Engineering Design and Automation 2:3 © 1996 John Wiley & Sons, Inc.

Conceptual Evolutionary Design by a Genetic Algorithm

Peter J. Bentley* & Jonathan P. Wakefield

Abstract: *This paper describes a prototype design system which uses a genetic algorithm to evolve new conceptual designs from scratch (i.e. without the input of preliminary designs). The system is applied to a set of design tasks which are 'hard' for a genetic algorithm: the design of optical prisms. These demonstration problems consist of the generation of appropriate geometries for optical prisms to allow light to be directed through them according to various design specifications. Despite the deceptive nature of the problem for evolutionary search, the system is shown to create numerous types of prism successfully, either performing the whole design process entirely itself, or assembling new designs out of smaller, previously evolved components.*

Keywords: genetic algorithms, evolutionary search, conceptual design, optical prisms

1. INTRODUCTION

The engineering design process performed by a human designer can be generally summarised as consisting of the following stages [8,11,18]:

1. preliminary or conceptual design
2. detailed design
3. evaluation
4. iterative redesign, if the evaluation results are unsatisfactory

To date, computers have been used successfully for all of these stages except the first: conceptual or *creative* design [8,11]. As stated by Goldberg [11]: "The creative processes of engineering design have long been regarded as a black art. While the engine of analysis steamrolls ever forward, our understanding of conceptual design seems locked in a timewarp of platitudes, vague design procedures, and problem-specific design rules."

A computer system capable of supporting this creative design process would help human designers produce better designs, faster, by presenting a range of new alternative designs. Moreover, a computer is not limited by 'conventional wisdom', so it could generate original designs based on entirely new principles.

This paper describes a prototype system capable of creating new designs and iteratively optimizing these designs, using a genetic algorithm (GA). The system is generic, i.e. capable of the creation and optimization of the geometry of a wide range of three dimensional solid objects. Instead of the more traditional approach of concentrating solely on the conceptual design stage, this system performs the whole design process without explicitly dividing the process into artificial stages. Indeed, as noted by Pham [18], there is no easy way to determine when creative design stops and the optimization of the design begins.

Previous work has investigated the application of this design system to simple, example design tasks [3], e.g. the design of a table [4]. In this paper, to explore (and hopefully overcome) current limitations of the system, it is applied to a set of 'hard' problems for a GA: the creation of suitable geometries for various optical prisms to allow light to be directed correctly through them.

2. BACKGROUND

Although research into the subject of design optimization is common [17], the area of automated creative design is still relatively unexplored. Relevant work in this area includes not only previous attempts to

generate designs, but also the creation of artistic images.

2.1 Creative Design by Computers

Early work concentrated on the cognitive area of creative design automation [8,9], i.e. attempting to make a computer 'think' in the same way as a human, when designing. Such systems attempted to create descriptions of designs at an abstract level, typically using an expert system to 'design'. For example, Dyer's 'EDISON' [9] represented simple mechanical devices such as doors and can-openers symbolically in terms of five components: parts, spatial relationships, connectivity, functionality and processes. Another approach consisted of invention based on 'visualising potential interactions' [22]. This generated descriptions of designs in terms of high-level components and the interactions between them, using qualitative reasoning and quantitative algebra. Unfortunately, such methods can typically only deal with highly simplified designs [18].

More recent work in this area has used adaptive search techniques such as the genetic algorithm (GA), to *evolve* new designs. For example, Michielssen [16] describes an approach for designing optimal multilayer optical filters using a GA. However, closer inspection reveals that the system optimizes existing preliminary designs rather than genuinely creating new designs. Pham [18] reduces the complexity of automated creative design by presenting the computer with a number of high-level design building blocks for transmission systems, such as rack and pinion, worm gear, and belt drive. A GA is then used to order these elements and thus create a new design.

Perhaps the work that can most accurately be described as creative design is the recent work of Rosenman [19], who attempts to evolve new floorplans for houses. Two dimensional plans are 'grown' using a simplified GA to modify 'cells' organised hierarchically using grammar rules. However, the system requires much problem-specific knowledge and the elaborate representation used may actually prevent complex shapes from being formed.

2.2 Artistic Image Creation by Computers

The use of computers to create art (usually with GAs and similar adaptive search algorithms) is growing in popularity amongst some artists. For example, Todd and Latham have successfully evolved many three dimensional 'artistic' images and animations [20]. Their two-part system, consisting of 'Form Grow' and 'Mutator' uses an evolutionary strategy which creates and modifies shapes composed of 'artistic' primitive shapes (e.g. spiral, sphere, torus). John Mount shows his 'Interactive Genetic Art' on the internet (at http://robocop.modmath.cs.cmu.edu. 8001). This work uses a GA to modify fractal equations that define two dimensional images. Additionally, the biologist Richard Dawkins has demonstrated the ability of computers to evolve shapes resembling those found in nature [7]. Using a simple evolutionary strategy that modifies shapes arranged in tree-structures, he has produced images resembling the shapes of life-forms, e.g. 'spiders', 'beetles', and 'flowers'.

However, all of these art creation systems require the images being evolved to be evaluated by a human (i.e. artificial selection). Moreover, despite the fact that some of these systems can produce some complex three dimensional shapes [20], none of them have been used to produce anything more than 'pretty pictures'.

The system described in this paper combines the creative evolutionary techniques pioneered by artists (and biologists) with the more rigorous methods of automatic creative design. This has resulted in a generic design system which has the 'creative properties' of the art systems and is capable of the generation of a wide range of useful designs. Furthermore, it is the 'innovative flair' [10] of the genetic algorithm that gives the system such capabilities.

3. THE GENETIC ALGORITHM

In addition to its use in creative design and art systems, the adaptive search algorithm known as the genetic algorithm (GA) has become widely used for the more traditional problem of design optimization [1,14,21]. In this and many other domains, the GA has been shown repeatedly to be a highly flexible stochastic algorithm, capable of finding good solutions to a wide variety of problems [10,13].

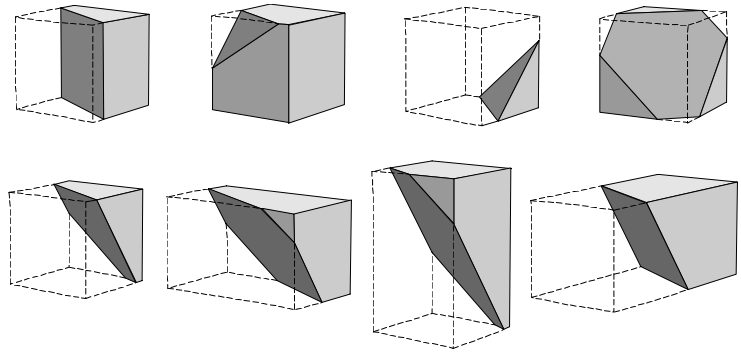

Figure 1 Clipped stretched cubes.

The GA is based upon the process of evolution in nature [13]. 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 [13]. As in nature, the GA manipulates a coded form of the parameters to be optimized, 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 [10]. GAs typically converge to good or 'fit' solutions to problems, but not necessarily to a globally optimal solution.

4. PROTOTYPE EVOLUTIONARY DESIGN SYSTEM

The design system outlined in this paper consists of three elements:

1. A suitable representation of solid objects to allow the computer to manipulate candidate designs effectively during the design process.
2. A modified genetic algorithm to *evolve* such represented designs from scratch.
3. Evaluation software to guide the evolution process.

4.1 Representation of Designs

In order to allow a computer to create and optimize the complete shape of a design, the design must be fully described by some form of representation. Since the system is intended to be capable of optimizing the shape of any three dimensional solid object, this representation must be able to fully describe any such object. Not only that, however, the representation must also be suitable for manipulation by genetic algorithms, to allow any such represented design to be modified in any way.

After some investigation, it was discovered that the requirements for a generic solid object representation to be manipulated by a genetic algorithm differ considerably from requirements where the representations are to be manipulated by a human [2]. A computer-aided-design (CAD) package requires that designs should be very easy to modify by users - it does not matter how complex the underlying representation is. For a generic evolutionary design system, it is the genetic algorithm that must modify the designs. Thus, the representation must use as few definition parameters as possible (to minimise the corresponding search space), whilst allowing designs to be easily modified in any way (to allow steady evolution). In other words, the design-space to be searched by the GA must be carefully specified to ensure that similar designs are always 'close' to each other in the space. This is not the case in many standard representations used in CAD (e.g. constructive solid geometry, where changing a single parameter value can radically alter a design).

Many existing representations were examined closely (e.g. polygon mesh, Hermite, Fourier, Bézier, constructive solid geometry, sweeping), with the conclusion that the most suitable form of representation is 'spatial

partitioning' [2]. However, since most partitioning methods require large numbers of definition parameters and are only capable of crude approximations of some surfaces, a new variation was created for this work. This combines ideas from the commonly used constructive solid geometry (CSG) and standard spatial partitioning methods, by allowing every partition, or *primitive*, to vary in size and position, and to be intersected by a plane of variable orientation, see Fig. 1. These highly flexible primitives, when used in combination, are capable of closely approximating any 3D (and 2D) solid object whilst requiring very few definition parameters. As with standard spatial partitioning representations, a design is represented by a number of these non-overlapping primitives. More specifically, a design is considered illegal if it has primitives which overlap to any extent. Software to enforce these rules is contained within the design system [3].

4.2 Evolution of Designs

To allow the manipulation of designs represented in this way, a genetic algorithm forms the core of the prototype evolutionary design system. This GA is initialised with a population of random designs (i.e. starting from scratch). 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. (This ability to evolve in two dimensions is highly beneficial for a subset of suitable two-dimensional design problems, however most design applications typically require evolution in three dimensions.)

Although initial experiments were performed with a version of Goldberg's 'simple GA' [10], this was soon found to be inadequate [4,5]. 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. 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 (i.e. a design with overlapping primitives is corrected by 'squashing' such primitives until they touch instead of overlap).

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 [5]. The resulting multiobjective genetic algorithm can deal with any number of separate objectives, automatically scaling separate fitness values during evolution in order to allow all objectives to be treated equally, or according to user-specified relative importance values.

The third notable aspect of the GA is the way that 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 [10] 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 (i.e. a steady-state GA

with preferential selection). This means that designs evolved by this GA can only improve, and that the speed of the evolution process is not reduced.

The GA modifies the genotypes of designs by using standard, single-point uniform crossover (i.e. crossover at a single, random point) to generate two offspring from each pair of parent solutions. Mutation occurs with a probability of 0.05, simply 'flipping' a random bit in the genotype. Every primitive shape requires nine definition parameters to specify its 3D position, width, height, depth, and orientation of its clipping plane. Hence, the genotype consists of a pre-defined number of multiples of nine parameter values, coded as 16-bit binary numbers. For example, a two-primitive design requires eighteen parameters and a genotype of 288 bits. Alternatively, a two-primitive symmetrical design (i.e. a design that has been reflected in a plane to make it symmetrical) requires only a partial design of one primitive (nine parameters) coded in the genotype as 144 bits. Because no hard constraints are used (i.e. every possible genotype is mapped to a phenotype of some description), the search-space is continuous. Inevitably, however, the more primitives the user decides to allow in designs, the larger and more difficult to search this search-space will be.

4.3 Evaluation of Designs

To allow the GA to pick 'fit' solutions from the current population for reproduction every generation, the decoded solutions (the phenotypes) must be evaluated. To achieve this, *evaluation software* is used. By using software to evaluate designs, a human designer is saved the task of laboriously judging thousands of evolving candidate designs. Moreover, the potential for originality by the system is increased by removing the possible limitation of the 'conventional wisdom' of human designers.

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 criterion, 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 and 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.

Previous work has involved the creation of various modules of evaluation software, to allow the evaluation of designs for features such as size, mass, flatness, and stability under gravity. These have been used in combination to demonstrate the generic capabilities of the system to evolve solutions to various example design tasks [3,4,5]. In this paper, the system is required to design various types of optical prism. To specify the function of most types of prism, three main modules of evaluation software are required. One module specifies the upper and lower boundaries of the size of designs, another defines the input to and required output light characteristics from the designs, and a third specifies the fact that designs should be unfragmented.

MODULE 1: LIMITS UPON SIZE

Perhaps the most basic requirement for any design is that of appropriate size: a penta prism for a camera must not be larger than the camera itself. The size of the design is specified by minimum and maximum extents for the left, right, back, front, top and bottom of the design, see fig. 2. The fitness of a candidate design decreases the further its extents fall outside the specified limits.

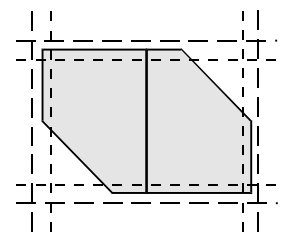

Figure 2 Desired minimum and maximum extents (shown by dotted lines) of evolved design.

Although this module defines a constraint rather than an objective, it is implemented as a soft constraint, i.e. designs of unsatisfactory size are penalised with poor fitness values, rather than being prevented from existing. The definition of acceptable limits of size allows other related criteria, such as the size of the reflected image and the distance the light should travel from source to destination, to be specified.

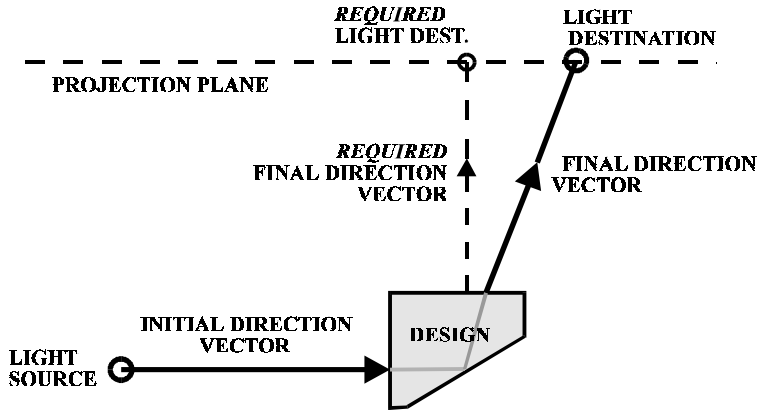

Figure 3 Specifying input and output light characteristics.

MODULE 2: RAYTRACING

Optical prisms are designed to bend and refract light from source to destination along a specific path and in a specific way in order to perform their various functions. In order to give the design system complete freedom during the design process, this software module does not directly specify the path the light should take. Instead, a light source and initial direction vector is given. This 'light ray' is then traced through the current design being evaluated (using standard ray-tracing techniques) and the emerging ray intersected by specified 'projection plane'. The output direction and destination point of the ray is then compared to the required direction and destination point to allow calculation of the fitness of the design, see fig. 3.

The further the actual direction vector differs from the required direction vector, the less fit the design is. Likewise, the further the actual destination point of the ray is from the desired destination point, the worse the fitness of the design. Any design with output direction vectors so incorrect that they do not intersect the projection plane at all (resulting in no destination points) is penalised heavily. Normally five 'rays' are used to specify the four corners of an image and a centre point. This allows the size and orientation as well as the position of the required output image to be specified. For some types of prism, additional criteria are required, e.g. no refraction permitted. Designs that do not conform to these additional criteria are penalised with very poor fitness values.

MODULE 3: UNFRAGMENTED

A design is considered fragmented if one or more of the primitives that represent it have become detached from the main part of the design, fig. 4. This is possible because the positions of primitives are defined independently of each other. Since the system is required to create a single prism at a time, all designs must be unfragmented.

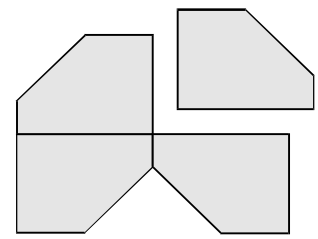

Figure 4 A design is fragmented if a primitive has become detached from the main part of the design.

Fragmented designs are detected by creating a network of the primitives and their connections in a design. A primitive is connected to another if it touches another. Two primitives are detected as touching by calculating the lines formed by the intersection of all planes forming the sides of the primitives. These lines are then clipped to both primitives - if one is not clipped out of existence, the two primitives must touch. By using efficient clipping routines in this way, this process is very fast. Once the network is formed, it is traversed recursively. Any primitive that is not part of the main design will not be visited, meaning that the design is fragmented.

This criterion is implemented as a soft constraint, with fragmented designs being penalised very heavily. A large relative importance is also specified for this criterion, meaning that this is usually the very first criterion to be evolved correctly in a design.

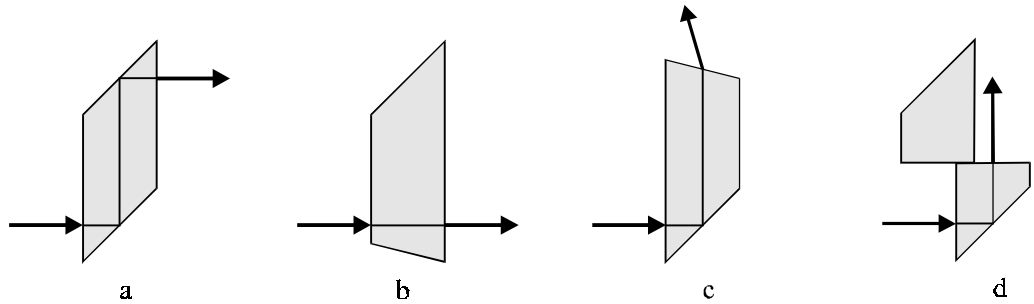

Figure 5 Correct rhomboid prism (a), failed rhomboid prism attempts (b,c,d).

5. EVOLUTION OF OPTICAL PRISMS

To investigate the current capabilities of the prototype system, a 'hard' example design task was set: to evolve a variety of different types of optical prism. This problem was chosen for two reasons: first to demonstrate that the system can successfully create designs using the full representation described earlier. (Previous versions of the system have only used a simplified version of the representation, using primitives without intersecting planes [3,4,5]. Using planes to 'slice' portions off primitive shapes enhances the ability of the representation to define solids, but increases the number of parameters that need to be considered by the system.) Second, by using this problem, the abilities of the system to evolve good solutions to problems deceptive to GAs can be explored and improved.

This section explores the problem and discusses why it is deceptive to GAs. Typical evolved designs of seven different types of prism are then described.

5.1 Optical Prisms

An optical glass prism can have three effects on light directed through it. Depending on the refractive index of the prism, the wavelength of the light and the angle at which the light hits a surface of the prism, the path of the light will be unaffected, refracted (i.e. bent) or reflected. For the design tasks described in this paper, only monochromatic light (light of a single wavelength) is considered, with surface reflections being ignored. These restrictions are purely to simplify and speed up the evaluation process; more realism could be introduced by repeatedly evaluating designs using light of varying wavelengths. Surface reflections are usually insignificant with optical prisms (as opposed to total internal reflections) and can be safely ignored without significant loss of realism for most applications.

Prisms are used in many optical devices, for numerous reasons. Binoculars require a prism erecting system to both 'squash' their overall length to a more manageable size, and to keep the images erect (in the same orientation as the object being viewed) [6]. Periscopes require derotating prisms to keep the image erect for the observer, no matter to what degree they are rotated [15]. An SLR camera requires a constant-deviation prism (usually a penta prism) to ensure that the deviation of the optical axis is unchanged by rotation of the prism [6]. Whilst mirrors can also be used for these tasks, prisms have a number of advantages. Firstly the relation between their reflecting faces is not subject to change because of mechanical misalignment or movement. Secondly, dust does not affect reflectivity in the same way as with mirrors, and finally, when total internal reflection occurs within a prism, reflectivity is higher than can be obtained with a mirror [6].

5.2 Deceptive Problems

For a genetic algorithm, it is 'more difficult' to evolve good solutions to certain types of problem [12]. Such problems are termed *deceptive* for GAs, with many readily accessible local minima or *deceptive attractors*, and a single, hard to reach global minima. It transpires that certain types of prism design problem are deceptive for GAs. For example, consider the rhomboid prism which acts like a naïve periscope, fig. 5a. Initially, light is reflected upwards, then it is reflected again to travel across to its destination. If either the top or bottom reflection is absent (fig. 5b, 5c), or if the two primitives making up the design are not correctly aligned (fig. 5d), the design fails. In other words, every part of the design relies completely on all of the other parts of the design to be correct for the prism to work correctly as a whole.

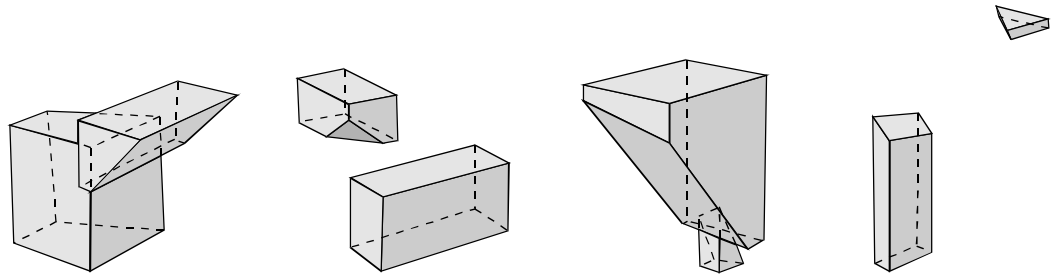

Figure 7 Four examples of initial two-primitive random designs.

Although designs such as those shown in fig. 5b and 5c could be combined to form a perfect rhomboid prism, the GA would rarely pick these designs for reproduction. This is because both of these designs are very unfit compared to the deceptive attractors [12] to the problem, see fig. 6. Being simpler designs, deceptive attractors are statistically far more likely to be created by chance than any close approximation to the rhomboid prism. The fitness of these designs is reasonable, and so any GA, picking the best for reproduction (and/or 'killing' the worst), almost always ensures that it is this type of design that predominates in future populations. Thus, designs such as those shown in fig. 5b and c are quickly eliminated from the population in favour of the deceptive attractors, before the good features of the less fit designs can be exploited. Hence, the usual methods used in evolution of combining features from different good designs to create better designs is prevented from working.

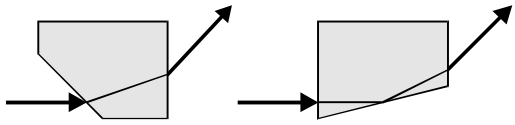

Figure 6 Two deceptive attractors to the 'rhomboid prism' problem.

The problem of designing prisms increases in difficulty as the required path of the light becomes more complex. The more the light needs to be reflected inside a prism, the smaller the number of acceptable designs, and the larger the number of deceptive attractors. A possible remedy to this problem is to increase population sizes, to increase the probability of at least a nearly correct conceptual design appearing and hence being picked. However, unlike nature, there are real limits to population sizes because of limits on computer memory and processing time. Better results can be obtained by greatly modifying the GA [12], but since this system is intended to be generic, specialising the GA for a single class of problem is not a desirable solution.

Other possible solutions involve providing additional problem-specific information to decrease the number of deceptive attractors and thus lower the probability of them being picked instead of genuinely good conceptual designs (i.e. 'smooth' out some of the local minima). For example, for the rhomboid prism problem, by requiring that no refraction takes place and specifying specific areas in front of the design that light must not enter (using soft constraints), the number of deceptive attractors such as those shown in fig. 6 is substantially reduced. This reduction occurs because the fitness of such designs is decreased, changing them from deceptive attractors to poor designs (i.e. the 'fitness landscape' of the search space is reshaped, without introducing any search restrictions in the space).

5.3 Evolution from Scratch

The current capabilities of the design system were initially explored by using the system to evolve five different types of prism from scratch (i.e., beginning with purely random initial designs). Beginning with the right-angle prism, the complexity of each prism required was increased. For any prism which the system found 'hard' to design, methods to encourage production of more acceptable designs were explored. All designs except for the right-angle prisms were defined by two primitives of the solid object representation (18 parameters). Fig. 7 shows typical examples of initial, random designs prior to evolution.

Population sizes used within the GA were varied from 100 to 400, depending on the difficulty of the problem. All results shown were generated after 500 generations (although the GA had converged on solutions to some of the simpler problems after only 50 generations). At least twenty test runs for each design task took place, with typical results for

each being shown. Unless otherwise stated, all designs were evaluated using five 'light rays' as shown in fig. 8. Input rays are shown by arrows, output rays are displayed terminating with circles to show the positions of the light destination points. Light paths as calculated by the ray-tracing module of evaluation software are shown within the designs. For clarity, not all rays will be shown for every result.

RIGHT-ANGLE PRISM

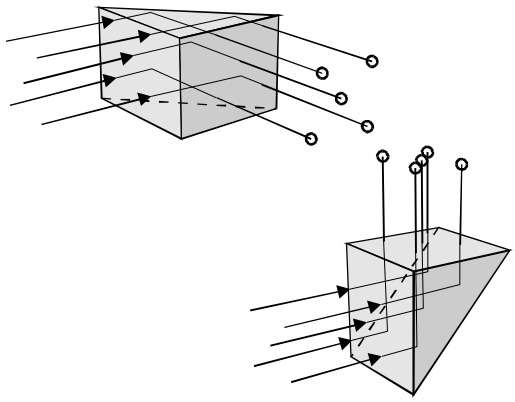

Figure 8 Typical evolved right angle prisms.

The function of a right-angle prism is to reflect light by exactly 90 degrees. Using a single primitive of the representation, a variety of differently oriented prisms were successfully evolved, with highly accurate designs being produced every time, fig. 8.

ROOF PRISM

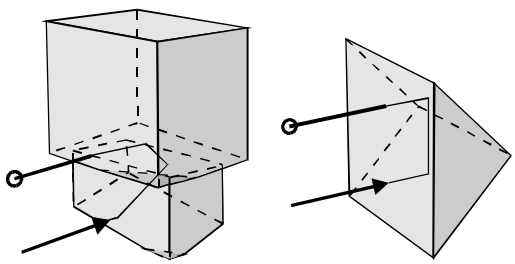

Figure 9 Typical roof prism attempt (left), perfect roof prism with symmetry (right).

The purpose of a roof prism is to bend the path of light back in the opposite direction to the source direction, with the destination being at a pre-defined distance above the source. The output light should not be erect (i.e., the output image should be upside-down). Initial results for this problem used ingenious combinations of refraction and reflection to achieve good, but not perfect, results, fig. 9 (left). Typical imperfections consisted of inaccurately directed output rays and the use of refraction, which would 'blur' the colours of the output image. By specifying that a symmetrical design was required, the system evolved perfect roof prisms every time, fig. 9 (right).

DEROTATING PRISMS

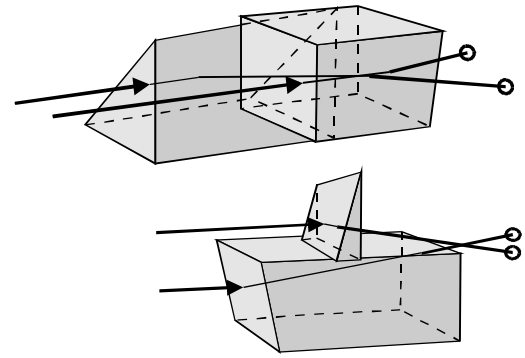

Figure 10 Typical 'cheating' derotating prism attempts.

Derotating prisms rotate the orientation of an image when the prism is rotated about the optical axis (commonly used in periscopes). For this problem, however, this feature was not directly evaluated for the GA. Instead, to keep the complexity of evaluating the designs simple, a design that turned an image upside-down was specified (a common feature of many derotating prisms). To do this, any design without the positions of its output rays mirroring the positions of the input rays was penalised. Unfortunately, by only partially specifying the required design, the system was able to 'cheat' and split images, sending each half of the image to the correct side, but leaving both halves the wrong way up, fig. 10. Although the designs usually did act in the way required by the evaluation software, many were not true derotating prisms. In other words, the design system was exploiting the 'loop-hole' in the design specification, to produce the simplest types of designs it could.

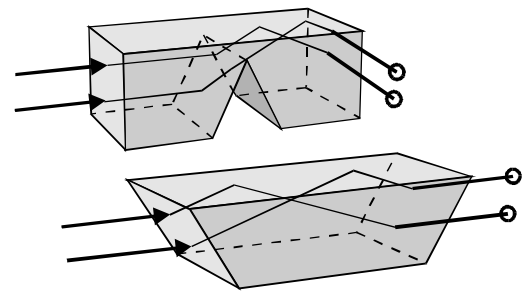

Figure 11 Variation of a 'K' derotating prism with symmetry (top),
nearly perfect derotating 'dove' prism with symmetry (bottom).

By specifying that symmetrical designs were required, most of these 'cheats' became unavailable, and the system successfully evolved some nearly perfect derotating prisms. Figure 11 (left) shows an unusual variation of a 'K' prism which uses refraction and only a single reflection (a normal 'K' prism reflects three times, with no refraction). Although the output rays are bent by refraction, this is a 'fit' derotating prism. Figure 11 (right) shows a traditional 'dove' derotating prism, evolved by the system.

RHOMBOID PRISM

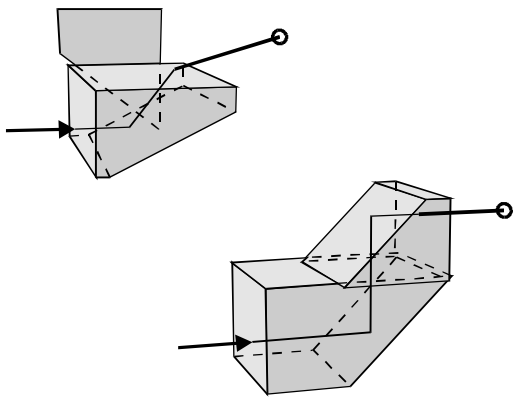

Figure 12 Typical rhomboid prism attempt (top), nearly perfect 'rhomboid' prism (bottom).

The purpose of a rhomboid prism is to reflect the light entering it, so that the light emerges in the same direction and orientation, but at a specified distance above the source (used in less expensive binoculars). As mentioned previously, this problem has many deceptive attractors to it, meaning that the typical design produced is as shown in fig. 12 (left). Since specifying symmetry does not help in this case, different problem-specific knowledge was introduced to the evaluation software. It was found, after some experimentation, that the system could evolve good solutions to this 'hard' problem by penalising any designs that refracted the light, and specifying that the designs should be two-dimensional, fig. 12 (right). The latter did not need to be rigidly enforced: parameters specifying depth and position on the Z-axis were simply initialised with a set value instead of a random value, but were not fixed thereafter (i.e., the system could, and did, change these values).

PENTA PRISM

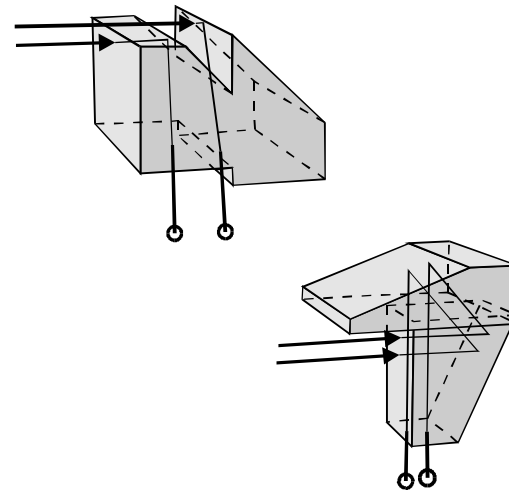

Figure 13 Typical 'cheating' attempt at penta prism (top), nearly perfect penta prism (bottom).

The function of a penta prism is to reflect light by 90 degrees, whilst keeping the image erect. (Unlike a right-angle prism which reflects light by 90 degrees, but the output image is the wrong way up.) These prisms are almost always used in SLR cameras to direct the light from the mirror up to the viewfinder. This is a more difficult problem than the rhomboid prism problem, because, for a correct output, the light must be reflected twice, initially in a direction almost opposite to the final, desired direction. Again, the system initially 'cheated', by producing designs that were effectively composed of two right-angle prisms joined, fig. 13 (left). The image is split by these, placing the top half at the top and the bottom half at the bottom, but leaving both halves upside-down. Unlike the similar problem encountered with the derotating prisms, specifying symmetry for these designs does not help. After some experimentation, it was found that by increasing the number of light rays traced through the designs, and increasing the importance of placing the rays in the correct positions, the system was able to evolve good solutions to this problem, fig. 13 (right). (Additionally, the system was provided with the knowledge that the design was two-dimensional, as described earlier.)

5.4 Evolution from Previously Evolved Components

As demonstrated above, by giving a more detailed problem specification within the evaluation software it is possible to make the system evolve good solutions to deceptive design problems. However, when the difficulty of the problems are increased further, the designs of the system deteriorate in quality and it becomes clear that just giving more details is insufficient - a different approach is required.

This new approach consists of initialising the system with previously created components of designs, and allowing the system to evolve the optimal positions for these components. For these prism design problems, suitable components are right-angle prisms. Since the system can itself create right-angle prisms, it seems appropriate to solve more difficult problems in two stages: first evolve the components, then evolve the placement of the components. Hence, for the second stage the system starts with a selection of differently orientated, previously evolved right-angle prisms at random positions relative to each other, see fig. 14.

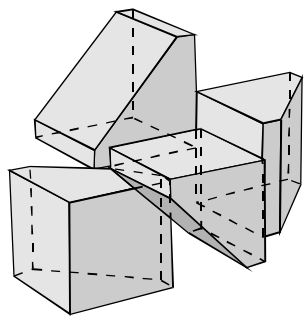

Figure 14 Randomly positioned, previously evolved right-angle prisms.

No genes are fixed, allowing the system to determine not only the positions of the components, but also optimize the components themselves if required. This alternative approach to the creation of conceptual designs was explored by using the system to create two new prisms with not two, but four total internal reflections.

ABBE AND PORRO PRISMS

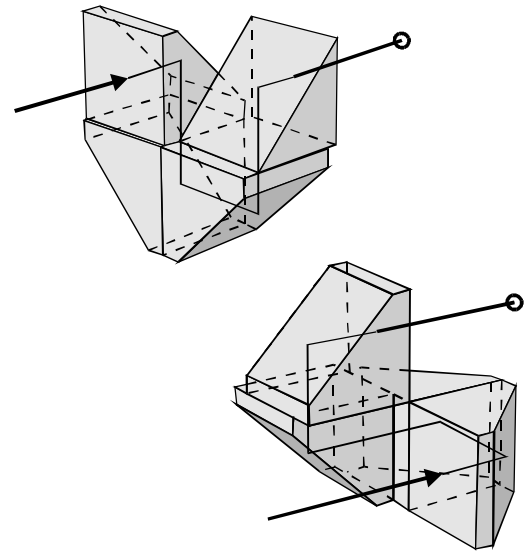

Figure 15 Almost perfect abbe prism (top), almost perfect porro prism (bottom).

The purpose of abbe and porro prisms is to provide a large distance from source to destination for the light to travel, whilst containing the entire path of the light in a small package. The output image must be kept erect (i.e., in the same orientation as the input image). These prisms are typically used in high quality binoculars. The system was first used to evolve various differently orientated right-angle prisms (as described previously). Using populations of 400 individuals evolved for 500 generations in the GA, these components were then successfully arranged into the correct configurations to match the input and output requirements of both the porro and abbe prisms. The designs (defined by four primitives of the representation) were also optimized automatically by the system to fine-tune the precise angles and positions of the reflecting surfaces of both types of prism. Resulting designs were consistently both conceptually good, and accurate in detail, fig. 15.

5.5. Summary of Evolved Results

Despite the fact that the system was used to evolve designs for deceptive problems, the results were all good. Usually any designs that were conceptually flawed were created as a result of an incomplete or flawed design specification within the evaluation software. For simple types of prism (i.e., prisms with at most two total internal reflections), most problems could be overcome by simply giving more information. By specifying additional

requirements such as symmetry, no refraction and that designs should be two-dimensional, the range of acceptable designs becomes more tightly constrained, thus reducing deceptive attractors and increasing the probability of evolving good designs.

However, for more complex types of prism (i.e., prisms with four total internal reflections) providing more information was insufficient. The system was simply unable to avoid the many 'cheating' deceptive attractors to the problem, producing unusual, but unacceptable designs. By using the system in an alternative way, to evolve designs from previously created components, the problem is simplified. Although for such complex designs, the system must manipulate 36 parameters (compared to 18 for the simpler designs), by initialising some of the parameters with data describing right-angle prisms, the initial population begins at much 'closer' points to a good solution. This means that, although all parameters are still adjustable by the system, with less far to travel in the search space, the system has a much greater chance of finding an acceptable solution. Indeed, using this method, conceptually correct and nearly optimal designs of greater complexity were consistently produced.

6. CONCLUSIONS

This paper has demonstrated that evolutionary search is capable not only of optimising designs, but also creating new conceptual designs. A prototype generic computer design system based on a genetic algorithm was described, capable of evolving solutions to a wide range of solid object design problems. T system performs the whole design process, both generating the design concept, and also optimizing the designs.

Having previously applied the system to other simple design tasks [3,4,5], in this paper it was applied to a set of optical prism design problems which are deceptive to evolutionary search, to explore and extend the limits of its capabilities. It was found that for simple deceptive problems, the system could still generate acceptable designs from scratch if additional information on the specification was given. For more complex deceptive design tasks, the system was used successfully in an alternative way, to create new designs from previously created components.

Using the system in these ways, good solutions were evolved to all seven of the prism design problems tackled, despite the deceptive nature of these problems for evolutionary search. The system was made to evolve this range of designs, all with very different geometries, and all conceptually different, purely by changing the design specification in the evaluation software and initialising the system appropriately. Inevitably, however, results are improved and design criteria simpler to specify when the system is applied to other non-deceptive applications [3,4].

Future work will concentrate on improving the capabilities of the system further, by allowing the number of primitives that represent a design to be optimized by the GA in addition to the geometries of the primitives. In this way, the GA can adjust the size of the effective search space itself by adding or removing primitives (and hence the corresponding definition parameters). This can be achieved by using a mutation operator to allow groups of nine coded parameter values to be added or removed from genotypes (thus adding or removing primitive shapes from phenotypes). A crossover operator capable of meaningfully generating offspring from chromosomes of different lengths would then be required. Finally, a simple modification to the system would also allow it to automatically check for features such as symmetry and two-dimensionality, using them if the fitness values of such designs are improved by them.

7. REFERENCES

1. Adeli, H. and Cheng, N., Concurrent Genetic Algorithms for Optimization of Large Structures, *ASCE Journal of Aerospace Engineering*, Vol. 7, No. 3, pp. 276-296, 1994.
2. Bentley, P. J. and Wakefield, J. P., Generic Representation of Solid Geometry for Genetic Search, *Microcomputers in Civil Engineering*, Vol. 11, No. 3, pp. 151-159, 1996.
3. Bentley, P. J. and Wakefield, J. P., The Evolution of Solid Object Designs using Genetic Algorithms, *Proceedings of Applied Decision Technologies* (ADT '95), London England, pp. 391-400, April 1995.
4. Bentley, P. J. and Wakefield, J. P., The Table: An Illustration of Evolutionary Design using Genetic Algorithms, *Proceedings of Genetic Algorithms in Engineering Systems: Innovations and Applications* (GALESIA '95), Sheffield, England, pp. 412-418, Sept. 1995.
5. Bentley, P. J. and Wakefield, J. P., An Analysis of Multiobjective Optimisation in Genetic Algorithms, Submitted to *Evolutionary Computation*, 1996.
6. Brown, E. B., *Modern Optics (Second Edition)*. Reinhold Pub. Corp., Chapman and Hall Ltd., London England, 1966.
7. Dawkins, R., *The Blind Watchmaker*, Longman Scientific & Technical Pub, 1986.
8. Dym, C. L. and Levitt, R. E., *Knowledge-Based Systems in Engineering*, McGraw-Hill Inc, 1991.
9. Dyer, M., Flower, M. and Hodges, J., 'EDISON': an engineering design invention system operating naively. *Artificial Intelligence 1*, pp. 36-44, 1986.
10. Goldberg, D. E., *Genetic Algorithms in Search, Optimization & Machine Learning*, Addison-Wesley, 1989.
11. Goldberg, D. E., Genetic Algorithms as a Computational Theory of Conceptual Design, *Proceedings of Applications of Artificial Intelligence in Engineering 6*, pp. 3-16, 1991.
12. Goldberg, D.E., Deb, K., and Horn, J., Massive multimodality, deception and genetic algorithms, *Proceedings of Parallel Problem Solving from Nature 2*, pp. 37-46, 1992.
13. Holland, J. H., *Adaption in Natural and Artificial Systems*, The University of Michigan Press, Ann Arbor, 1975.
14. Holland, J. H., Genetic Algorithms, *Scientific American*, pp. 66-72, 1992.
15. Meyer-Arendt, J. R., *Introduction to Classical and Modern Optics (Third Edition)*, Prentice-Hall International, Inc, 1989.
16. Michielssen, E., Ranjithan, S., and Mittra, R., Optimal multilayer filter design using real coded genetic algorithms, *IEE Proceedings-J (Optoelectronics)*, Vol 139, No. 6, pp. 413-420, 1992.
17. Parmee, I. C. and Denham, M J., The Integration of Adaptive Search Techniques with Current Engineering Design Practice, *Proceedings of Adaptive Computing in Engineering Design and Control -'94*, Plymouth, England, pp.1-13, 1994.
18. Pham, D. T. and Yang, Y., A genetic algorithm based preliminary design system, *Journal of Automobile Engineers*, Vol 207, No. D2, pp. 127-133, 1993.
19. Rosenman, M. A., A Growth Model for Form Generation Using a Hierarchical Evolutionary Approach. *Microcomputers in Civil Engineering*, Vol 11, No. 3, pp.163-174, 1996.
20. Todd, S. and Latham, W., *Evolutionary Art and Computers*, Academic Press, 1992.
21. Tong, S.S., Integration of symbolic and numerical methods for optimizing complex engineering systems, *Proceedings of IFIP Transactions (Computer Science and Technology)*, Vol A-2, pp. 3-20, 1992.
22. Williams, B. C., Visualising potential interactions: constructing novel devices from first priciples, *Proceedeings of Eighth National Conference on Artificial Intelligence (AAAI-90)*, Boston, Mass, 1990.