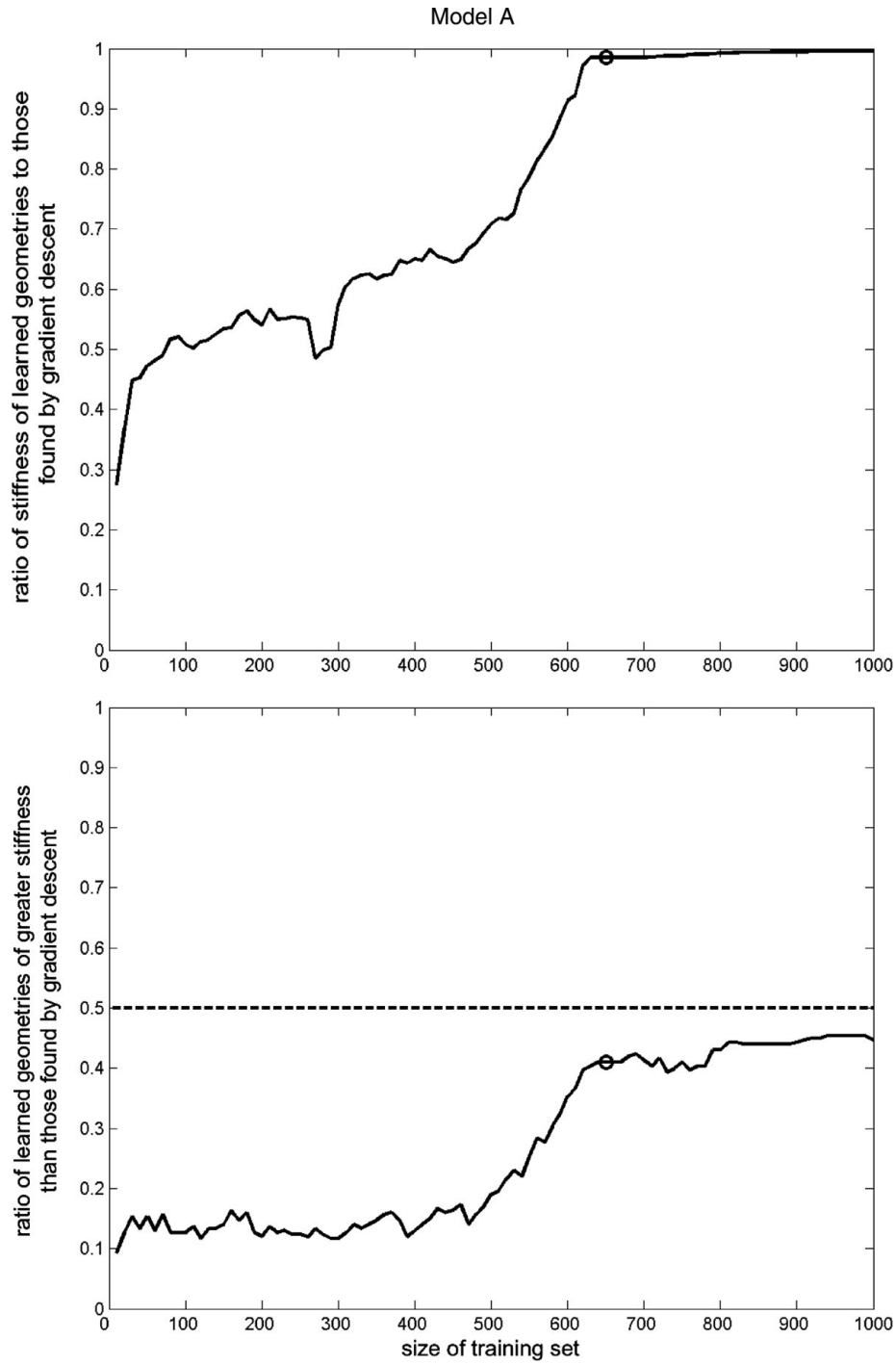

Fig. 11. The performance and learned improvements over the training set.

structures perform (on average) as well as those found by GD. As learning progressed, some predicted structures actually performed better than their equivalents as found by GD, and the ratio of number of structures of greater stiffness found by learning is shown in the lower plots of Figures 11 and 12.

In model A, improvement with an increasing training set is evident over the same range of 1–650 samples, with a mean stiffness ratio of nearly 1.0 (equal to the GD validation set) when the function is fully learned. At this point the percentage difference between the resulting displacement of the original samples and the predicted geometries had dropped to 1.51%. The number of structures with lower deflection than their GD counterparts also increased steadily over this range. Where 50% would represent the maximum expected value of a perfectly learned function on data with noise, the SVM approaches

https://doi.org/10.1017/S0890060407000327 Published online by Cambridge University Press

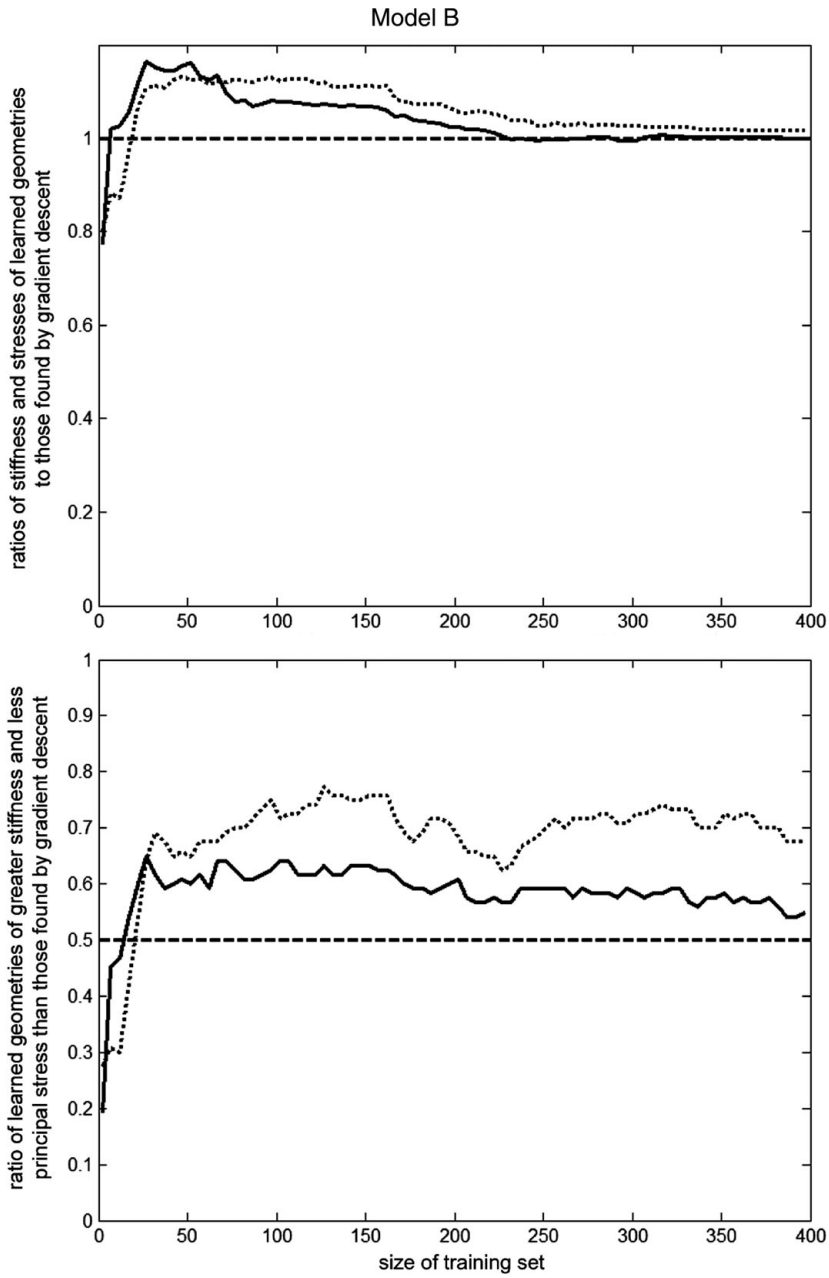

Fig. 12. The performance and learned improvements over the training set; (—) deflection and (···) principal stress.

this value at a training set size of 650 with 42% of structures having greater stiffness than the supposed ideal set.

The fact that many, or indeed any, structures can outperform the optimal structures in the data from which the SVM was trained can be explained by the method in which the data were generated. GD as a search method is itself prone to error because of local optima in the fitness landscape, and is thus not guaranteed to find the globally optimal solution. Although it has been shown to be an appropriate method for solving the structural shape optimization problem, it can only do so within an acceptable variance in node positions (Haroun Mahdavi & Hanna, 2004). It is this variance that causes some of the optimized geometries in the training and validation sets T and V to fall slightly below the true optimal solution. It can be considered equivalent to noise in a set of examples collected from real-world measurements. In avoiding overfitting, the regression process performed by the SVM effectively "smoothes out" the function learned so that some of these optimized structures lie either side of the function $f'(\mathbf{x})$. Thus, although the learned function may not be accurate enough to predict the exact node positions in the validation set, in these cases this actually is an advantage, providing an even stronger, more optimal structure.

This is even more evident in the results of learning in model B (Fig. 12), in which the noise introduced by local optima is greater. Again, the mean stiffness of learned structures

https://doi.org/10.1017/S0890060407000327 Published online by Cambridge University Press

is approximately equal to those found by GD after the function is fully learned at 275 samples, but, for the range of training set sizes below this and beginning at approximately 15 samples, the overall stiffness of the predicted structures is actually greater than those in the set on which the SVM was trained. The same is true when principal stress rather than deflection is taken as the measurement of performance, plotted in the dotted lines of Figure 12.

As learning progresses in a noisy data set, the closest approximation to the function generally occurs before overfitting, at which point the error θ of an independent validation set increases. Unusually, this did not happen for the validation error θ (Fig. 10) but appears to be evident in the performance curves, which peak at about 50 training samples for stiffness and 120 samples for stress. At this point the learned function is roughly 10–15% better than GD optimization before declining to equal that of GD. This would appear to be a result of the fact that the noise in the data is not random and uncorrelated, but a result of local attractors in the fitness landscape found by a deterministic GD search. Because the entire data set was generated by starting this search at the same point in the search space these same local attractors would recur many times in the data, rather than cancelling one another out as would random noise, and would be learned by the SVM as part of the function. This occurs at 275 samples where performance is seen to equal GD, but it is evident that the optimal performance is found by earlier generalization with a much smaller data set. Two practical observations can be drawn from this: that models with more noise because of GD are likely to be approximated more effectively by using fewer samples, and the performance of learning should be evaluated by testing the structures rather than measuring error directly on the function.

4.3. Assembling a modular structure

In addition to the ability of the learned function to outperform some of the structures optimized by GD, there is a secondary benefit offered by this smoothing that effects a composite structure formed of many unit cubes. The ideal situation for a complex arrayed structure (as described in Section 1.1) is that stress conditions change gradually and continuously over its volume, and adjacent unit cubes under similar stresses will have a similarly shaped structure. With any optimization process applied to individual unit cubes the variance in accuracy, or noise, will cause changes in node position or strut width to be more abrupt between some adjacent cubes. The repeated optimization of many separate structures amplifies the discretization caused by the initial sampling of the unit stresses, and these abrupt transitions result in weak points in the overall structure. By using the learned, continuous function to derive the structural geometry, the transitions between adjacent cubes are smoother, and the composite structure benefits in overall stiffness. As shown in the plots in Figures 11 and 12, some predicted structures perform better than what would be found by GD in instances where GD results in suboptimal local optima. This section tests the overall performance of an array of these against a benchmark solution found by GD.

The design problem is a simple cantilever like the examples shown in Figure 4, for which modular GD has been found effective in increasing stiffness (Hanna & Haroun Mahdavi, 2004). The overall distribution of loads throughout the object was first determined using FEM on a regular grid of eight-node isoparametric volumes to estimate the local relative displacements in each axis for a homogeneous material, and these then taken as the local loads $\mathbf{x}$ with which to optimize the structure of each unit cube. The positions of the nodes $\mathbf{y}$ were found by GD, and also using the function $\mathbf{y}' = f'(\mathbf{x})$ estimated by the SVM using training set derived from model B. A training set of 50 samples was used, as this was seen (Section 4.2) to result in the best performance of individual structures. These units were reassembled into the original cantilever and analyzed by FEM under the original applied load.

As only the node positions are learned, the base solution for comparison has been optimized by GD for these, but member section sizes within a cantilever structure are constant. The force applied to the end nodes has been scaled so that the mean displacement of nodes in this structure is one unit. For comparison, another structure was assembled such that the strut section sizes in a given unit are scaled in proportion to the magnitude of the local vector $\mathbf{x}$, an approximation to distribute more mass where stresses are higher resulting in a stiffer overall structure. All structures have the same overall mass.

The overall assemblies found by the SVM outperformed the GD optimized versions in each case, as shown in the list of mean and maximum deflections in Table 1. For structures of uniform strut thickness, SVM learning provides a slight improvement of 5% in mean deflection and 16% in maximum deflection, but this is far greater (approximately 80%) when the section sizes are scaled. A greater improvement can be made by changing the data set D on which the SVM is trained. Rather than using the generic normal distribution of input vectors $\mathbf{x}$ as in Section 3.3, the final row of Table 1 shows the results of training the SVM with samples taken from the same distribution as the local force vectors within the cantilever, which covers a more restricted range. The resulting learned function was thereby tuned to the particular problem and deflections were reduced over 90%.

Table 1. *Performance of modular cantilever beams produced by gradient descent (GD) and support vector machine (SVM)*

	Uniform Strut Sections		Sections Scaled to $\|\mathbf{x}\|$	
	Mean Def.	Max. Def.	Mean Def.	Max. Def.
GD	1.000	2.186	0.500	1.080
SVM 50 ex.	0.949	1.833	0.111	0.215
SVM 50 ex. in cantilever distribution	0.216	0.443	0.039	0.079

https://doi.org/10.1017/S0890060407000327 Published online by Cambridge University Press

These are far greater than the small improvements in individual structures in Section 4.2. The greatest benefit to the overall combined structure appears to be the continuity of the learned function compared to the noise introduced by GD. The structure benefits by a continuous functional estimation by producing a more gradual transition between adjacent unit cubes, avoiding potential weak points caused by recombining individually optimized structures.

4.4. Computational efficiency of learning compared to GD

Although GD is not a computationally demanding method of optimization in comparison with simulated annealing or GAs, the iterative sampling of gradient does require a large number of FEM analyses. The use of the learned function provides greater efficiency. The typical GD optimization implemented in MATLAB and using OpenFEM for analysis (Section 3.1.1) required approximately 1000 such analyses and 58 s of computation time for each unit of structure. Training and evaluation time of the LS-SVM increases with the square of the number of training samples, but thereafter evaluation time increases only linearly with the number of evaluations. For a training set of 50, as used in the previous sections, total training time is 0.062 s and evaluation time is 0.0012 s for each structural unit, for a total time of far less than even one unit optimization by GD. For a structure of 51 units in which 50 runs of GD are required for training there is already some saving of computation time, but this method is intended specifically for much larger structures. In the time required to optimize the next unit for a total number of 51, over 48,000 units can be computed using the SVM.

Improved methods of optimization may be more efficient or more accurate, with improved accuracy resulting in less noise in the function to be learned and requiring a larger training set for optimal performance. Even with a training set of 650 samples, total training time is 16.5 s and evaluation time is 0.0037 s for each unit of structure, still negligible in comparison to optimization by GD.

4.5. Summary

This work was aimed principally at investigating whether machine learning algorithms, in particular SVMs, could accurately predict the optimal geometries of structures, and thus be used as a substitute for a traditional optimization algorithm. An SVM was trained on example structures that had been optimized for stiffness, and used to predict structures that were compared against an independent validation set. GD was chosen as a benchmark optimization method for comparison because it is well understood and has been previously shown to be effective in similar problems. Several conclusions can be drawn from the observations:

1. *The accuracy of the learned function approaches that of the GD optimization*. Although the learned function is not as accurate as GD for optimization, it does come close, with minimal error θ after several hundred training samples (Section 4.1). At this point the function error was within the manufacturing tolerances of the structure and can thereby substitute as equivalent to GD.
2. *The learned function is quicker for optimizing larger structures*. Although essentially equivalent in output, the SVM is thousands of times more efficient in terms of speed (Section 4.4). Finding an optimal structure based on the learned function is far quicker than performing a full optimization via GD, as each sample of the latter requires a full finite element analysis, and one sample must be made for each dimension to calculate the gradient at each step. The production of the data set for learning is time consuming, but only in that a set of optimizations must be carried out in advance. In the example cases, from 50 to 650 fully optimized examples were required to learn the function at the outset. Many structural problems require the optimization to be performed only once, but for those in which a similar structural optimization is needed repeatedly, as in the case of an object composed of many modular units, the initial investment in learning the function of optimal geometries is justified. As this method of optimization is meant to be scalable to objects of thousands of modules, the learned function represents a substantial advantage in speed.
3. *The learned function results in a smoother and stiffer overall structure*. The discontinuities in an array of modules caused by separate optimizations are a disadvantage to the performance to the structure as a whole. The continuous function learned by the SVM ensures that the changes in the field of stress vectors are met with a similarly continuous change in structure. Overall performance is therefore improved by a greater degree than performance of individual modules in isolation (Section 3).

The two structural models A and B differed in terms of complexity and constraints. The smooth search space of the first resulted in a function that, once learned, appeared to predict the same optimal structures as GD. More unexpected of the findings was that in generalizing from the examples presented of model B, the learning algorithm was often able to outperform the original optimizations on which it was trained. The characteristic peak in the performance curve indicates overfitting with increased training that is expected in noisy data, but the fact that it appeared only in structural performance measured by stiffness or strain suggests that the noise is not random, but inherent in the function that generated the data in the first place: the space searched by GD. The fact that structures outperformed those optimized by GD at this point indicates that the learned function was better able to approximate the true underlying optimal function even with noisy data.

It should not be assumed, however, that this function is truly optimal. The low numbers of training samples used to achieve it (50–120 vs. 650 for model A) indicate that it may still be a rather rough approximation, and could be im-

https://doi.org/10.1017/S0890060407000327 Published online by Cambridge University Press

proved by a more thoroughly optimized data set. Beginning the search at random start points, running GD for longer (it was terminated at 110 iterations), or using other optimization methods entirely would be likely to improve it. It can be supposed from the number of training samples required for model A that such improved data sets would take longer for SVM training, but a closer approximation to the true optimal would be worth the small computational cost.

5. CONCLUSION

For well-defined structural problems in which the environment and topology are constant and the loads quantified by a continuous valued vector, it has been possible to learn the function mapping local vector to optimal structure. The results compared favorably to traditional optimization both in quality of solutions and speed. Rather than optimization by repeated sampling and evaluation of a physics-based model, it is thus possible to make design decisions for this structural problem based entirely on learning from previous examples. The principle motivation of such a process is greatly increased computation speed, but two secondary benefits also appear evident: that empirically measured performance can be easily incorporated without explicit analysis, and that the approximation made in learning can improve the performance of the structure by generalizing a continuous function of geometry change.

The adjustment of material properties because of strut angle stands in for real-world noise and complexities, and could be extended to incorporate physical manufacturing inconsistencies, geometry changes because of resin pooling, and other details not normally included in the finite element model. These are not associated element properties at the resolution of the strut model, but may still be a function of relative point positions. They can therefore still be learned and improvements made over the GD. It is important that these can be incorporated directly into the same function whether they are derived from analytical methods or empirical measurement.

This dual nature of data sources indicates the two directions for future work. The first is the improvement of the analytical optimization model itself. More variables can be introduced to the output vector, including the diameter of struts but possibly extending to other section shapes or material properties. The FEM representation of the single module can be further refined, and homogenization optimization models (Bendsøe & Sigmund, 2003) for the whole structure may be applied to improve the current assumption on the primary distribution of stresses. The data sets themselves may be improved by beginning optimization at varying points or using stochastic search to improve the distribution of local optima in the learned function. The second direction requires more empirical testing. In principle, the model learned can take on any degree of material or geometrical complexity because of microstructure or manufacturing, even when the details are not known explicitly. In practice, the degree to which these can be expressed as a function of points and other input variables deserves further research.

These refinements would serve to reinforce the benefits to speed and performance that appear evident in the initial models presented here. Once trained on successful precedents, the machine, in a sense, knows intuitively what works based on its prior experience, and can then predict optimal structures that rival or even exceed the initial training set in stiffness. This is a result not of strict analysis, however, but of inductive learning.

6. QUESTIONS AND ANSWERS

6.1. Question 1

"If I understand that right, you optimize the topology using the GA and then you optimize the geometry (shape) and section sizes. My question is what is the limitation of doing that and have you compared that with optimizing topology, geometry and section sizes simultaneously?"

6.2. Answer 1

"The initial GA uses a generic set of test loads to calculate the fitness of a topology, so there is a theoretical limitation in the second stage in that the globally optimal structure for a given local load might be missed because it requires a different topology. This is outweighed, however, by the advantage of modularity in the repeated topology. If all three aspects were optimized simultaneously for the local loads on each structural unit, this topology would change, and the points at which differing topologies connect would become weak points within the overall structure. An extension of the method in which differing topological zones can be connected is a possible direction for further research."

6.3. Question 2

"Also, you used gradient descent methods that optimize only locally. Have you looked at global optimization algorithms?"

6.4. Answer 2

"Global optimization is the standard on which this method improves. For structures with the total number of members that we are considering, the computation time for global optimization is prohibitively long. The technique of local optimization that we have developed approximates the global solution, but at a manageable speed. It is this approximation that then also allows inductive learning to be applied."

6.5. Question 3

"Do you think that you could use off-the-shelf software to do some of the analysis?"

6.6. Answer 3

"Finite element analysis is performed by a wide range of off-the-shelf software, and any package that is sufficiently

https://doi.org/10.1017/S0890060407000327 Published online by Cambridge University Press

flexible to allow the analysis to be driven by an external optimization loop is applicable. Most of the analyses described in this paper were performed with OpenFEM, an open-source finite element toolbox accessed via MATLAB."

6.7. Question 4

"Also, the optimization, is that completely manual at the moment?"

6.8. Answer 4

"All aspects of optimization were automated and based on digital finite element models. The topology was optimized using a GA. Using this result, sample structures were generated for a random set of load requirements. Geometry optimization was performed by an iterative gradient descent algorithm."

ACKNOWLEDGMENTS

Initial experiments using structural model A discussed in this paper were performed in collaboration with Dr. Siavash Haroun Mahdavi, and their results were reported in Hanna and Haroun Mahdavi (2006). Dr. Joel Ratsaby, Prof. Bernard Buxton, and Prof. Alan Penn have also provided guidance and helpful suggestions in this work. This research has been sponsored in part by the Engineering and Physical Sciences Research Council, UK.

REFERENCES

Adeli, H., & Cheng, N. (1993). Integrated genetic algorithm for optimisation of space structures. *Journal of Aerospace Engineering 6(4)*, 315–328.

Arciszewski, T., & Ziarko, W. (1990). Inductive learning in civil engineering: rough sets approach. *Microcomputers in Civil Engineering 5(1)*, 19–28.

Astley, R.J., Harrington, J.J., & Stol, K.A. (1997). Mechanical modelling of wood microstructure, an engineering approach. *IPENZ Transactions 24(1)*, 21–29.

Bendsøe, M.P., & Kikuchi, N. (1988). Generating optimal topologies in structural design using a homogenization method. *Computer Methods in Applied Mechanics and Engineering 71(2)*, 197–224.

Bendsøe, M.P., & Sigmund, O. (1999). Material interpolation schemes in topology optimisation. *Archives of Applied Mechanics 69*, 635–654.

Bendsøe, M.P., & Sigmund, O. (2003). *Topology Optimisation: Theory, Methods and Applications.* Berlin: Springer.

Chen, Y.M. (2002). *Nodal based evolutionary structural optimisation methods.* PhD Thesis. University of Southhampton.

Duda, R.O., Hart, P.E., & Stork, D.G. (2001). *Pattern Classification*. New York: Wiley.

Duffy, A.H.B. (1997). The "what" and "how" of learning in design. *IEEE Expert: Intelligent Systems and Their Applications 12(3)*, 71–76.

Hanna, S. & Haroun Mahdavi, S. (2004). Modularity and flexibility at the small scale: evolving continuous material variation with stereolithography. In *Fabrication: Examining the Digital Practice of Architecture* (Beesley, P., Cheng, W., & Williamson, R., Eds.), pp. 76–87. Toronto: University of Waterloo School of Architecture Press.

Hanna, S., & Haroun Mahdavi, S. (2006). Inductive machine learning of microstructures: estimating a finite element optimisation using support vector machines. In *Design Computing and Cognition '06* (Gero, J.S., Ed.), pp. 563–582. Dordrecht: Springer.

Haroun Mahdavi, S., & Hanna, S. (2003). An evolutionary approach to microstructure optimisation of stereolithographic models. *Proc. CEC2003, Congr. Evolutionary Computation*, pp. 723–730, Canberra, Australia.

Haroun Mahdavi, S., & Hanna, S. (2004). Optimising continuous microstructures: a comparison of gradient-based and stochastic methods. *Proc. SCIS & ISIS 2004, Joint 2nd Int. Conf. Soft Computing and Intelligent Systems and 5th Int. Symp. Advanced Intelligent Systems*, p. WE-7-5, Yokohama, Japan.

Kicinger, R., Arciszewski, T., & De Jong, K. (2005). Parameterized versus generative representations in structural design: an empirical comparison. *Proc. GECCO '05*, pp. 2007–2014.

Meiler, J., & Baker, D. (2003). Coupled prediction of protein secondary and tertiary structure. *Proceedings of the National Academy of Sciences of the United States of America 100(21)*, 12105–12110.

Murawski, K., Arciszewski, T., & De Jong, K. (2000). Evolutionary computation in structural design. *Engineering with Computers 16(3–4)*, 275–286.

Murdoch, T., & Ball, N. (1996). Machine learning in configuration design. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing 10(2)*, 101–113.

Neocleous, C.C., & Schizas, C.N. (1995). Artificial neural networks in marine propeller design. *Proc. ICNN'95 Int. Conf. Neural Networks*, Vol. 2, pp. 1098–1102 New York: IEEE Computer Society Press.

Pezeshk, S., & Camp, C.V. (2002). State of the art on the use of genetic algorithms in design of steel structures. In *Recent Advances in Optimal Structural Design.* New York: ASCE.

Ping, Y. (1996). *Development of genetic algorithm based approach for structural optimisation.* PhD Thesis. Singapore: Nanyang Technological University.

Quinlan, J.R., (1993). *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann.

Reich, Y. (1997). Machine learning techniques for civil engineering problems. *Computer-Aided Civil and Infrastructure Engineering 12(4)*, 295–310.

Reich, Y., & Barai, S.V. (1999). Evaluating machine learning models for engineering problems. *Artificial Intelligence in Engineering 13*, 257–272.

Schoenhauer, M. (1996). Shape representations and evolution schemes. *Proc. 5th Annual Conf. Evolutionary Programming*, pp. 121–129.

Schwabacher, M., Ellman, T., & Hirsh, H. (1998). Learning to set up numerical optimisations of engineering designs. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing 12(2)*, 173–192.

Shea, K., & Cagan, J. (1998). Topology design of truss structures by shape annealing. In *Proc. DETC98: 1998 ASME Design Engineering Technical Conf.*, pp. 1–11 New York: ASME.

Sischka, J., Hensel, M., Menges, A., & Weinstock, M. (2004). Manufacturing complexity. *Architectural Design 74(3)*, 72–79.

Suykens, J.A.K., Van Gestel, T., De Brabanter, J., De Moor, B., & Vandewalle, J. (2002). *Least Squares Support Vector Machines.* Singapore: World Scientific.

Szczepanik, W., Arciszewski, T., & Wnek, J. (1996). Empirical performance comparison of selective and constructive induction. *Engineering Applications of Artificial Intelligence 9(6)*, 627–637.

Vapnik, V. (1995). *The Nature of Statistical Learning Theory.* New York: Springer–Verlag.

Von Buelow, P. (2002). Using evolutionary algorithms to aid designers of archictural structures. In *Creative Evolutionary Systems* (Bentley, P.J., & Corne, D.W., Eds.). San Francisco, CA: Morgan Kaufmann.

Wang, L.H., Liu, J., Li, Y.F., & Zhou, H.B. (2004). Predicting protein secondary structure by a support vector machine based on a new coding scheme. *Genome Informatics 15(2)*, 181–190.

Sean Hanna is a Research Councils UK Academic Fellow at University College London (UCL), and he currently directs the Bartlett Graduate School's MSc in Adaptive Architecture and Computation. His background is initially in architecture, and his application of design algorithms has included major projects with architects Foster and Partners and sculptor Antony Gormley. Sean's current research covers structural optimization and rapid prototyping, and computational methods for dealing with complex systems in architecture. He is on the advisory boards of two UCL spin-out companies developing these technologies. His publications address the fields of artificial intelligence, collaborative creativity, robotics, and the optimization of structures and materials.