Overview of a Generic Evolutionary Design System

P. J. Bentley & J. P. Wakefield

Abstract

This paper presents an overview of a generic evolutionary design system. The system uses a hybrid 'steady-state' multiobjective genetic algorithm with an explicit mapping stage between genotypes and phenotypes to evolve designs. The system evolves the geometries of a range of solid object designs from scratch. A selection of evolved designs are presented.

Key words: evolutionary design, genetic algorithm

1. Introduction

The optimisation of designs by computers is an area in which research is rapidly growing. Computers remove the need for the laborious iterative process of manually fine-tuning design parameters - designs can be optimised automatically [7]. However, current design optimisation methods are usually application-specific, i.e. a system will only be able to optimise a single design. This has the major disadvantage that a completely new system is required for almost every new design optimisation task.

Additionally, such systems are limited to optimising existing designs - very few (if any) are truly capable of creating new designs from scratch. In other words, these optimisation systems require a human designer to input a preliminary design to the computer, and specify exactly which parameters of this design are to be optimised. A system capable of creating designs from scratch, (i.e. creating entire designs from nothing) would not require any preliminary designs, or laborious manual specification of which parameters are to be optimised.

The main advantages of a system that could create new designs are apparent: a computer is not limited by 'conventional wisdom', so it could generate original designs based on entirely new principles. Consequently, a design system capable of creating new designs from scratch, combined with the generic ability to create more than one type of design, would be highly desirable. Such a system would help human designers produce better designs, faster, by presenting new alternative designs, already optimised for the design task. Moreover, because of its generic nature, it would be able to do this, not just for one design task, but for a range of different design tasks.

In summary, it is the goal of this research project to develop such a generic evolutionary design system, and to explore its capabilities by applying it to a number of example design tasks.

2. Genetic Algorithms

So what type of computer program could achieve this? For the more traditional problem of design optimisation, *adaptive search* has become one of the most popular methods. In particular, the adaptive search algorithm known as the genetic algorithm (GA) has become widely used in design optimisation tasks [6,7]. The GA has been shown repeatedly to be a highly flexible stochastic algorithm, capable of finding good solutions to a wide variety of problems [6]. It therefore seems that, for the complex problem of design creation, a good choice would be to use one of the most flexible and powerful of search algorithms known in computer science: the genetic algorithm.

The GA is based upon the process of evolution in nature [5]. Evolution acts through large populations of creatures which individually reproduce to generate new offspring that inherit some features of their parents (because of random *crossover* in the inherited chromosomes) and have some entirely new features (because of random *mutation*). Natural selection (the weakest creatures die, or at least do not reproduce as successfully as the stronger creatures) ensures that more successful creatures are generated each generation than less successful

ones. It can be argued that in nature, evolution has produced some astonishingly varied, yet highly successful forms of life. These creatures can be thought of as good 'solutions' to the problem of life. In other words, evolution optimises creatures for the problem of life.

In the same way, within a genetic algorithm a population of solutions to the problem is maintained, with the 'fittest' solutions (those that solve the problem best) being randomly picked for 'reproduction' every generation. 'Offspring' are then generated from these fit parents using random crossover and mutation operators, resulting in a new population of fitter solutions [5]. As in nature, the GA manipulates a coded form of the parameters to be optimised, known as the *genotype*. When decoded, a genotype corresponds to a solution to the problem, known as a *phenotype*.

Genetic algorithms are typically initialised with a completely random population (i.e. every member of the population having random genotypes, and thus random phenotypes). Even if this was not the case (e.g. if an existing solution was to be optimized), because of the random search operators, the end result often cannot be predicted [5]. For a simple problem with only one optimal solution, the population of solutions in the GA will typically converge to a solution close to this single solution every time. However, for more complex problems with more than one optima, exactly which optimum the GA will converge to cannot be predicted. Hence, the GA optimises solutions provided at the start (whether random or not), and will always converge to a 'fit' solution, but not necessarily a globally optimal solution.

3. Evaluation Software

To allow the GA to pick 'fit' solutions from the current population for reproduction every generation, the decoded solutions (the phenotypes) must be evaluated. Although such evaluation can be performed by a human designer, this would cause some problems. Firstly, for a typical 'run' of the GA, thousands of separate evaluations are required - a rather tedious task for a human (especially if prototype models must be built for each solution). Secondly, such evaluations must be precise enough to allow the optimisation of the designs, meaning that to evaluate almost any type of design, a considerable amount of calculation is required. Thirdly, by using a human as the evaluator, the system will inevitably only be guided by 'conventional wisdom', thus removing any potential for originality from the system. Hence, the quickest, most accurate, and potentially most beneficial solution is to use *evaluation software* to guide evolution. Furthermore, such software is normally used already for design optimisation tasks (for speed and accuracy).

Most design problems can be broken down into a number of separate criteria (e.g. the most basic of these being correct size and mass). By creating 'modular' evaluation software, with each module capable of evaluating any suitably represented design for a particular criteria, complete design problems can be specified by a combination of such modules. In this way, new design applications can be specified using mostly existing modules [2] and thus require a minimum of new evaluation software (or interfacing to existing software). Importantly, all such software must only specify the *function* of the desired design. Should any part of the software specify the shape directly, the system is again constrained against original design. (Additionally, if the shape of the design is already known, there is little reason to use a design system to create a design of that shape.)

4. The System

The generic nature of the system is currently limited to the creation and optimisation of the *geometry* of three dimensional solid objects. Although it is possible to optimise the surface appearance of the objects (e.g. colour, texture) and the material the objects are composed of, this project is concentrating on the geometry of objects.

Because of the complexity involved in specifying even simple designs, the task of creating a GA capable of generating new designs is not trivial. Design problems are typically highly constrained, multicriteria problems, making consistent evolution to good solutions very difficult, even for an algorithm as powerful as the GA.

The GA that forms the core of the prototype system is initialised with a population of random designs (i.e. starting from scratch). These designs are represented in memory using a spatial-partitioning representation

designed for this purpose [1]. The algorithm then begins an iterative process of evaluation and reproduction to generate new populations of increasingly better designs. Alternatively, the system can generate new designs using given components, by seeding the initial population with randomly positioned design components, and then continuing as before. By fixing all parameters specifying depth, two-dimensional designs can be created in addition to three dimensional designs.

Although initial experiments were performed with a version of Goldberg's 'simple GA' [5], this was soon found to be inadequate [3,4]. The genetic algorithm currently used is a hybrid of a number of different types of GA. Perhaps the three most notable aspects of the GA are as follows: firstly, an explicit mapping stage between genotypes (coded designs) and phenotypes (designs) is maintained within the algorithm [2]. Although some researchers blur the distinction and actually evaluate genotypes directly, by having a mapping stage, a simple coded design can be mapped to a complex actual design. The system uses this, when required, to generate symmetrical designs by reflecting designs in one or more planes during the mapping process. In this way, a complex symmetrical design need only have the non-reflected portion manipulated by the GA, thus reducing the difficulty of the design task for the GA. Additionally, this mapping stage is used by the system to enforce the rules of the solid object representation, by mapping illegal designs to legal designs.

A second point of note concerning the GA is the use of multiobjective optimization techniques, to allow multicriteria design specifications to be handled effectively. Various alternative multiobjective ranking methods were explored and compared in detail, with a new method created for this work producing the most consistently good results [4]. The resulting multiobjective genetic algorithm can deal with any number of separate objectives, and treats all objectives equally, or according to user-specified relative importance values.

Finally, the third significant aspect of the GA is the way new populations of solutions are generated. A standard GA replaces the whole population of solutions with an entirely new population, every generation. This can mean that a single, very good solution is lost before it can contribute sufficient offspring to future generations. Perhaps more distressing however, is the fact that during the final stages of evolution, solutions can actually get worse, instead of better. An alternative algorithm known as the 'steady-state' GA does exist to tackle these problems. This algorithm only replaces solutions in a population with better solutions (i.e. 'killing' the least fit). However, it does not pick the fittest members of a population for reproduction, so the selection pressure is considerably reduced, resulting in slower evolution. The hybrid GA used within the design system uses a similar replacement method to the steady-state GA, in that less fit solutions are usually replaced by more fit solutions, but additionally, the fittest are picked for reproduction. This means that designs evolved by this GA can only improve, and that the speed of the evolution process is not reduced.

5. Results

A variety of different design tasks have been presented to the system. Early work involved the design of simple tables [2,3] (evaluated for size, mass, stability, supportiveness, and a flat upper surface). Very fit designs, with and without symmetry, were consistently evolved, fig. 1. Additionally, the system has been applied to the task of evolving a range of different optical prisms (evaluated for size and desired ray-traced optical characteristics), fig. 2 (left). Most recent work involves the evolution of 'pseudo aerodynamic' designs (evaluated by a naïve particle flow simulator, with the desired forces exerted on the designs being specified), fig. 2 (right).

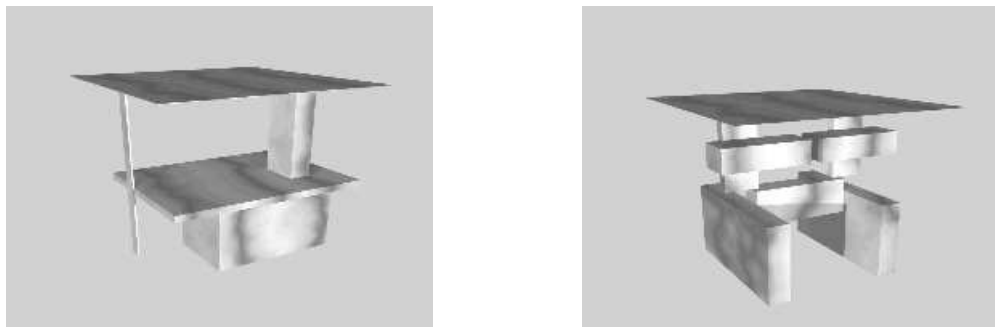

Fig. 1. Two tables evolved by the system.

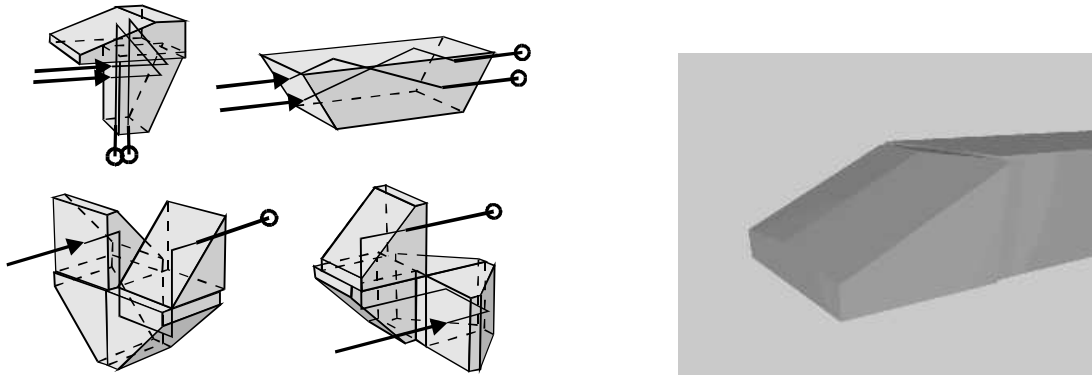

Fig. 2. Some optical prisms evolved by the system (left).
The first simple 'aerodynamic' design evolved by the system (right).

6. Conclusions

Genetic algorithms are capable of more than just design optimisation - they can be used to create entirely new designs. This paper has given an overview of a generic evolutionary design system, capable of evolving a range of different solid object designs from scratch. The system uses a hybrid 'steady-state' multiobjective genetic algorithm with an explicit mapping stage between genotypes and phenotypes. A selection of results evolved by the system were presented.

7. References

1. Bentley, P. J. & Wakefield, J. P. (1994). Generic Representation of Solid Geometry for Genetic Search. *Microcomputers in Civil Engineering* 11:3 (to appear).
2. Bentley, P. J. & Wakefield, J. P. (1995). The Evolution of Solid Object Designs using Genetic Algorithms. In *Applied Decision Technologies* (ADT '95), April 1995, London, (pp. 391-400).
3. Bentley, P. J. & Wakefield, J. P. (1995). The Table: An Illustration of Evolutionary Design using Genetic Algorithms. In *Genetic Algorithms in Engineering Systems: Innovations and Applications* (GALESIA '95), Sept. 1995, Sheffield, (pp. 412-418).
4. Bentley, P. J. & Wakefield, J. P. (1995). Multiobjective Ranking with Genetic Algorithms: Range-Independence and Importance. Submitted to *Evolutionary Computation*.
5. Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization & Machine Learning*. Addison-Wesley.
6. Holland, J. H., (1992). Genetic Algorithms. *Scientific American*, 66-72.
7. Parmee, I C & Denham, M J, (1994). The Integration of Adaptive Search Techniques with Current Engineering Design Practice. In *Adaptive Computing in Engineering Design and Control -'94*, Plymouth, (pp.1-13).