Defining Implicit Objective Functions for Design Problems

Sean Hanna
Bartlett School of Graduate Studies
University College London
1-19 Torrington Place
London WC1E 7HB
+44 (0)20 7679 1661
s.hanna@ucl.ac.uk

ABSTRACT
In many design tasks it is difficult to explicitly define an objective function. This paper uses machine learning to derive an objective in a feature space based on selected examples of previous designs, thus implicitly capturing the features that distinguish that set from others without requiring a predetermined measure of fitness. A genetic algorithm is used to generate new designs, and these are shown to recognisably display the appropriate features. It is demonstrated that the range of relevant features and optimal solutions is easily varied in proportion to the examples selected to define the objective. Methods for improving the function for GA search are discussed.

Track: Real-World Applications.

Categories and Subject Descriptors
I.5.5 [**Pattern Recognition**]: Implementation – *interactive systems, special architectures.*

J.6 Computer-Aided Engineering – *Computer-aided design (CAD*

General Terms
Algorithms, Design, Experimentation.

Keywords
Architecture, design/synthesis, fitness evaluation, genetic algorithm, machine learning.

1. INTRODUCTION
The ability of evolutionary algorithms and related search techniques to explore a varied space of solutions with efficiency and often surprising innovation makes them useful tools for design [1]. This typically requires the explicit definition of a goal or objective function and so has been ideally suited to engineering optimisation tasks. For many design problems however, and particularly for those of great complexity, it is difficult to specify such a goal in advance.

GECCO'07, July 7–11, 2007, London, England, United Kingdom.

Design and creativity themselves, particularly in a social context, are often seen as processes of guided, but open exploration. Steels [17] has shown that effective languages can be generated without an external measure of quality by allowing robots to speak and evaluate each other in an environment, and Nehaniv and Dautenhahn [12] propose an algebraic framework for similar imitation and interaction. Such approaches have been incorporated into genetic algorithms by allowing the objective to change over time. Saunders and Gero [13] allow a consensus of a community of agents to guide this, and alternatively Maher and Poon [11] co-evolve both goals and solutions simultaneously. Alternatively, multiobjective optimisation methods such as Pareto optimisation [20] allow broader definitions of a goal by finding a range of possible solutions, but each single objective must be clearly defined.

The method presented here generates an objective function for a genetic algorithm (GA) based only on labelled or classified examples of previous designs, and thus requires no explicit statement of objectives. It uses supervised learning to reduce the dimensionality of initial parameters to a low dimensional feature space Φ, in which only the features relevant to the original classification apply. This space is related to the notion of the archetype as described in [6]. Fitness is then determined by a distance measurement within this space.

This is demonstrated in the design of office floor plans, in which the location and orientation of many desk units form a complex spatial configuration. It is a problem of sufficient complexity that no consensus on measures of fitness currently exists. In practical use the designer need not specify one, but simply classify plans. The grid representation is generic, and can just as well apply to other domains such as urban form. The representation of solutions, both in the genome and in fitness evaluation is described first below, followed by the method of constructing Φ. Methods for improving the mapping are detailed, and the method is tested to evolve solutions as defined by various sets of archetypal plans.

2. PLAN REPRESENTATION
Several methods were used in this research to represent the layout of desks within a plan, one of which is detailed here. These vary in the amount of constraints used and the corresponding size of the search space. All take an underlying orthogonal grid as their structure, and as such, only produce orthogonal arrangements as implemented, but could be used with plans that contain several non-orthogonal grids or in principal extended for use with hexagonal or other planning grids if required. The methods ultimately represent desk/chair units as a filled/void pair within

one grid square. These may be in one of four orientations, as well as completely empty or completely filled.

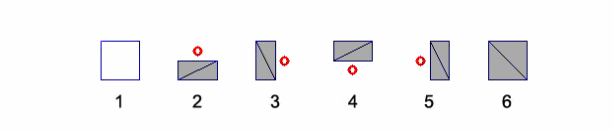

Figure 1. Possible desk types and positions in the grid.

Without constraints the search space is large: for an $n \times n$ plan grid the number of possible solutions is 6^{n^2} or roughly 10^{112} for even a small plan of n=12. Within this the majority are nonsensical arrangements and unusable spaces: large numbers of desks clumped together with no access or impassable groups blocking individual units. It is more efficient to assume the convex open area as the basic unit of space [7], and define this instead of individual desks. To enable access to all seats, it is assumed that desks are placed around the perimeter of a convex space with chairs facing toward the centre. Convex spaces may be connected together by the absence of desks to allow passage from one to the next.

2.1 Genome Representation

A parametric model is used to determine the shape, size and connectivity of spaces, but these can be repeated over any sized plan. Convex spaces are laid out according to a larger grid, its size and the configuration of desks within spaces determined by five distinct parameters:

di — the i-axis size of a space unit (integer 2 to 9)

dj — the j-axis size of a space unit (integer 2 to 9)

off — optional offset of every second row (integer 0 to 7)

C — distance along the side of a space for a passage (4 × 4 array of reals –1 to 1)

G — indices indicating which spaces to use from C (4 × 1 array of integers 1 to 4)

The five parameters are converted to binary values and assembled to form a single gene string for use in the GA. The real valued distances of *C* are approximated to five bits to ensure a resolution not less than that of the maximum length of *di* or *dj*. All parameters taken together form a 97 bit string which is operated upon using standard binary mutation and two point crossover. There is some redundancy in the encoding of four desk configurations in *C* when some may not be used, and in the unused, negative portion of the range of *C*, but the total size of the search space is no greater than 2^{97} or roughly 10^{29}. This is far smaller than the unconstrained search space and is constant regardless of the plan size.

2.2 Graph Representation of Spatial Features

In evaluating the phenotype, the parameters that will form the objective function are initially unknown, so we must begin with initial data that is as generic and comprehensive as possible. A method is required through which to encode all qualities of the space that may become necessary features in the feature space Φ. Two related techniques, *visibility graph analysis* [19] and *axial graph analysis* [7,18] have demonstrated the effectiveness of graph representations as a general approach for varied plan types.

Properties of visibility and axial graphs have been shown to be strongly related to both spatial perception and resulting behaviour of people within spaces. Strong correlations have been found with measures of visibility graphs and observed way-finding, movement and use in buildings [19] and urban pedestrian movement [3]. Axial graphs have likewise been shown to be closely related to directly observed movement [13,8], building use and social interaction [16], and indirect behaviour such as land values and crime [9].

As a generic evaluation technique, graph representations are both spatially relevant and capable of capturing necessary variation. Crucially, the graph is also very different from the genome representation. Three methods of graph representation are used that can be derived from the plan phenotype: an axial or visibility graph between seats, and two versions of a modified boundary graph – one for open spaces and another for desk groups. Common to all three is that the basic unit of spatial division is the desk or empty grid square.

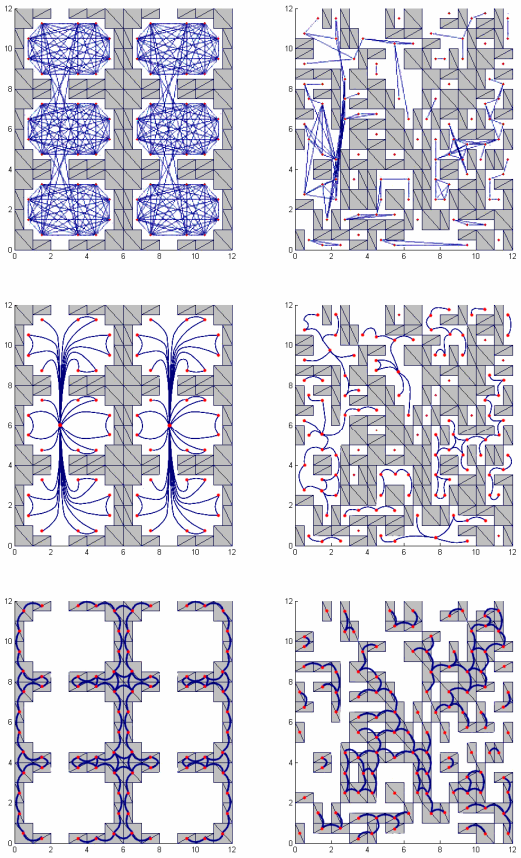

Figure 2. Three graphs derived from the plan: axial (top), boundary (centre) and desk graphs (bottom) are shown for two differing plans.

2.2.1 Axial / visibility graph features

An algorithm was implemented to generate axial (visibility) maps by drawing links between each chair node to every other chair

node that can be connected by an unobstructed, direct sight line through empty space. This differs from an axial line map defined by plan vertices [18], but the use of the set modules with chair points as the only possible nodes greatly reduces the computation time necessary to generate a graph. Graph nodes in this representation are not the lines themselves, but the chair points in the plan. Two examples of the resulting axial graphs are displayed in figure 2 (top).

2.2.2 Boundary graph features

The boundary graph is meant to capture all immediate or continuous spatial connections, regardless of sight. The basic node is still taken as the individual chair point, but empty space is also considered and links are drawn only between nodes that are directly face-wise adjacent. A chair point is connected to any other chair immediately behind or to the side or to an open space onto which it backs. All adjacent void spaces are then grouped together into one single node, regardless of size, representing all continuous space within the boundaries formed by the desks. Boundary graphs are shown in figure 2 (centre).

2.2.3 Desk graph features

Desk graphs capture the relationships between adjacent desks. They are generated by the same method as the boundary graphs, except the graph nodes are the desks rather than chairs, and these are connected to all other adjacent desks. The resulting graph will generally not be unified, but segmented into discrete sub-graphs each corresponding to a connected group of desks. The desk graphs for two plans are in figure 2 (bottom).

2.3 Measuring between graphs

Several approaches to similarity measurement have been based on small graphs of adjacency or connectivity of spaces in plan. Dalton and Kirsan [2] use the number of transformations necessary to derive one such graph from another to measure the similarity between buildings, and Jupp and Gero [10] suggest an analysis based on similarity and complexity measures of semantic graphs. With larger and more complex graphs as generated by axial lines, calculation of similarity becomes more difficult, but this can be overcome with graph spectral analysis. In image recognition applications, this has been used to effectively query a large database of logotype images for matches [14]. Spectral analysis of a graph uses the eigenvalues and eigenvectors of its connectivity matrix, and is relevant to the kinds of analysis considered here. The spectrum of a graph, or ordered set of eigenvalues, is useful in that it can be used to represent the graph as a single feature vector.

For any graph with a set of nodes V and a set of edges E, the adjacency matrix A is a $|V| \times |V|$ matrix defined by:

$$A(i,j) = \begin{cases} 1 \text{ if } (i,j) \in E \\ \quad \text{or} \\ 0 \text{ otherwise.} \end{cases} \quad (1)$$

The spectrum of the graph is defined as the set of ordered eigenvalues of this matrix, sorted such that any isomorphic graphs will have the same order of eigenvalues. Sorting by value, such that $\lambda^1 < \lambda^2 < \ldots < \lambda^{|V|}$, the spectrum will be a vector composed of the $|V|$ eigenvalues:

$$S = (\lambda^1, \lambda^2, \ldots, \lambda^{|V|})^{\mathrm{T}}. \quad (2)$$

A feature vector of dimensionality n is derived from this by performing a cubic spline interpolation of the $|V|$ values in S to a new vector

$$S' = (l^1, l^2, \ldots, l^n)^{\mathrm{T}}, \quad (3)$$

and thus all spectra may be mapped in the same n-dimensional space [5].

3. METHOD OF PLAN GENERATION

With such a measurement, and given an existing plan as a goal, fitness in the GA may be calculated as inversely proportional to the distance between a given plan's graph and that of the given prototype:

$$f(i) = 1 / \sqrt{\sum_{j=1:n} (S'_i(j) - S'_{\text{goal}}(j))^2} \quad (4)$$

where S' is the spectrum as given in (3).

When such a goal is impossible to determine explicitly, a group of plans may be selected as examples. As a simple case, one might desire a general arrangement of groups of regular, outward facing desks as in Figure 2 (left) as opposed to linear rows, inward facing groups or random clumps. The size of these groups may be irrelevant, as may be their position relative to one another. Such a design goal is admittedly straightforward to describe, and to evaluate visually, but does not appear explicitly in the spectral analyses. A group of existing plans may be selected that have this arrangement of desks, along with another that has an undesirable arrangement of rows.

As a source of plans, the *ClassPlan* data set was created manually from 128 permutations of five parameters. Three affect local desk arrangement:

variable 1: convex groups or linear rows
variable 2: groups aligned or staggered
variable 3: local length of desk groups

and two represent global configuration:

variable 4: space filled with desks or partially open
variable 5: global plan shape and size (12 × 12, 24 × 12, 12 × 24 or 24 × 24)

These plans were intentionally created always to slightly differ from those which can be created by the parametric representation in section 2.1.

The feature space mapping $\Phi(\cdot)$ is created in which to measure the plans:

$$\{z_1, \ldots, z_i\} = \Phi(S') = \Phi(\{ l^1, l^2, \ldots, l^n \}). \quad (5)$$

To do so, a feed-forward multilayer perceptron (MLP) was trained to classify two sets of 64 plans, one consisting of convex groups with a single target output of $z = -1$, and another of linear rows with a target of $z = 1$. The resulting mapping places the n-dimensional spectrum S' in a one-dimensional feature space Φ in which all convex group plans are clustered near a single point regardless of their other features of size, etc. The major consideration in the configuration of the network is the size of the hidden layer, which determines ability to discriminate or generalise, and was found to affect classification accuracy by up to 15% between 10 and 120 nodes. A hidden layer of 20 nodes was found by cross-validation to be the optimal configuration for

classifying the plans in *ClassPlan*, and was used for all subsequent experiments.

The fitness is calculated much the same as in (4), except the distance is measured not to an objective prototype plan in the full feature space of the spectrum but as a distance within this reduced feature space.

$$f(i) = 1 / \sqrt{(\Phi(S'_i) - \Phi(S'_{goal}))^2} \quad (6)$$

The goal is set either to the measured position of the mean of the ideal plans, or simply to the original target of z = ±1. With a trained network these values are similar; in the following example the means are –0.996 and 0.982, and so the targets –1 and 1 were used.

Figure 3 shows the results of two typical runs of the GA to evolve examples of a given class in the above feature space, either convex groups or linear rows. The algorithm has a population size of 40, a mutation rate of 4% and employed fitness proportional selection. Both results are recognisably of the target class but the designs evolved are novel: neither were present in the initial data set on which the MLP was trained.

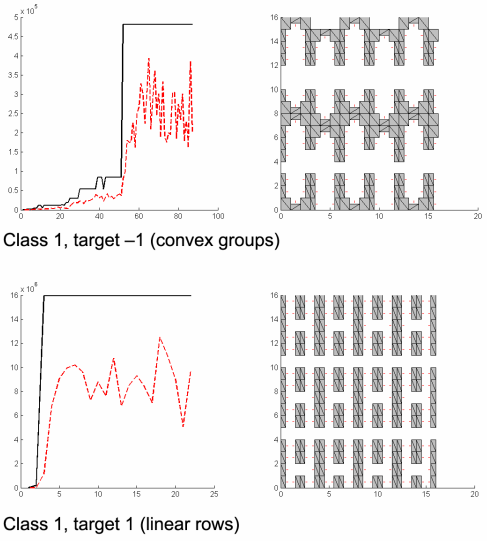

Figure 3. Plans produced by the GA for opposing objectives.

3.1 Clarifying the Search/Perception Space

An examination of the fitness over time in figure 3 (left) reveals in both cases an extreme jump in fitness after the first few generations after which no further improvement is made. For a GA to conduct an efficient search, it is desirable to have changes to the genome affect corresponding and ideally proportional changes to the fitness of the phenotype [1], but with an ideally trained two-class classifier that the MLP approximates, the response of all inputs within a class will be identical and equidistant from all examples of the opposite class. There is thus little variance in fitness unless the goal class has been achieved and no capacity to direct the search by fitness.

This section will examine ways in which the perceptual space in which fitness is measured can be made more amenable to search by GA, and deals with two issues of usability of such a network in actual design. Two methods are suggested for improving the *training* of the MLP to yield a more navigable fitness landscape. While a continuous fitness evaluation is desired, it is often easier for a user to make a binary classification than to assign accurate continuous values, so the first develops a continuously variable feature space from training on discrete targets by tuning the activation function $g(\cdot)$ to saturate at the targets during training. The second begins training from weights initialised to the principal components of the data. These are tested below.

3.1.1 Tuning the activation function g(·)

Alteration of a neural network between training and evaluation to access learned internal representations is employed in unsupervised learning to make evaluations not explicit in the training output. The removal of the output layers of auto-encoders, for example, can yield a principal or non-linear component analysis in a hidden layer [4]. A similar strategy is examined here to refine the feature space beyond the labels used in training. Classification of data toward discrete, opposing targets is appropriate given the construction of the *ClassPlan* data set and makes initial labelling easier for the user, but is detrimental to the later GA search. It is therefore desirable to allow the machine to be trained on a binary classification but make its evaluations on a continuous scale. It is the sigmoid activation function [4] in the MLP

$$g(net) = \alpha \tanh(\beta\, net) = \alpha\, (e^{+\beta\, net} - e^{-\beta\, net}) / (e^{+\beta\, net} + e^{-\beta\, net}) \quad (7)$$

that ultimately determines the placement of a sample in Φ, and the parameter α limits the range of activation values to $\pm\alpha$, where all extreme values saturate at the function's asymptotes. If the network is trained with such a function and then used in fitness evaluation with the linear activation

$$g(net) = \alpha\, net \quad (8)$$

in the output units only, this will disperse the values formerly saturating the asymptotes at $\pm\alpha$ over the entire linear range $\pm\infty$.

This can be enhanced by tightening the training function even further. A typical setting of $\alpha = 1.716$ [4] results in a $g(\cdot)$ that is nearly linear in the target range [–1, 1]. If $\alpha = 1.0$ is used so that $g(\cdot)$ saturates at the classification target values ±1, almost all values for $net > 0$ result in $g(net) \approx 1.0$ and values for $net < 0$ result in $g(net) \approx -1.0$. This produces a more continuous space for evaluation.

3.1.2 Validating the Continuity of Φ

Zhu and Wilson [21] measure the effectiveness of spectral representations by comparing sequential distances between spectra with known edit distances when creating the graphs themselves. Similarly, the continuity of the feature space Φ can be judged by comparing distances against changes to a known parameter in the plans. A new plan set, *SeqPlan*, was used, consisting of four groups of 100 plans, each of which varies monotonically in ten steps of two parameters. One of these parameters is local (number of desks to a group) and corresponds roughly to variable 3 in the *ClassPlan* set, and the other is global (averall plan size), corresponding to variable 5. Each of these four plan sets has a different general configuration of desks. As

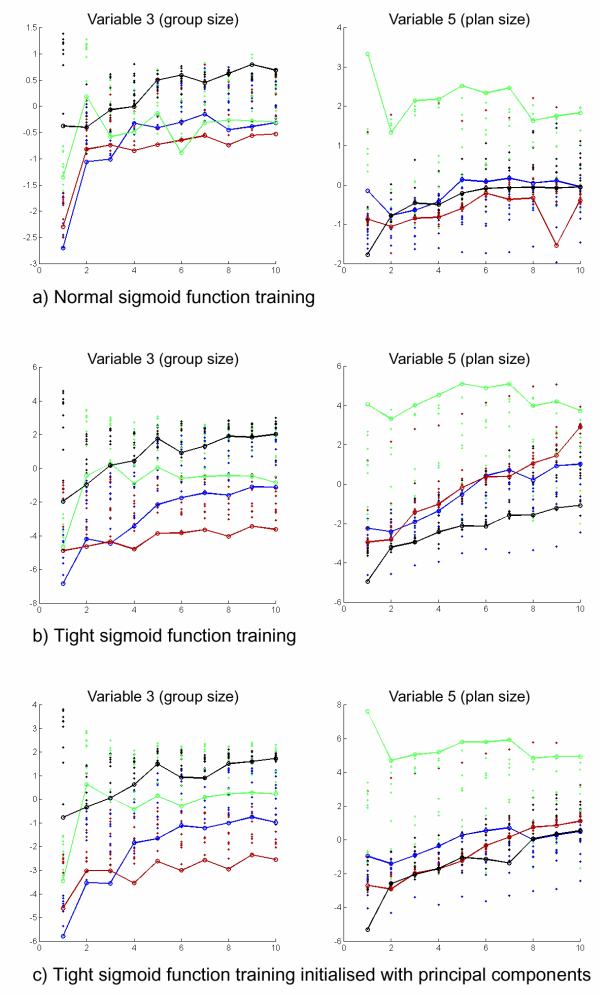

Figure 4. Uniform changes of parameters plotted in various feature spaces.

in Section 3, the mapping Φ(·) was created based on variables 3 and 5 of the ClassPlan set and then used to evaluate the corresponding features of the sequential data set. Ideally, even while trained only on opposing examples, the placement of sets of sequentially varying plans will still be incremental in Φ. The utility of the resulting space is then judged in terms of how closely linear the resulting output is.

Results of training with the 'tight' sigmoid of $\alpha = 1.0$ are displayed in figure 4b in comparison with a larger $\alpha = 1.716$ (figure 4a), and a general straightening of the sequences of plans is evident. The corresponding mean relative deviations are displayed in table 1. The overall relative deviation is seen to decrease by more than one third when trained with the 'tight' sigmoid $\alpha = 1.0$, reducing in all cases except for that of the plan type 3 (the lightest line on the plots).

Table 1. Mean relative deviations in Φ for various training methods.

	Random initial weights				Weights initialised from PCA			
	Normal training		Tight sigmoid [-1,1]		Normal training		Tight sigmoid [-1,1]	
variable	5	3	5	3	5	3	5	3
Type 1	.394	.410	.120	.375	.191	.363	.159	.351
Type 2	.468	.348	.133	.300	.205	.200	.170	.194
Type 3	.749	.571	.208	.750	.418	.581	.317	.590
Type 4	.453	.344	.166	.341	.311	.269	.257	.284
Mean	**.516**	**.418**	**.157**	**.441**	**.281**	**.353**	**.226**	**.355**
Method mean	**.467**		**.299**		**.317**		**.291**	

3.1.3 Initialising weights with principal components

An algorithm trained on the two class polarity of variables in *ClassPlan* must interpolate to a much finer degree in evaluating a complete set of ten sequential plans in *SeqPlan*. With the initialisation of MLP weights at random, it is possible that an inappropriate initial setting is not adequately 'unlearned' by the limited examples in the training set, and so the use of a decent first guess for these weights is compared to that of random initialisation.

The principal components of the data set by definition capture most of the variability of the examples, and their orthogonality ensures that there will be no redundancy in the network. PCA is performed on the training set to yield the set of eigenvalues λ and eigenvectors φ of the covariance matrix, which are sorted in order of decreasing magnitude of |λ|. The MLP is to be trained with 20 hidden nodes, so the first 20 $\{ \lambda_1 , \lambda_2 , \dots , \lambda_{20} \}$ and $\{ \varphi_1 , \varphi_2 , \dots , \varphi_{20} \}$ are used. The eigenvectors representing the orthogonal principal components are used for the hidden layer weights, such that the net activation net_j at a hidden layer node represents the projection of a given sample onto that component, and the eigenvalues are used for the output layer weights to weight the output in favour of the largest components. Because the projections can be quite large, and enter the saturation regions of $g(\cdot)$, these initial weights were scaled by 0.2 to maintain a *net* largely within the near-linear range of $g(\cdot)$. Each weight then is given by

$$w_{ij} = 0.2 \times \varphi^j(i) \quad (9)$$

$$w_{jk} = 0.2 \times \lambda^j \quad (10)$$

for the input and hidden layers respectively.

Again, the performance was measured in terms of relative deviation of the sequences in *SeqPlan*. The output of the network trained on the 'tight' sigmoid with weights initialised as above in (9) and (10) is displayed in Figure 4c and the results of training with both activation functions are listed in Table 1. The mean relative deviations are seen to decrease with the use of principal components as the initial weights, and this occurs for most of the sample sets in both classes. The best overall definition of Φ appears to be given by training with both the tight sigmoid and initialising the MLP with the principle components of the data set.

4. RESULTS

The method was tested first in optimising to particular objectives defined by a series of plans as in section 3, and then in attempting to match individual plans as goals within the feature space Φ. The GA was run with a population size of 40, each time for 200 generations, to produce plans constrained to a 12×12 grid. Fitness is calculated as inverse distance to a target in Φ as in (6). In all cases, examples of the *ClassPlan* set were used as the examples from which to derive the objective. It is relevant to note that because of the intentional mismatch between the genome representation and the *ClassPlan* set it is impossible to exactly

duplicate a particular ClassPlan example, so optimisation is intentionally restricted to approximating these plans.

4.1 Optimising to Learned Objectives

Although any labelling of example plans could be used to create the feature space Φ in which fitness is measured, the effectiveness of the algorithm is more easily judged when the goal is visually recognisable. The variables initially used to create the data set were used as objectives for the GA as each easily provides two opposing classes of 64 plans, but because global size was restricted to 12×12, only the local variables were used.

Figures 5, 6 and 7 show the results of five runs of the GA for objectives set to each of the extremes of variables 1, 2 and 3. Two example plans from the training set are shown at left to represent the objective defined by Φ, although these represent a much larger group of 64 plans of varying arrangement and overall size. At right the best solutions found are drawn with their distance to the goal indicated.

Each set of evolved plans can be seen to indicate the features common to the example plans: straight rows, small group size, etc., indicating both that the objective has been defined and that the GA was able to satisfy it. It is also evident that for each run the features not included in Φ are irrelevant. In Figure 5, the top row of plans contains only non-linear, convex groups of desks, but these range in size from small clumps of three or four (first plans at left) to large room like spaces of more than twenty (last plans on right).

Some variation exists in the apparent accuracy to which the GA meets the supposed objectives. Although these can only be judged empirically, this accuracy may be graded by counting each plan recognisably in the correct class as a value of 1, each one not as a value of 0, and those undecided as 0.5. Over a trial of ten runs, the statistical accuracy for each of the classification variables was judged to be the following:

variable 1. convex groups or linear rows:	95%
variable 2. aligned or staggered groups:	70%
variable 3. small or large groups:	90%

This appears to match the ability of the MLP to learn the distinction between the initial sets on which it was trained. The accuracy of the classification as judged by cross validation on these three variables was 100%, 92% and 93%, differing in value from the above figures, but ranking the variables in the same order in terms of clarity or distinctiveness in the feature space. As indicated by the small distances to the objectives in Figure 5, these inaccuracies in the aligned and staggered group plans are due to the difficulty in initial MLP training to this variable, rather than the GA optimisation. Of the three, this also is the most difficult feature for the human eye to discern, and it is likely that a different, perhaps larger, training set of example plans would be needed to improve the results.

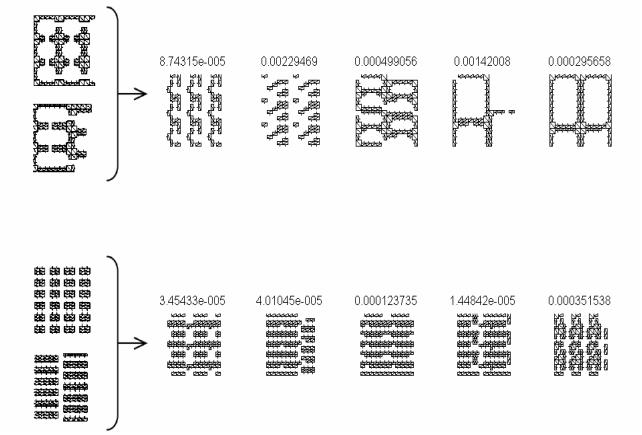

Figure 5. Results of the GA producing convex groups (top), or linear rows (bottom). Sample plans representative of the training sets are shown at left.

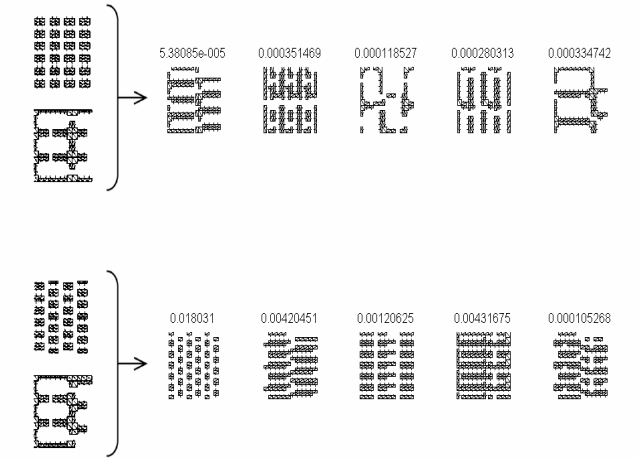

Figure 6. Results of the GA producing aligned (top), or staggered groups (bottom).

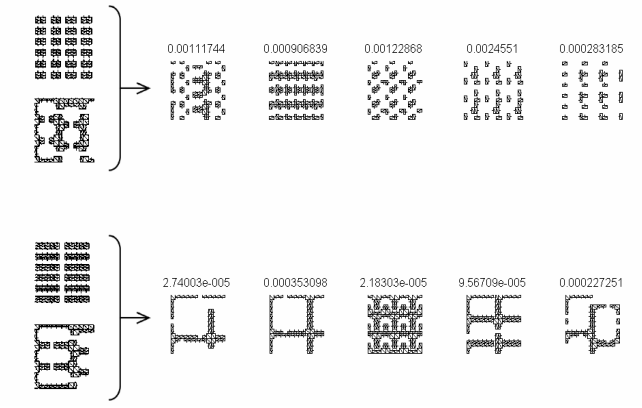

Figure 7. Results of the GA producing small (top), or large groups (bottom).

4.2 Matching Plans in *ClassPlan*

The above used a single labelling of plan groups to implicitly define an objective in a one dimensional space, but the same can be extended to a greater number of dimensions. The space may be created either by combining the dimensions of several training sessions of the MLP, or by using a label of several dimensions for each plan and training once. As before, the objective is represented by a goal point within this space Φ.

A test of the method in optimising toward a multidimensional objective was conducted by deriving goals from all five of the variables represented by the *ClassPlan* set simultaneously. For purposes of visual evaluation these objectives were set equal to a particular known plan. Fitness would be a simple comparison against the existing plan, with one crucial difference in that all evaluation is performed in the *learned* feature space: a space only of five dimensions rather than hundreds. All five variables were expressed in the feature space, for a five dimensional measurement of the objective function (6).

The optimisation was performed with targets set to match eight different plans in *ClassPlan*. Again, the three local variables were used to select these, and the selection of all permutations of the polar targets –1 and +1 for each of these variables in combination yielded the eight separate plans displayed in Figure 8 (left). The GA was run three times for each objective, and the results are displayed in Figure 8 (right), again with the distances in feature space of the resulting solutions displayed above.

A visual analysis of the resulting plans confirms the similarity of the matches to the targets chosen. The first variable, contrasting convex groups with straight rows is easy to see in the examples, and is clearly emulated in all 24 cases except one: an attempt to match plan 81 which produces large convex groups rather than short rows. The second variable contrasts aligned groups (plans 1, 4, 65, 68) with staggered (plans 17, 20, 81, 84). Although the novel group arrangements introduce some subjectivity into the validation, between 13 and 18 of the 24 results are members of the appropriate class. There is again some subjectivity in the judgement of the third variable of group length, as the 'short' example plans (1, 17, 65, 81) have two to three desks in a 'typical' line while the 'long' examples (4, 20, 68, 84) have five or six. If one attempts to find the typical length of a row of desks in any of the resulting plans and counts 2, 3 or 4 as 'short', and 5, 6 or 7 as 'long' then the number of plans that accurately meet their target is between 18 and 23 of the total 24.

Unsurprisingly, the inclusion of more criteria in the objective function results in a more difficult search than in section 4.1. Evaluated by the same method the statistical accuracies over 24 runs are slightly lower:

variable 1. convex groups or linear rows:	96%
variable 2. aligned or staggered groups:	67%
variable 3. small or large groups:	83%

Again the varying ability of the GA to produce plans to fit each of the classes correlates with the varying ability of the MLP to learn each of the variables as indicated by the cross-validation in sec. 4.1.

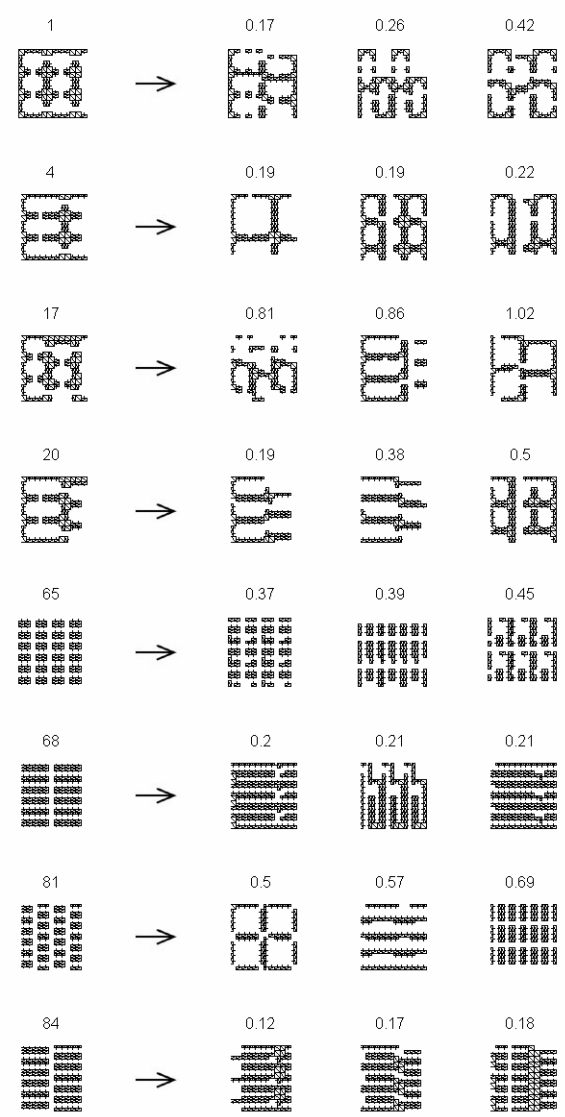

Figure 8. Results of the GA searching for a match to a plan in a multidimensional feature space Φ.

5. CONCLUSION

Rather than explicitly stating a measurement of fitness, this paper proposed that an objective function can be derived implicitly from a set of examples. A method has been developed to generate a space Φ in which fitness can be measured as a distance to a goal. Only the features relevant to the examples provided are expressed in this space, and so for an example set of given variance, the objective will have a corresponding variance. It has been shown how the space can be refined for easer search, improving the utility of the method for use in practice.

A GA was implemented with implicitly defined objective functions for an office plan problem, and evolved plans show the features suggested by the sample sets on which the mapping to Φ was created. At no point were these features ever stated explicitly as a goal, but only implied by the set of selected example plans. The similarity of the results to the features selected in Section 4.1

and to specific plans in 4.2 indicates both that the objective can be learned, and that it is useful in GA optimisation.

Because the reduced dimensionality of the feature space in which the fitness is measured only captures the relevant features, there may be many points in the full search space that are equidistant from any particular goal in Φ. Within this range there is a large and varied set of optimal solutions, not unlike the Pareto optimal set of a multiobjective problem. Increasing the dimensionality of Φ but maintaining a specific point as the goal effectively adds criteria to the objective, making it more specific. This is evidenced by the results of Section 4.2, in which the three plans shown for each objective when matching plans (Figure 8) in general appear far more alike than those for a given run of the single dimension GA in Section 4.1.

The classes chosen to define the objectives in Section 4.1 appear obvious to the eye, but are not clearly evident in the graph spectrum itself. The fact that they can be distinguished in this way indicates that arbitrary classes may be used, including far more subtle but relevant distinctions. The benefit of the method is that it could be extended to more complex problems in which the objectives are not evident to the user.

In many design situations where it is difficult to determine a measurement of fitness, this would provide a potentially more natural way for designers to work with setting objectives. These are still set precisely, but are implicit, not explicit. Experience and empirical observation of designs in the real world often indicates examples that perform well or poorly for reasons too complex to be fully evident even under close analysis, and such a definition of objectives is equivalent to saying 'make something like these'. The method has been demonstrated for the example of office plan layout, but applies to all domains of design.

6. ACKNOWLEDGMENTS

I would like to thank Alan Penn for helpful discussion and particularly on the analysis of plans. This work is supported by the Engineering and Physical Sciences Research Council, UK.

7. REFERENCES

[1] Bentley P: 1999, *Evolutionary Design by Computers. Morgan Kaufmann*, San Francisco.

[2] Dalton R C, and Kirsan C: 2005, Small Graph Matching and Building Genotypes. *Environment and Planning B: Planning and Design* (forthcoming).

[3] Desyllas, J and Duxbury E: 2001, Axial Maps and Visibility Graph Analysis, *Proceedings, 3rd International Space Syntax Symposium,* Georgia Institute of Technology Atlanta.

[4] Duda, RO, Hart, PE and Stork DG: 2001, *Pattern classification*. John Wiley & Sons, NY.

[5] Hanna S: 2007, Representation and Generation of Plans Using Graph Spectra, *Proceedings, 6th International Space Syntax Symposium.* (Forthcoming)

[6] Hanna S: 2007, Automated Representation of Style by Feature Space Archetypes: Distinguishing Spatial Styles from Generative Rules, *International Journal of Architectural Computing.* (Forthcoming)

[7] Hillier B and Hanson J: 1984, *The Social Logic of Space*. Cambridge University Press.

[8] Hillier B, Penn A, Hanson J, Grajewski T and Xu J: 1993, Natural movement, *Environment and Planning B: Planning and Design,* vol. 20 pp. 29-66.

[9] Hillier B and Shu S: 2001, Crime and urban layout: The need for evidence, in Ballintyne S, Pease, K and McLaren V: *Secure Foundations: Key Issues in Crime Prevention and Community Safety*. IPPR, London.

[10] Jupp J, and Gero, JS: 2003, Towards computational analysis of style in architectural design. in S Argamon (ed), *IJCAI03 Workshop on Computational Approaches to Style Analysis and Synthesis*, IJCAI, Acapulco, pp 1-10

[11] Maher ML & Poon J: 1996, Modelling design exploration as co-evolution, Microcomputers in Civil Engineering (Special Issues on Evolutionary Systems in Design)

[12] Nehaniv CL and Dautenhahn K: 1999, Of hummingbirds and helicopters: An algebraic framework for interdisciplinary studies of imitation and its applications. *In Learning Robots: An Interdisciplinary Approach*, J. Demiris and A. Birk, eds., World Scientific Press, in press.

[13] Peponis J, Hadjinikolaov E, Livieratos C and Fatouros DA: 1989, The spatial core of urban culture, *Ekistics* 56(334/335), pp. 43-55.

[14] Robles-Kelly A, and Hancock E R: 2003, Edit Distance From Graph Spectra. *Proceedings of the ninth IEEE International Conference on Computer Vision* (ICCV 2003)

[15] Saunders R and Gero JS: 2001, Artificial creativity: A synthetic approach to the study of creative behaviour, in JS Gero and ML Maher (eds), *Computational and Cognitive Models of Creative Design V*. Sydney, pp. 113-139.

[16] Spiliopoulou G and Penn A: 1999, Organisations as Multi-Layered Networks, *Proceedings, 2nd Intl. Space Syntax Symposium*, pp. 1-24.

[17] Steels L: 2000, The Emergence of Grammar in Communicating Autonomous Robotic Agents, in Horn W (ed), *ECAI2000*. IOS Press, Amsterdam. pp 764-769.

[18] Turner, A: 2005, An Algorithmic Definition of the Axial Map, *Environment and Planning B: Planning and Design*, 32(3) 425-444.

[19] Turner A, Doxa M, O'Sullivan D, and Penn A: 2001, From isovists to visibility graphs: a methodology for the analysis of architectural space. *Environment and Planning B: Planning and Design*, 28(1) 103-121.

[20] de Weck O: 2004 Multiobjective Optimization: History and Promise", *The Third China-Japan-Korea Joint Symposium on Optimization of Structural and Mechanical Systems*, Kanazawa, Japan.

[21] Zhu P and Wilson RC: 2005, A Study of Graph Spectra for Comparing Graphs. *British Machine Vision Conference 2005.*