8

Designs Evolved by the System

8.1 Introduction

Using the evaluation software described in the last chapter, the generic evolutionary design system was applied to five different types of design problem. This chapter gives the parameter values that were used for the modules of evaluation software and describes the designs evolved by the system for each task.

The design problems were presented to the system during the various stages of its development in order to investigate the current capabilities of the system. To ensure that any new techniques incorporated during the development of the design system for later problems did not reduce the generality of the system, the initial two design problems were also presented for a second time, after the completion of the development of the system.

For each of the five types of design problem, three aspects of the evolved results are assessed. First, designs evolved by the system are assessed in terms of acceptability, i.e. how 'good' a design is, based not only on the fitnesses calculated by the evaluation software, but also on the judgement of humans (from opinions given during informal discussions with researchers, designers and artists). In addition, the evolved *concept* behind each design solution is identified

and explained. As will be shown, although the system itself has no notion of design concept, the process of evolving good designs generates a variety of different types of solution that perform the desired function using different methodologies. Finally, the performance of significant elements of the generic evolutionary design system is assessed.

8.2 Tables

8.2.1 Evolving Early Designs

The problem of evolving a simple table was the first design task to be presented to the prototype system. At that time, the design system used a simple, real-coded GA with no multiobjective capabilities. Hence, in order to allow the GA to evolve designs for this problem, the six fitness values returned by the five evaluation modules: 'size', 'specific mass', unfragmented', 'flat upper surface' and 'supportiveness', were weighted and summed into a single value. Table 8.1 shows the values of the weights used, with 'unfragmented' deliberately being given a large weighting in an attempt to increase its importance. However, because the effective ranges of the criteria were unknown, whether these weights were specifying increased or reduced importance for other evaluation modules was unknown. Each weight was set by trial and error, and took a significant amount of time to perfect, leading to the conclusion that a multiobjective GA was needed (see Chapter Six).

Evaluation Module	Weight
SIZE	1.0
SPECIFIC MASS	6.0
UNFRAGMENTED	100.0
FLAT UPPER SURFACE	1.8
SUPPORTIVENESS	0.5

Table 8.1 Weights used to combine separate fitness values.

The desired extents of evolved designs are specified using twelve parameter values with the 'size' module of evaluation software. For this problem, the desired table size was specified as being no larger than a cube of 90 units, and no smaller than a cube of 80 units, see fig. 8.1. If each unit of measurement corresponded to a centimetre, this specifies designs of about average 'table size'. However, the fact that the desired extents specify cubes is considered more important than the actual parameter values used. In other words, the 'size' module is used more to define desired *relative* x, y, z dimensions, than to define specific real-world measurements.

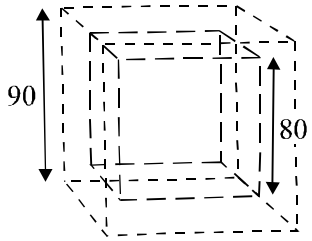


Fig. 8.1 Desired inner and outer extents for tables.

The desired mass for tables was defined as a low value of 5 units. This can be compared to the mass of a typical, initially random solid object, which normally has a mass of about 70 units. The desired flat upper surface was defined as having a width and depth of 80 units (the same as the desired inner extents), and a height of 85 (halfway between the desired inner and outer topmost extents), see fig 8.2.

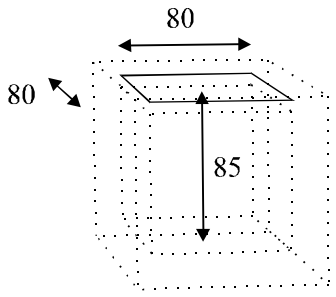


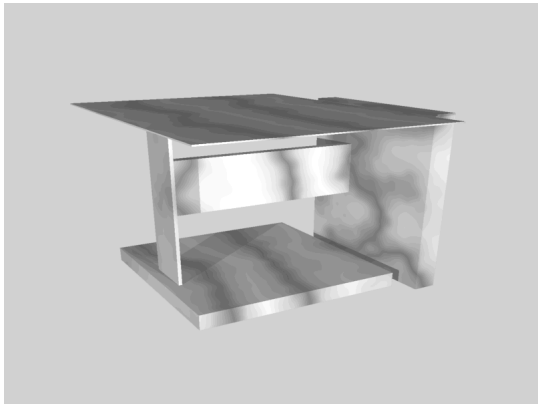
Fig. 8.2 Desired flat upper surface for tables.

Finally, it was specified that the table should be able to support a mass of 10 units (twice the desired mass of the table itself) at the edges of its upper surface, without the design toppling over.

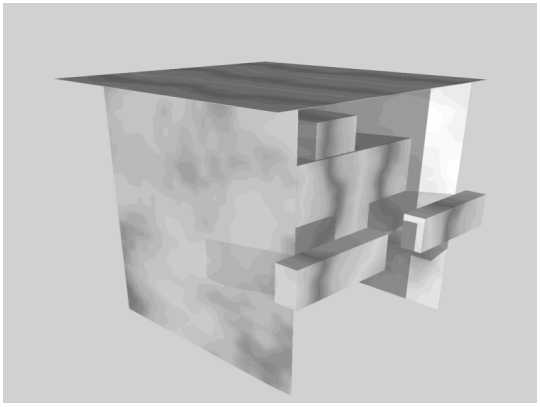
The simple GA used at this stage of the testing had a population size of 100 individuals, a probability of 1.0 that crossover would be used, and a probability of 0.004 that an allele in a chromosome would be mutated (by incrementing or decrementing by a random value). Each design was represented using five primitive shapes (before any reflections) without intersecting planes (the 'slanting surfaces' they provide were not needed for this application). All genotypes were filled with random alleles (i.e. starting from scratch). The GA was allowed to evolve designs for up to two hundred generations.

Figure 8.3 shows six tables evolved by the early prototype system. These images were rendered using the shareware raytracing program: *Polyray v1.7* (developed by Alexander Enzmann) to give a more realistic appearance to designs, compared to the real-time 3D graphical display program described previously. In order to allow *Polyray* to render evolved designs, a simple program to convert the output files generated by the design system into standard constructive solid geometry format representation (Foley, et. al., 1990) was developed as part of this work. Using this representation, a 'stretched cube' with intersecting plane can be constructed from the intersection of a 'box' primitive with a 'disc' primitive of large radius. All table designs were then rendered with a 'white marble' texture.

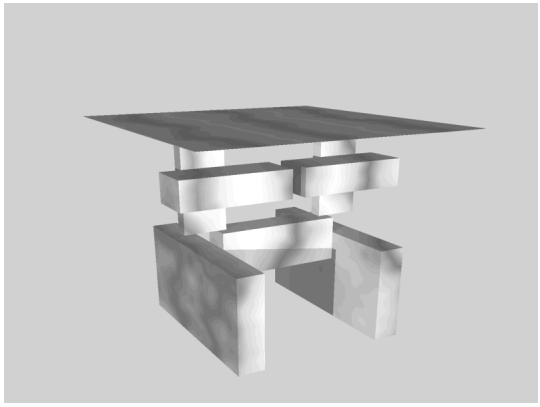
[A] Asymmetrical table



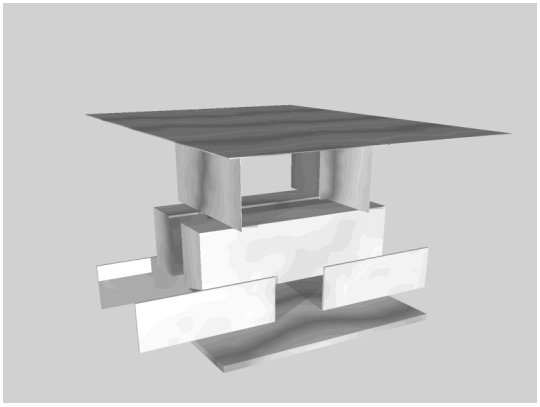
[B] Asymmetrical table



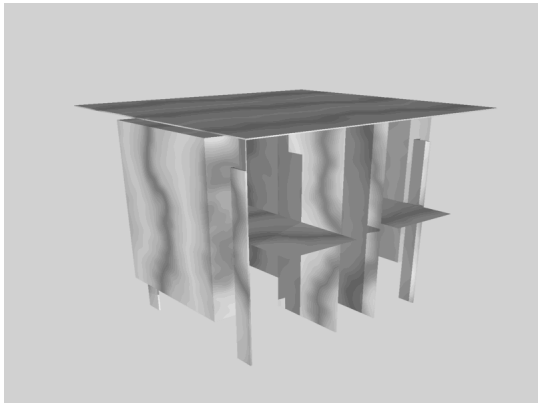
[C] Table symmetrical in $x = 0$



[D] Table symmetrical in $x = 0$



[E] Table symmetrical in $x = 0$ and $z = 0$



[F] Table symmetrical in $x = 0$ and $z = 0$

Fig. 8.3 Six table designs evolved by the early prototype system.

Despite the fact that only a very simple GA was used to evolve designs at this stage, all results were gratifyingly fit, as judged by the evaluation software. For example, design [A] shown in fig. 8.3 has a table top with a depth and width of exactly 80.0, at a height of precisely 85.0. Most designs were judged to be perfect for some criteria and very fit for all others. However, the system did manage to produce some designs that received perfect fitness scores from all five evaluation modules, see design [F] of fig. 8.3. Typically, it was discovered that the introduction of symmetry seemed to improve the ability of the system to evolve fit designs. This is probably because it is easier to locate symmetrical tables that satisfy criteria such as stability and flat upper surfaces, than asymmetrical tables. In other words, a symmetrical table is considerably more likely to have, for example, four legs which provide extra stability (Bentley & Wakefield 1995a/1996a, 1995b).

Whilst most of the table designs are highly unusual and distinctive, they all perform the function of a table correctly, and hence are all acceptable and usable designs. Moreover, all of the tables are based upon a number of sound design concepts. For example, design [A] has most of its mass at the bottom of the design, giving a very low centre of mass, and making the table highly unlikely to topple over. Alternatively, design [B] uses a wide base to increase stability (the bigger the base, the harder it is for the centre of mass to fall outside the area of the base). Design [C] uses yet another method: two side supports to provide the equivalent of a wide base, but also a large centre weight. This weight prevents the centre of mass of the design from being displaced too much by heavy objects placed on the table, and hence makes this one of the most stable designs ever evolved. (This idea is commonly used in washing machines - in the form of a large lump of concrete - to prevent the mass of internally rotating wet laundry from allowing them to become unstable and 'walk' around kitchen floors.)

Design [D] uses a combination of these concepts, with twin supports to provide a wide base, and most of the mass of the table either in the base, or in the two weights below the table-top. Likewise, design [E] has a large base and two massive centre weights to increase stability.

Finally, despite appearing somewhat cluttered, design [F] uses the conventional four legs to provide the equivalent of a large base, and good stability.

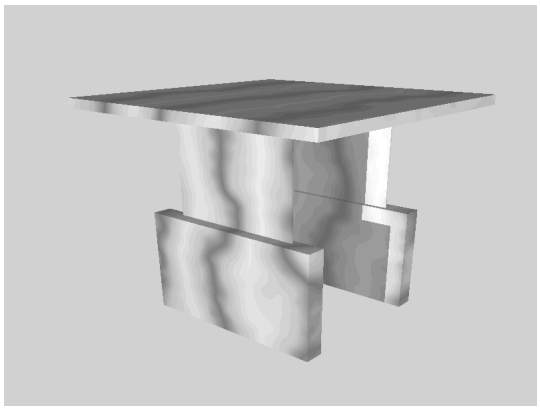
8.2.2 Evolving Designs using the Complete System

This 'table' problem was presented for a second time to the completed generic evolutionary design system, to compare the performance of the final system with the early prototype system. To allow the comparison to be more realistic, the same population size of 100 was used for the external population, and evolution was limited to 200 generations. All other parameters were at their default values, see table 8.2.

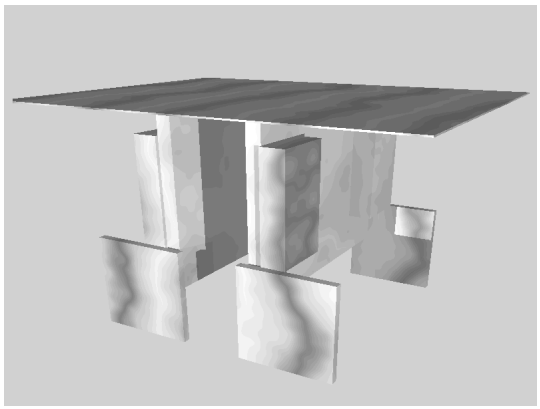
System Parameter	Value
external population size	100
internal population size	80
mutation probability per bit	0.001
mutation probability per primitive	0.01
maximum life span	6
percentage of parents picked from intl. pop.	80 %
maximum no. of generations	200

Table 8.2 System parameter values used for the 'table' design problem.

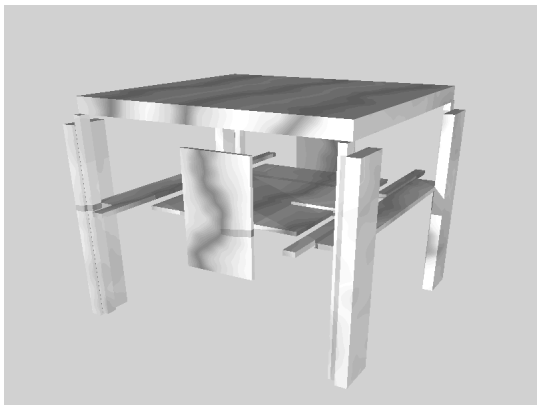
Again, random alleles were used to initialise all genotypes, and the number of primitives in each design was specified as being five (before reflections). However, since the complete design system has the ability to add or remove primitives, this number was not fixed within phenotypes as it was for the prototype system. All designs were required to be symmetrical in $x = 0$ and $z = 0$. Initially, importance values of 1.0 were defined for every criteria except 'unfragmented' which was set at 100.0 to ensure that the system would evolve unfragmented designs first. However, in order to illustrate the effect of importance, the 'mass' evaluation criteria was given a reduced relative importance value of 0.1 for the last few designs. Figure



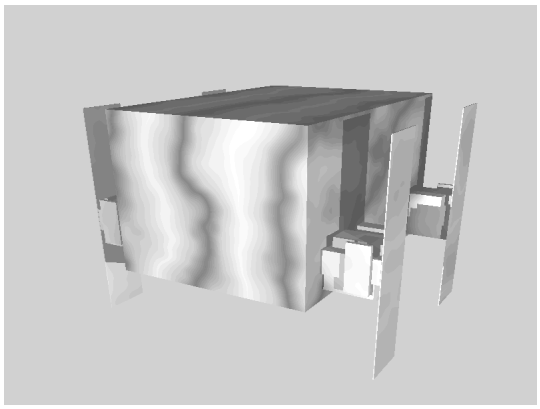
[A]



[B]



[C]



[D]

Fig. 8.4 Four table designs evolved by the complete generic evolutionary design system.

8.4 shows four new tables evolved by the system: [A] to [C] evolved with 'mass' having an importance of 1.0, and [D] evolved with 'mass' having a reduced importance of 0.1 (rendered as before).

Again, very fit designs were consistently evolved, with most criteria either being completely, or very nearly satisfied. The exception, shown in fig. 8.4, was design [D], which is obviously somewhat too massive. This clearly demonstrates the effect importance can have on evolution - by making the 'mass' criterion ten times less important than all for other criteria (which were set at 1.0), the system automatically ensured that the fitnesses for these other criteria were at

least ten times better than the fitness for mass. Indeed, all of the designs that were produced using this reduced importance value were similarly massive.

The three designs [A] to [C] that were evolved using a more appropriate importance level for 'mass' can be considered acceptable, since they all correctly perform the function of a table. As before, the tables are all highly original and distinctive in appearance.

These new evolved designs used concepts such as lowering the centre of mass (design [A]) or using four legs (designs [B], [C] and [D]) in order to increase the stability of the tables. Because the complete system had the ability to remove unneeded primitives, the designs were typically less cluttered in appearance, compared to the symmetrical designs generated by the prototype system. However, perhaps the most significant difference between the evolution of tables with the prototype system and the complete system was the number of generations required to generate acceptable designs. As mentioned earlier, the prototype system required around two hundred generations to evolve the tables. The complete generic evolutionary design system typically required less than one hundred generations to produce designs of equal quality. Moreover, since the complete system only evaluated 80 phenotypes per generation, this represents a considerable reduction in computation time.

8.2.3 Summary of the Evolved Tables

The 'table' design problem has shown that it is possible to use a computer to evolve entirely new designs, from scratch. These designs are all acceptable and usable tables, based upon sound design concepts, if somewhat unorthodox in appearance. This design task has also demonstrated the benefits of using symmetry, which was found to help the system to evolve fitter designs. Setting the weighting values to allow separate fitness values to be summed for the prototype system was found to be laborious and difficult. In contrast, the complete multiobjective design system required no such weights, and was able to generate designs as fit or fitter than the prototype. The use of 'importance' was demonstrated, and a reduction in the

number of generations needed to evolve acceptable designs was observed when using the complete system.

8.3 Steps

8.3.1 Evolving Early Designs

The second design task to be presented to the system was to evolve a set of steps. A good solution to this design task would be a reasonably light set of steps, capable of supporting the weight of a person on each step in turn (suitable for use, for example, in a library). Three steps were desired. This problem was designed in order to test six different multiobjective ranking methods within the prototype evolutionary design system (see Chapter Six). At that time the system used a simple GA with binary coding (having changed from real coding - see Chapter Five), but without overlapping populations or the ability to vary the length of chromosomes.

The 'steps' design problem used the evaluation modules: 'size', 'mass', and 'unfragmented' once, and the modules 'supportiveness' and 'flat surface' three times. It was defined that the desired size of each set of steps should be the same as the tables. The desired mass was specified as 4 units - slightly less massive than the desired value of 5 that was defined for tables. The three desired flat surfaces were defined as having a width of 80 and depth of 30 units, and with three stepped centre positions of (0,-12,-30), (0,16,0), (0,44,30), see fig 8.5.

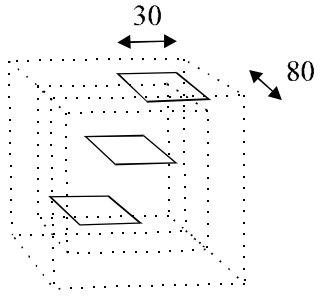
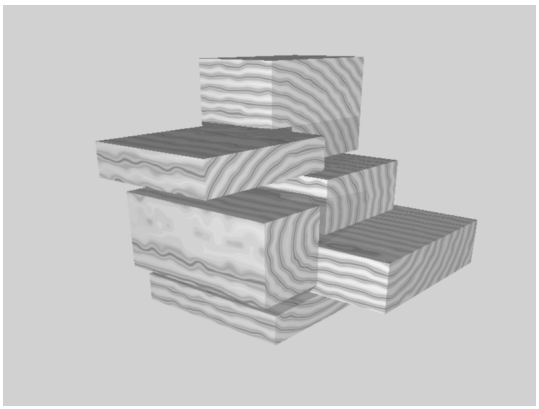


Fig. 8.5 Desired flat surfaces for steps.

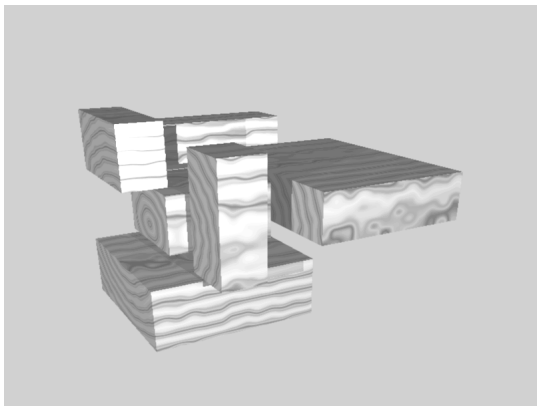
Finally, it was specified that each flat surface should be able to support an object with a very large mass of 500 units at each of its edges without the whole design toppling over. In other words, the set of steps should be capable of supporting even the heaviest of people on each step without becoming unstable.

The simple GA used at this stage had a population size of 100 individuals, a probability of 1.0 that crossover would be used, and a probability of 0.001 that a single bit in a chromosome would be mutated. Each design was represented using six primitive shapes, without intersecting planes and without using reflections for symmetry. All genotypes were filled with random alleles (i.e. starting from scratch). The GA was allowed to evolve designs for up to five hundred generations.

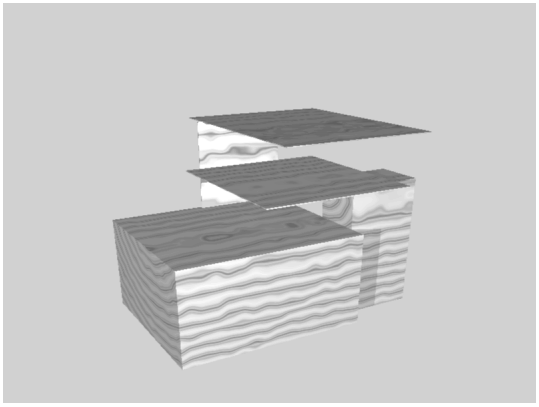
In total, over one hundred designs were evolved using this prototype version of the system for the 'steps' problem, using each of the six multiobjective ranking methods within the system in turn. Figure 8.6 (designs [A] to [D]) shows typical designs evolved using three of the different techniques (rendered with *Polyray* using a 'wood' texture).



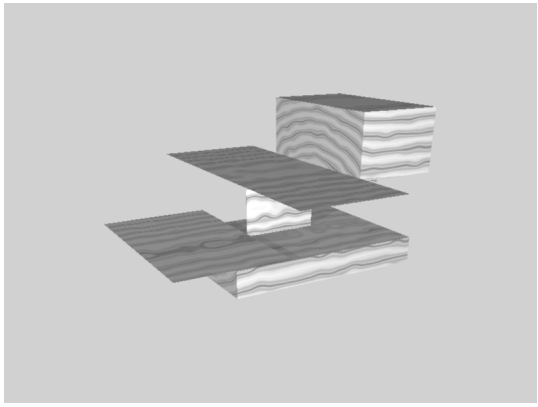
[A] Typical evolved design using NDS



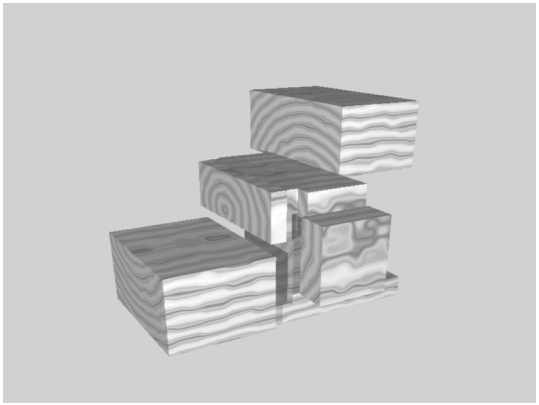
[B] Typical evolved design using WMR



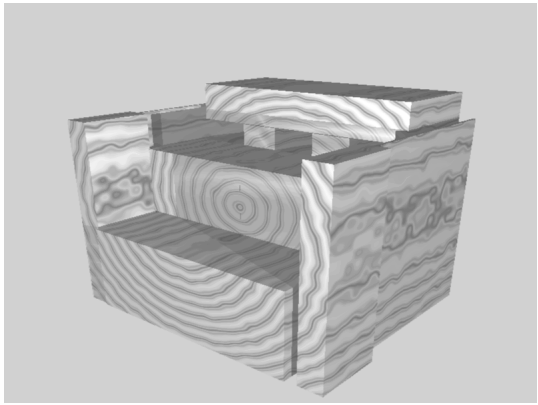
[C] Typical evolved design using SWGR



[D] Typical evolved design using SWGR



[E] Evolved design using complete system



[F] Evolved design using complete system

Fig. 8.6 Six sets of steps evolved by the system.

As is apparent by the images shown in fig. 8.6, the quality of typical designs evolved by different multiobjective methods varied enormously. The methods NDS, WMR and WAR all consistently produced very poor designs, most unrecognisable and not functioning as steps, see designs [A] and [B] shown in fig. 8.6. It became clear that only the methods SWO, SWR and SWGR permitted the system to produce recognisable designs of steps, because only these methods allowed the GA to converge on a small number of acceptable designs, defined by importance values (Bentley & Wakefield, 1996f). Of these three, the new method SWGR consistently allowed the system to produce some of the best solutions to the 'steps' problem, and was thus chosen for the generic evolutionary design system (see Chapter Six).

Design [C] in figure 8.6 shows a set of steps created using SWGR. This design has a heavy base to provide a low centre of gravity and three well-defined steps. The system has 'cheated' in an intelligent way with the top step, placing it further forward than was desired, in order to increase the stability of the steps when extra mass is placed on the top step. Alternatively, design [D] has three well-defined and correctly positioned steps, with a large base (which does extend below the top step) to provide stability. This design uses a top step of large mass in order to counter-balance the design when extra mass is placed on the bottom step, and thus prevent the design from tipping forwards. Because of the lack of stress-analysis, both these designs have two steps optimised to be very thin, potentially causing problems unless a material of appropriate strength was used for them (e.g. metal).

8.3.2 Evolving Designs using the Complete System

As with the 'tables' design task, the 'steps' problem was presented for a second time to the completed generic evolutionary design system, to compare the performance of the final system with the prototype system. The system was permitted to evolve designs for up to 500 generations, with all other parameters being the same as for the 'tables' problem (see table 8.2). Random alleles were used to initialise all genotypes, and the initial number of primitives in each design was specified as being six (before reflections). Importance values of 1.0 were defined for all criteria except for the three flat surfaces which were increased to 2.0 and

'unfragmented' which was set at 100.0 to ensure that the system would evolve unfragmented designs first. A number of both asymmetrical designs and designs symmetrical in $x = 0$ were evolved by the complete system, see [E] and [F] in fig. 8.6.

Despite the fact that this problem is substantially more difficult to evolve solutions for than the 'tables' problem (because of the higher number of partially-conflicting criteria; Bentley & Wakefield, 1996f), fit designs were consistently evolved by the system. For example, design [E] in fig. 8.6 shows an asymmetrical design with three well-defined, positioned and supported steps, and a large base (slightly obscured in the image) which provides stability. Alternatively, design [F] is an extremely good symmetrical design of steps, using 16 primitives (having been increased by mutation from 12). This design has twin side supports to provide excellent stability, as well as having the bottom step form part of the base, and twin supports under the top step that also form part of the base. Behind the three steps the design is largely hollow to reduce the mass. In addition, the thickness of the side supports is increased at the front of the design, in order to counter-balance any excessive extra weight placed on the top step. Once again, the complete evolutionary design system was able to evolve such fit designs in considerably fewer generations, compared to the prototype system (e.g. around 200 generations instead of 500).

8.3.3 Summary of the Evolved Steps

The 'steps' design problem further demonstrates the ability of the generic evolutionary design system to evolve designs from scratch. This problem helped identify the most suitable multiobjective ranking method to use within the system. Moreover, despite the increased difficulty of this problem, the complete system was able to evolve some intricate and acceptable solutions in fewer generations than was required by an earlier prototype system.

8.4 Heat Sinks

8.4.1 Evolving the Designs

The 'heat sink' design problem was created in order to investigate whether the system could successfully vary the number of primitives in a design and hence optimise the number of primitives in addition to their dimensions.

Four modules of evaluation software adequately specify the function of a heatsink: 'size', 'specific mass', 'unfragmented' and 'specific surface area' (see Chapter Seven). Heatsinks were specified as having a desired size no larger than a box of dimensions 100 by 100 by 50 units, and no smaller than a box 98 by 98 by 48 units, see fig. 8.7. If each unit corresponded to half a millimetre, this defines an object of about the size of a heat sink for an average processor. However, as usual, the 'size' module was used more to define the fact that designs should have a height half the size of their width and depth, than to define precise measurements.

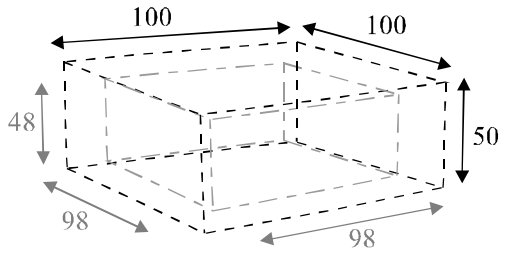
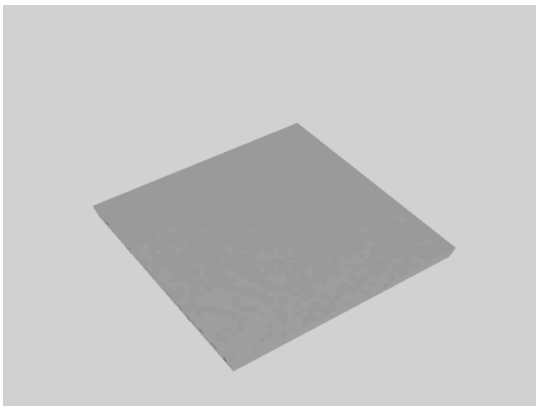


Fig. 8.7 Desired inner and outer extents for heat sinks (not to scale).

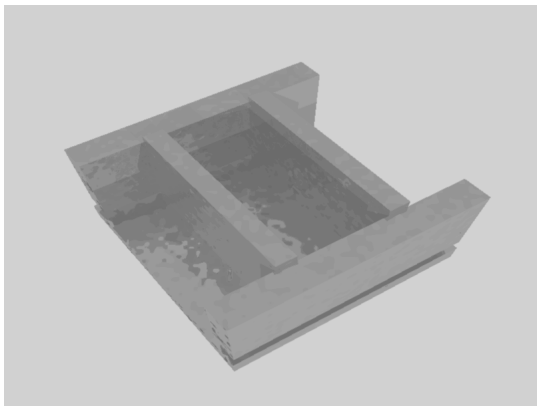
Although it does not matter how heavy a heatsink is, ideally the amount of metal used to make each heatsink should be minimised, in order to minimise the cost. To achieve this, a low desired mass of 10 units was specified. Finally, a huge desired surface area of 99999.9 was specified (to define, in effect, that the surface area should be maximised).

Heatsinks should conduct and radiate heat away from the processor evenly. In other words, any heatsink that allows one part of the processor to heat up more than another, is a poor design. Consequently, to ensure that all the evolved designs would conduct heat in an even and symmetrical manner, all heatsink designs were defined as being symmetrical in $x = 0$ and $z = 0$.

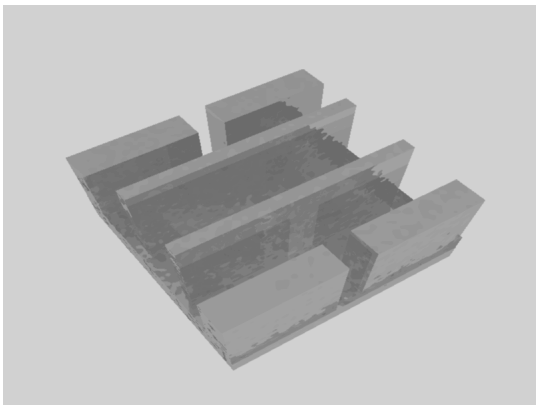
In addition, a common feature of all heatsinks is a solid, flat base, used to cover the processor that is to be cooled. This feature is a fundamental requirement of any heatsink design and cannot be altered in size without reducing the ability of the design to conduct heat from the processor. Since it seems likely that anyone wishing to create a heatsink will know the dimensions of the processor to be cooled, and hence the corresponding dimensions of the obligatory base, there seems little reason to evolve this base. Consequently, for this design problem, a fixed base placed just below the desired bottom extent of the design was defined by initialising the alleles of one primitive in every genotype. This primitive was then declared inflexible (to prevent any evolving primitives from 'squashing' it) and its genes were all fixed (to prevent evolution from modifying its dimensions). Because it was inflexible (and had fixed values) it was automatically also declared 'non-mutatable' by the system (see Chapter Five) and hence could not be split or deleted by mutation. Since all designs were specified as being symmetrical using two reflections, this initialised primitive in every genotype was mapped onto four fixed primitives in every phenotype. Figure 8.8 [A] shows the fixed, inflexible base on which all heatsink designs were evolved.



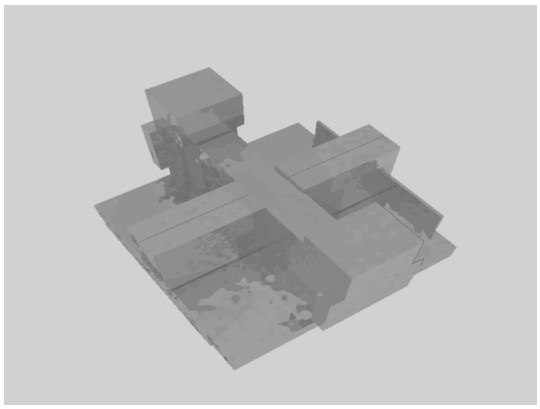
[A] 4-primitive user-defined fixed base



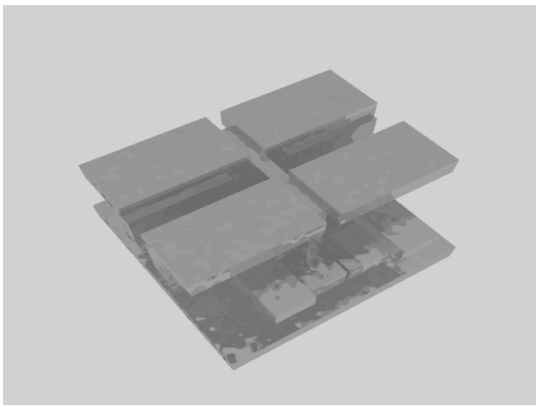
[B] Simple 12-primitive heatsink design



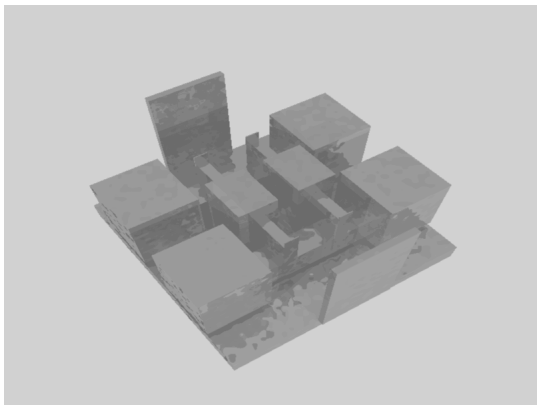
[C] 16-primitive heatsink design



[D] 28-primitive heatsink design



[E] 28-primitive heatsink design



[F] 32-primitive heatsink design

Fig. 8.8 Heatsink designs evolved by the generic evolutionary design system.

The complete generic evolutionary design system was used to evolve heat sinks (this problem was not presented to any earlier versions of the system). System parameters were as shown for the 'tables' design problem in table 8.2. The system was permitted to evolve designs for up to 200 generations and the genotypes of all individuals were initialised with two random primitives and one fixed primitive (the base), corresponding to initial phenotypes with twelve primitives (after two reflections for symmetry). Primitives were used without intersecting planes for this application. It should be noted that the maximum number of primitives permitted in a phenotype was limited to thirty-two (the default value). Equal importance values of 1.0 were used for all criteria, except for 'unfragmented', which was increased to 100.0 to ensure that all designs were whole. Figure 8.8 shows five heatsink designs evolved by the system (rendered with a 'bumpy metal' texture).

When the number of primitives in designs was defined as being invariable (using a Boolean command in the initialisation file), the system was consistently able to produce a number of reasonably fit designs of heatsinks, e.g. see [B] in fig. 8.8. Nevertheless, having a fixed number of primitives available for each design severely limited the system when attempting to maximise surface area. When this number was permitted to be variable in designs, the system dramatically increased the number of primitives, often close to or exactly on the maximum permitted in phenotypes.

Designs [D], [E] and [F] in fig. 8.8 show original and unusual heatsink designs evolved using large numbers of primitives. All have large surface areas, created by adding more and more primitives to the designs, and by positioning those primitives to minimise the amount each primitive touches another. This generates a detailed, uneven surface - a method often used in nature to increase the surface areas (e.g. the surface of coral). Alternatively, design [C] shows an alternative type of evolved heatsink design, having only sixteen primitives. This design uses the conventional concept of a number of upright slats in a row in order to increase the surface area - making this design closely resemble a conventional human-designed heatsink.

8.4.2 Summary of the Evolved Heat Sinks

The 'heat sink' design problem demonstrates the ability of the evolutionary design system to automatically modify the number of primitives in a design, in addition to optimising the shape and position of primitives. This design application used hierarchical crossover to its full, and detailed testing showed that this new genetic operator does always produce meaningful offspring from two parent designs with different numbers of primitives. Finally, this problem has shown that the system can successfully evolve designs around (or in this case, on top of) fixed, inflexible sections of designs, without modifying these user-defined portions.

8.5 Optical Prisms

8.5.1 Evolving the Designs

The next design application to be presented to the system was to evolve a number of different types of optical prism. As described in Chapter Seven, the prism design problems were chosen for three reasons. First, to demonstrate that the system can successfully optimise the orientation of clipping planes of primitives. (The previous three design problems did not require the use of primitives with intersected planes.) Second, to investigate how the performance of the system and the completeness of the functional specification of designs could be improved in order to overcome the hard, or even deceptive problem (Bentley & Wakefield, 1996e) of correctly evolving the shape of an optical prism. Third, to demonstrate the evolution of designs using previously evolved components, in addition to evolving new designs from scratch.

All optical prism problems were presented to the complete generic evolutionary design system. Initially, five different types of prism problem were presented, in turn, to the system. Beginning with the right-angle prism, the complexity of each type of prism required was increased. For any prism which the system found 'hard' to design, methods to encourage the evolution of more acceptable designs were explored. All designs except for the right-angle prisms were defined by two primitives of the solid object representation, with the number of primitives in a design

being specified as invariable for these applications. Fig. 8.9 shows typical examples of initial, random designs prior to evolution.

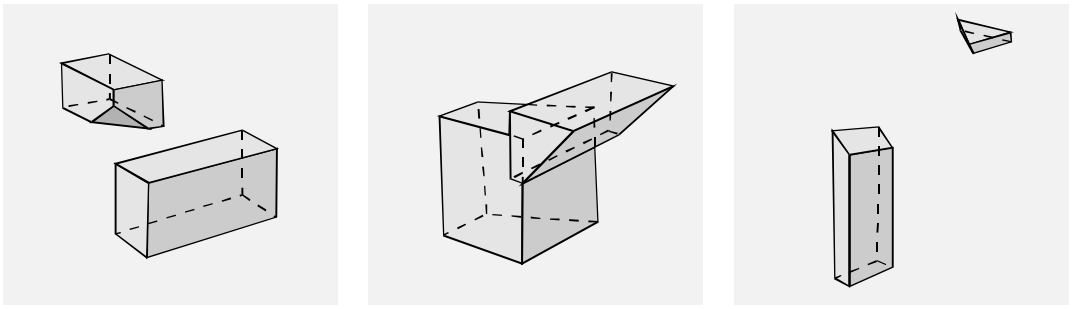


Fig. 8.9 Three examples of initial two-primitive random designs.

All prism design problems were specified using the evaluation modules for 'unfragmented', 'intersected' and 'size' once per design, to define that all prisms should be unfragmented, should have every primitive intersected by its plane, and should have a size within user-specified extents. In addition, the 'ray-tracing' evaluation module was normally used five times per design with different parameters, to specify five different input light rays and five corresponding desired output rays. By repeatedly using this module to trace the reflections and refractions of five light rays through each design, the orientation and size of a virtual image seen through the prism could be judged. Different prism design problems were specified by changing the parameter values defining desired size and input and output rays.

External population sizes used within the GA were varied from 100 to 400 (with the internal population size at 80% of the external size), depending on the difficulty of the prism design 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). Importance values were normally set equally for all criteria except for 'unfragmented' which was given increased importance as usual. All other system parameters were at their default values.

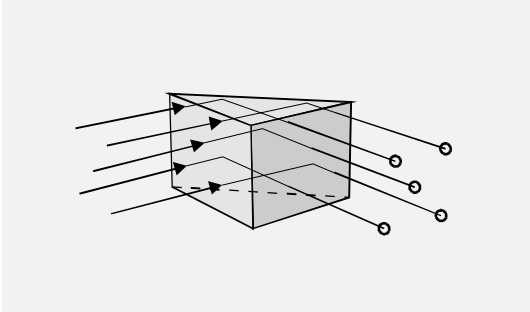
At least twenty test runs for each prism design task took place, with typical results for each being given. Figures 8.10 and 8.11 show example design solutions for each of the five

problems. Input rays are shown by arrows, output rays are displayed terminating with circles to show the positions of the light destination points. (A display program was created to calculate and display the paths of light rays through designs using the 'ray-tracing' module of evaluation software.) For clarity, not all five rays are shown for every design.

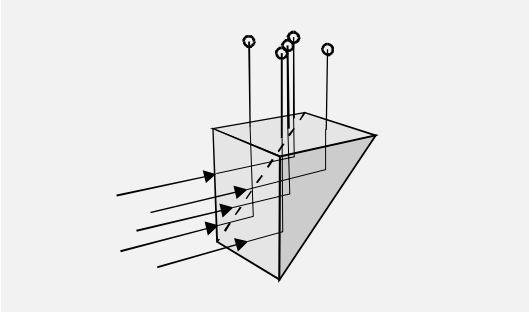
[i] RIGHT-ANGLE PRISM

The function of a right-angle prism is to reflect light by exactly 90 degrees. Right-angle prisms were desired to be no larger than a cube of dimensions 90 units and no smaller than a cube of dimensions 80 units. All five input rays were defined as initially travelling in the direction: < 1, 0, 0 > (using a direction vector < *x*, *y*, *z* >, where a positive *x* value is in the direction left to right) from positions 200 units left of the origin (see fig. 8.10, design [A]). The desired direction of the five corresponding output rays was defined as either < 0, 0, 1 > (i.e. from the back to the front) or < 0, 1, 0 > (i.e., straight up). Likewise projection planes and desired points of intersection were either placed above, or in front of the design.

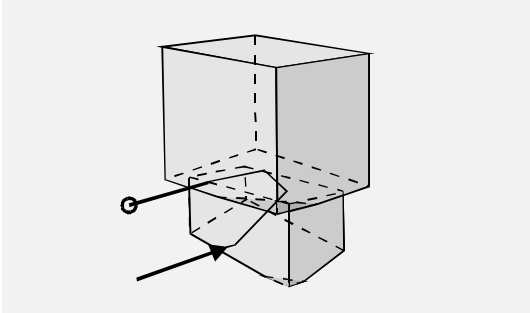
Figure 8.10 [A] and [B] shows the two correspondingly oriented right-angle prisms, using a single primitive of the representation, that were consistently evolved for this problem. All evolved designs correctly used total internal reflection rather than refraction to alter the paths of the light rays. All designs were accurately evolved and were either perfect for all criteria, or very nearly perfect. For example, the angle in the *x*-*z* plane of the intersecting plane of [A] used by the system to internally reflect light was evolved as -2.357 radians, compared to the perfect value of -2.356. Since the binary coding permits the system to evolve values to an accuracy of 1/128 (see Chapter Five), in this case, the system evolved a value as close to the perfect value as possible.



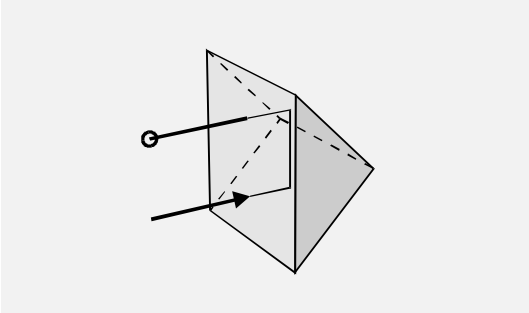
[A] Right angle prism



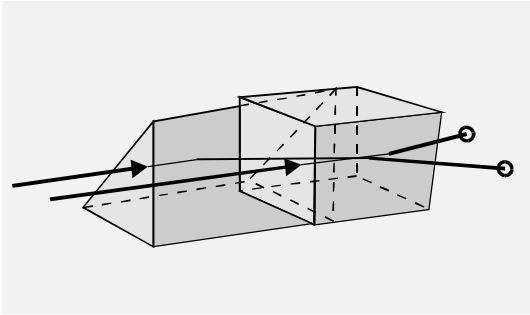
[B] Right angle prism



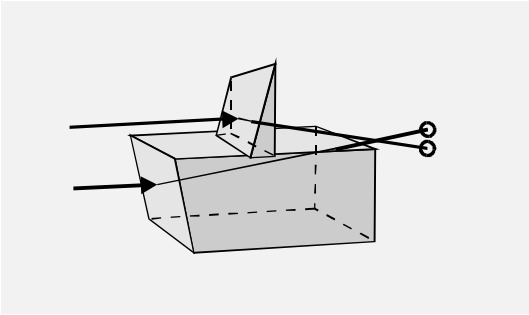
[C] Roof prism attempt



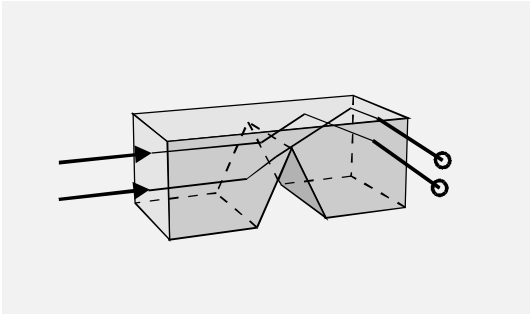
[D] Perfect roof prism



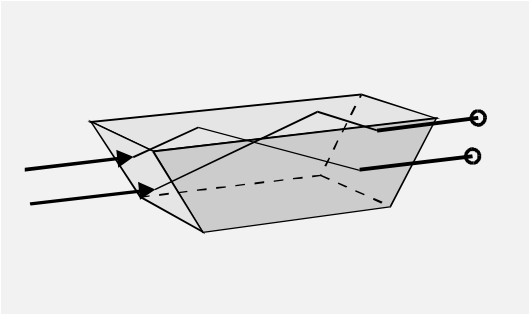
[E] Derotating prism attempt



[F] Derotating prism attempt



[G] Variation of 'K' derotating prism



[H] Derotating 'dove' prism

Fig. 8.10 The first three types of optical prism evolved by the system.

[ii] ROOF PRISM

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). To specify this problem, the input rays to designs where given initial directions and starting positions identical to those for the right-angle prism problem (travelling left to right). However, the direction of the desired output rays was defined as < -1, 0, 0 > (i.e. right to left) with the projection plane being placed to the left of the design (i.e., equation: x - 200 = 0). Care was taken to define desired points of intersection, such that a ray beginning below another ray before entering the design should finish above that ray after leaving the design (i.e. an upside-down image). Designs were desired to be no larger than a box of height 120, width and depth 90 units, and no smaller than a box of height 100, width and depth 80 units.

Initial results for this problem used ingenious combinations of refraction and reflection to achieve good, but not perfect, results, see fig. 8.10, [C]. Typical imperfections consisted of inaccurately directed output rays and the use of refraction, which would 'blur' the colours of a real output image. However, by specifying that symmetrical designs (in y = 0) were required, the system evolved perfect roof prisms every time, see fig. 8.10 [D]. Once again, the designs correctly used total internal reflections and evolved the orientation of the intersecting planes with a high degree of accuracy.

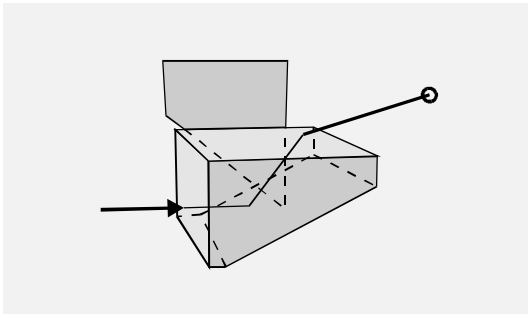
[iii] DEROTATING PRISMS

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, and to allow the 'ray-tracing' evaluation module to be used without modification, a design that turned an image upside-down was specified (a common feature of many derotating prisms). This was specified by defining that the output light rays should have their direction unchanged (travelling left to right), but that the two top light rays should finish at the bottom, the two bottom rays

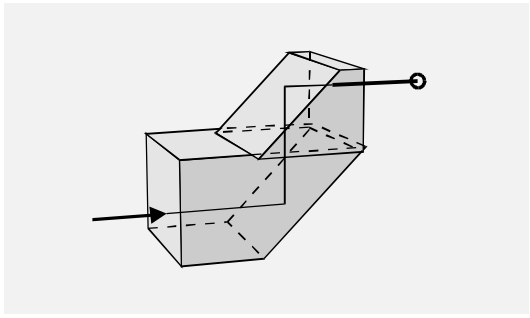
should finish at the top, and the centre ray should remain at the centre, after passing through the design. Undesired points of intersection were also defined, to encourage this reflection of the position of rays. Finally, designs were desired to be no larger than a box of height and depth 50, and width 100 units, and no smaller than a box of height 45, width and depth 80 units.

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, see fig. 8.10, [E] and [F]. Although the designs 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.

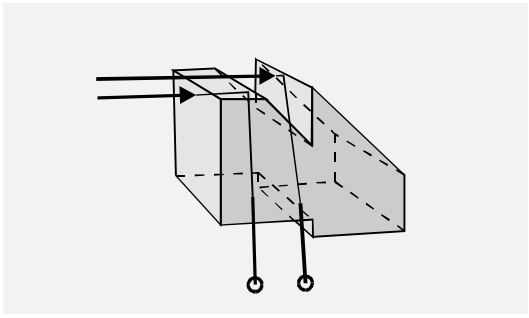
However, by specifying that symmetrical designs were required, most of these 'cheats' became unavailable, and the system successfully evolved some nearly perfect derotating prisms. Figure 8.10 [G] shows an unusual variation of a 'K' prism which uses refraction and only a single reflection (a normal 'K' prism reflects three times, with no refraction). Although the output rays are bent by refraction, this is a fit and acceptable derotating prism. Figure 8.10 [H] shows a nearly perfect traditional 'dove' derotating prism, evolved by the system.



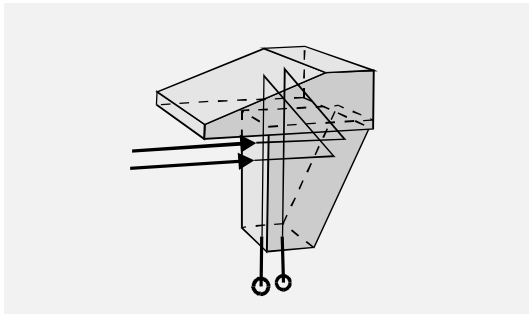
[A] Rhomboid prism attempt



[B] Nearly perfect 'rhomboid' prism



[C] Penta prism attempt



[D] Nearly perfect 'penta' prism

Fig. 8.11 Two 'difficult' types of prism evolved by the system.

[iv] RHOMBOID PRISM

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 (these prisms are used in less expensive binoculars). This was specified by defining that the output rays of light should be travelling in the same direction as the input rays (from left to right), but that the points of intersection should be a distance of 60 units above the initial starting points of the rays. Care was made to define that a ray beginning below a second ray should end below the ray (i.e. the image should be the right way up). Designs were desired to be no larger than a box of height 120, width and depth 90 units, and no smaller than a box of height 100, width and depth 80 units.

Unfortunately, this problem has many deceptive attractors (Deb et. al., 1993) to it, meaning that the typical design produced is as shown in fig. 8.11 [A]. This type of design approximates

the desired function of the prism, but uses a different and incompatible method, i.e. reflection and refraction instead of two reflections. If such a prism was used in a pair of binoculars, the refraction would cause the images to have unacceptably blurred colours. Such unacceptable designs were evolved for this problem because it is statistically more likely for the simpler 'cheating' designs to appear in populations, compared to functionally acceptable solutions. Because 'cheating' designs are more common in populations, they tend to dominate evolution and lead to convergence of the system to unacceptable designs (Bentley & Wakefield, 1996e).

In an attempt to overcome this problem, additional problem-specific knowledge was introduced to the evaluation software (i.e., to reshape the solution-space, removing the unwanted peaks of high fitnesses corresponding to deceptive attractors). 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 (an option of the 'ray-tracing' module), and also specifying that the designs should be two-dimensional to simplify the search-space, see fig. 8.11 [B]. 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). In addition, the importance of the emerging light trvelling in the desired direction was doubled to 2.0. With these extra measures, the system was consistently able to evolve nearly perfect designs, functioning as rhomboid prisms.

[v] PENTA PRISM

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 problem was specified by defining that the rays of light should initially travel in the direction < 1, 0, 0 > (i.e., left to right), and that the desired output rays should emerge travelling in the direction < 0, -1, 0 > (i.e. down). Care was taken to ensure that the desired output intersection of the rays were positioned correctly, i.e. a ray beginning below another ray should emerge to the left of that ray. Undesired points of intersection were

specified to encourage this. Finally, designs were desired to be no larger than a cube of dimensions 100 units, and no smaller than a cube dimensions 80 units.

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, see fig. 8.11 [C]. The image is split by these, placing the top half at the right and the bottom half at the left, but leaving both halves upside-down. 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 consistently able to evolve fit and acceptable solutions to this problem, e.g. see fig. 8.11 [D]. (Additionally, the system was provided with the knowledge that the design was two-dimensional, as described earlier.)

8.5.2 Evolving Designs from 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 'hard' or deceptive design problems. However, when the difficulty of the problems was increased further, the designs of the system deteriorated in quality and it became clear that just giving more details was insufficient - a different approach was required (Bentley & Wakefield, 1996e).

This new approach consisted 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 were right-angle prisms. Since the system could itself create right-angle prisms, it seemed 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 started with a selection of differently orientated, previously evolved right-angle prisms at random positions relative to each other, see fig. 8.12.

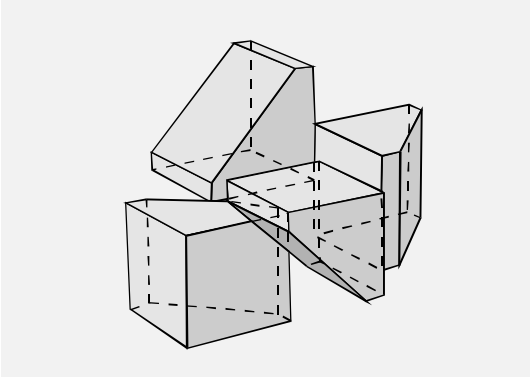


Fig. 8.12 Randomly positioned, previously evolved right-angle prisms.

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

[vi] ABBE AND PORRO PRISMS

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.

First, the system was used to evolve a number of differently oriented right-angle prisms, as described previously. The alleles defining the shape (but not the position) of each type of right-angle prism were then used to initialise the generic evolutionary design system for the two new and more difficult prism design tasks. The function of these desired designs were specified by defining that the direction of the output rays of light should match the initial direction of the input rays of light (travelling left to right), whilst the total distance travelled by the light through the designs should be maximised (as well as specifying appropriate desired points of intersection as usual). In addition, designs were desired to be no larger than a cube of dimensions 160 units, and no smaller than a cube dimensions 140 units.

Using external populations of 400 individuals evolved for 500 generations in the GA (all other system variables unchanged), 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 a fixed number of four primitives of the representation) were also optimised 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, closely resembling conventional designs of abbe and porro prism, see fig. 8.13.

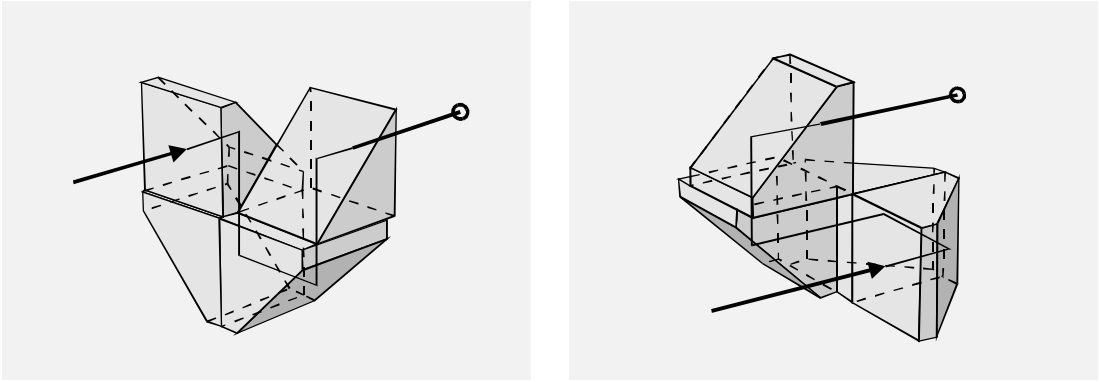


Fig. 8.13 Almost perfect abbe prism (left), almost perfect porro prism (right).

8.5.3 Summary of Evolved Prisms

Despite the fact that the generic evolutionary design system was used to evolve solutions to 'hard' prism design problems, the results were all good. The system was able to evolve the orientation of intersecting planes with a high degree of accuracy in order to guide the light through designs, making use of refraction and reflection. For prisms with at most two total internal reflections, any problems caused by deceptive attractors could be overcome by simply giving more information. By specifying additional requirements such as symmetry, importance, 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 (Bentley & Wakefield, 1996e).

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 problems, producing original, 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 genes (compared to 18 for the simpler designs), by initialising some of the genes with data describing the shape of previously evolved 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 prism designs of greater complexity were consistently produced.

8.6 Streamlined Designs

The final group of design problems presented to the complete generic evolutionary design system was to evolve a range of different streamlined shapes. This is perhaps the most realistic set of applications presented to the system, since creating aerodynamic or hydrodynamic designs is a common and difficult problem faced by designers. Moreover, since such problems typically take the form of the creation of a streamlined shape around a fixed chassis, this group of problems allows the demonstration of the ability of the system to evolve shapes around fixed, inflexible 'skeletons'.

In total, five different streamlined design problems were presented to the complete system: the evolution of the fronts of trains and boats, entire hulls of boats, aerodynamic saloon cars, and the evolution of an aerodynamic sports car. As described in Chapter Seven, all these design tasks were evaluated using four modules of evaluation software (all used just once): 'size', 'unfragmented', 'must have vertices', and 'particle flow simulator'.

8.6.1 Evolving the Fronts of Trains

The first of the streamlined design problems consisted of evolving an aerodynamic front to a 'train'. This was performed by initialising the system with an inflexible primitive of fixed size and position, and using the system to evolve the position and size of primitives in front of this fixed block in order to minimise the force generated by particles hitting the front of the design. A single enclosing volume was used with the 'particle flow simulator' module to specify desired forces generated by particles on the entire design, see fig. 8.14. A desired force of 0.0 was required in the *x*-direction (i.e. no wind resistance), a total desired force of 0.0 was required in the *z*-direction (i.e. no forces pushing the design to one side), and a small force was required in the *y*-direction (i.e. the 'wind' should push train designs downwards slightly). Finally, a desired maximum and minimum size for the front of the trains was specified as usual.

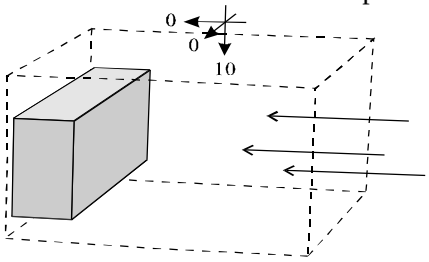
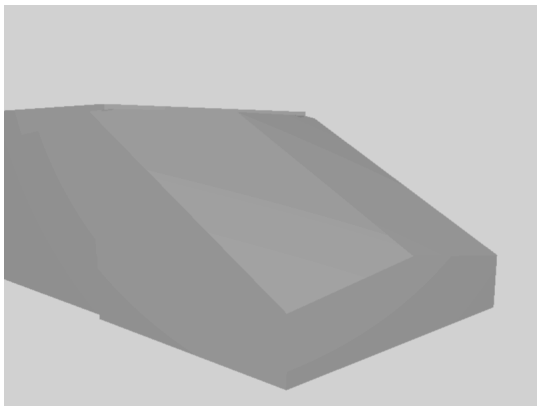
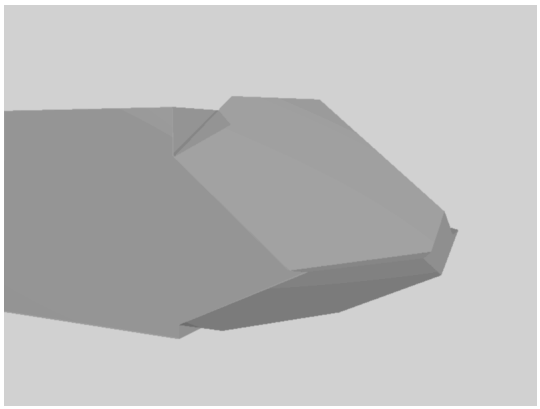


Fig. 8.14 A fixed block forms the skeleton, and a single enclosing volume is used to define the forces required on the train designs.

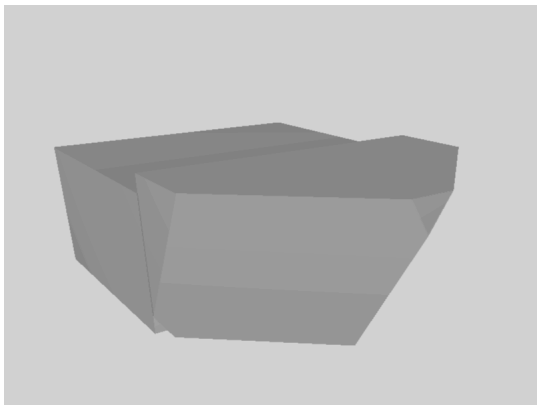
For reasons of aesthetics, the designs were specified as being symmetrical about $x = 0$. A range of train designs were then evolved, using a fixed number of either two primitives or four primitives to 'streamline' the fixed 'skeleton'. Importance values were set equally, except for the usual increased value for 'unfragmented'. All system parameters were set at their default values (i.e. external population size of 200, internal population size of 160, running for 500 generations). Figure 8.15 [A] and [B] shows two typical examples of train designs evolved by the system (rendered using *Polyray* with a 'steel' texture).



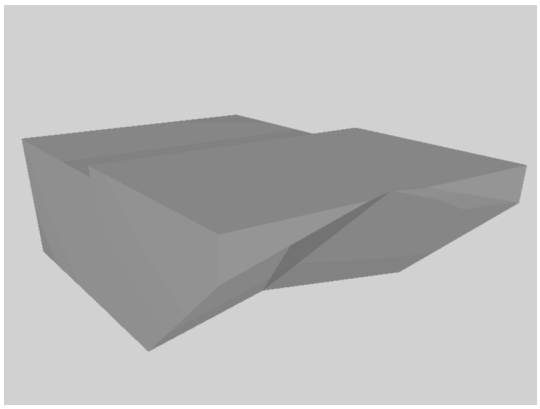
[A] Evolved 2-primitive train front



[B] Evolved 4-primitive train front



[C] Evolved 2-primitive boat front



[D] Evolved 2-primitive boat front

Fig 8.15 Streamlined fronts of trains and boats evolved by the system.

All designs evolved were highly fit and all used sound principles to reduce the 'wind-resistance' of the fixed and inflexible part of the designs. For example, fig. 8.15 [A] shows a train design with a down-sloping front, which effectively reduces the force generated by particles hitting the front of the design, whilst also generating the required down-force. Alternatively, fig. 8.15 [B] shows a more complex design that reduces 'wind-resistance' efficiently by directing particles above, below and to both sides of the design. This design is clearly aerodynamic and resembles not only the fronts of certain high-speed trains, but also the noses of some aircraft. Despite directing some particles below the train design, more are directed above the design, with the result that, in total, a down-force is generated, as required. (Note that the minor deficiencies

such as the small flat nose of [A] or the corners of the fixed block visible in [B] are the result of the system having to compromise streamlining in order to satisfy the 'size' constraints.)

8.6.2 Evolving the Bows of Boats

The second streamlined design problem presented to the generic evolutionary design system was to evolve the bows of boats. In exactly the same way as for the 'trains' problem, a fixed and inflexible primitive was defined, with the system evolving the position and shape of two primitives at the front. Again, a single enclosing volume was defined, with the desired force of 0.0 still required in the *x*-direction (i.e. no 'water' resistance) and a total desired force of 0.0 required in the *z*-direction (i.e. no forces pushing the design to one side). However, for the boats problem a much larger force was required in the *y*-direction (i.e. the 'water' should push boat designs upwards more than the 'air' pushed trains downwards). Finally, a desired maximum and minimum size for the bow of the boats was specified.

Again, symmetrical designs were specified, and the system was used with its parameters at their default settings to evolve a number of 'boat' designs, see fig. 8.15 [C] and [D]. Because of the more substantial 'upwards' force required, the designs did resemble boats (rather than upside-down trains). Designs were typically based upon two principles: either a conventional pointed front to force 'water' downwards and to either side of the design, or a groove down the centre of the front, to force 'water' downwards and inwards. Designs based upon the first principle strongly resemble conventional single-hull boats or ships, see fig. 8.15 [C]. Design based upon the second principle resemble twin-hull boats or catamarans, see fig. 8.15 [D]. The system favoured this latter type of boat design.

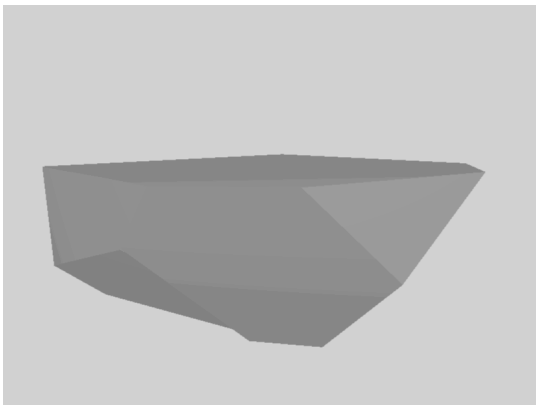
8.6.3 Evolving the Hulls of Boats

The third design problem using the 'particle flow simulator' was also to evolve boat designs, but this time the entire hulls of the boats were to be evolved, rather than simply the fronts. To achieve this, no fixed, inflexible block was defined - the system evolved the shape of all primitives in the designs. The function of the desired designs was specified identically to the

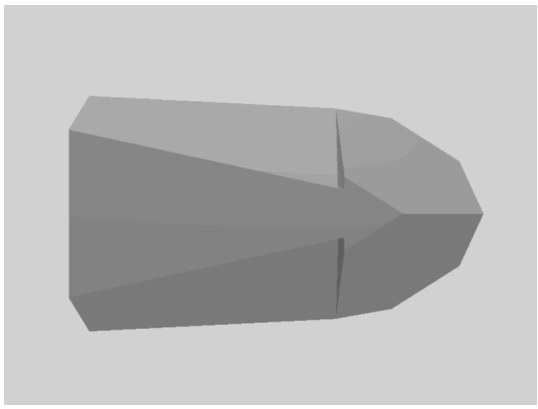
previous design problem. Symmetrical designs in $x = 0$ were specified, and four primitives (two before reflections) were used for each hull design.

This design problem was used to investigate whether the system was capable of evolving complete designs that were streamlined, rather than just the fronts of designs, given the highly simplified nature of the particle-flow simulator compared to true fluid dynamics. To help this investigation and guarantee that the primitives were positioned effectively, such that there was one primitive in front of another (instead of at the side or above), the *x*-positions of the primitives in the genotypes were initialised. All other genes (*y* and *z* positions, width, depth, angle1, and so on, for both primitives) were initialised randomly.

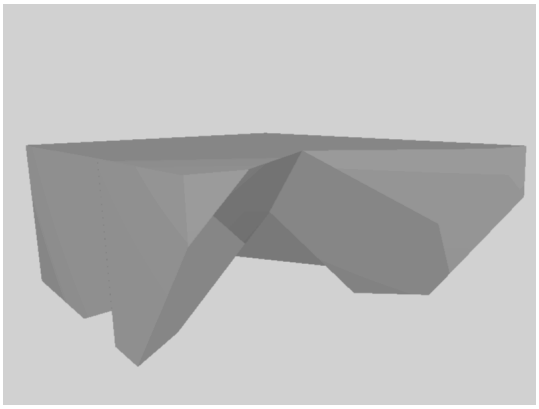
As can be seen from the images of the three evolved hulls shown in figure 8.16, the system was able to evolve a range of different and highly fit designs of boat hulls very successfully. Designs that used a variety of different methods to reduce the resistance to 'water' and increase the 'up-force' were evolved. For example, fig. 8.16 [A] and [B] shows a 'curved' boat hull with a shape very similar to a conventional small single-hull boat. Likewise, the hull design shown in fig. 8.16 [C] and [D] resembles the shape of a twin-hulled catamaran. Alternatively, the more unconventional design shown in fig. 16 [E] and [F] makes use of a single hull composed of two discrete elements. The smaller, higher portion of the hull at the front generates most of the upwards force required, whilst the larger portion at the rear minimises the resistance to 'water' by cleanly cutting the flow to each side.



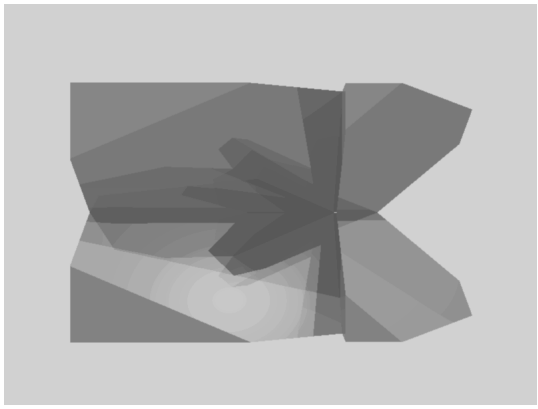
[A] Front view of evolved hull 1



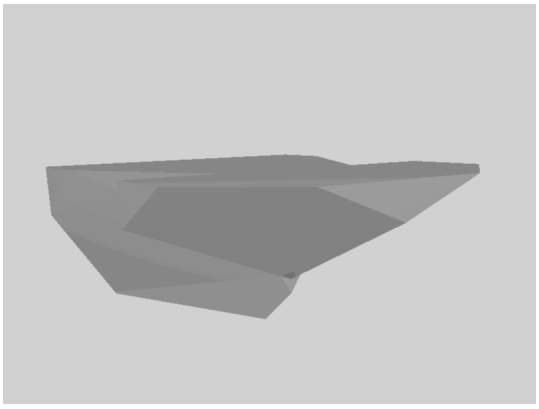
[B] Underside of evolved hull 1



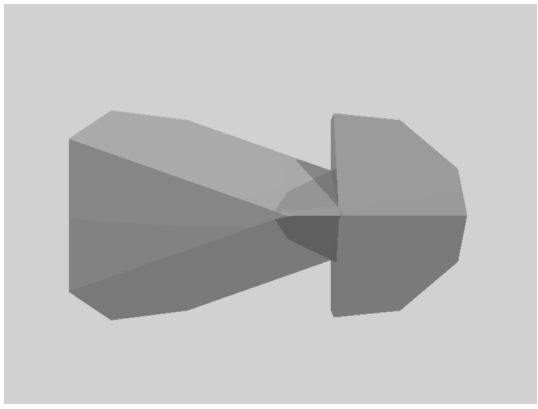
[C] Front view of evolved hull 2



[D] Underside of evolved hull 2



[E] Front view of evolved hull 3



[F] Underside of evolved hull 3

Fig 8.16 Three streamlined boat hulls evolved by the generic evolutionary design system.

8.6.4 Evolving a Saloon Car

The fourth type of 'streamlining' design problem presented to the system was to evolve the body of a saloon car. Although the 'size' constraint ensures that the extents of the design are of an appropriate size, cars require specific amounts of internal space to be available for the engine, storage and passengers. Hence, to reserve such space within all evolved designs, a skeletal 'block car' was defined using two fixed and inflexible primitives (using dimensions taken from a popular saloon car in production today).

As with the 'trains' design problem, the main function all evolved designs of saloon cars were required to perform was to have minimum 'air' resistence, i.e. to be aerodynamic. However, this time, in an attempt to encourage the evolution of 'rounded' sides for the designs (needed to reduce buffeting caused by side-winds), two enclosing volumes of space were defined, side by side, see fig. 8.17. These allowed the definition of two small equal and opposite desired side forces on the designs in the *z*-direction, in addition to a total desired force of 0.0 in the *x*-direction and desired down-forces in the *y*-direction.

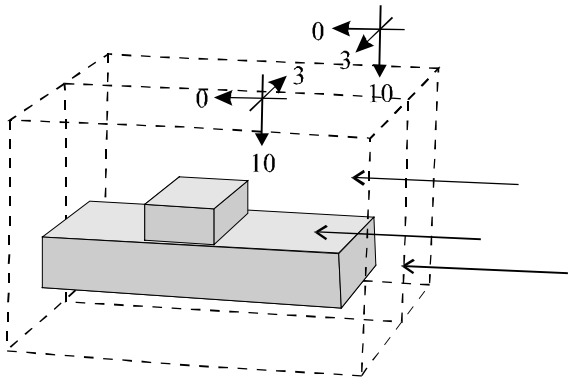


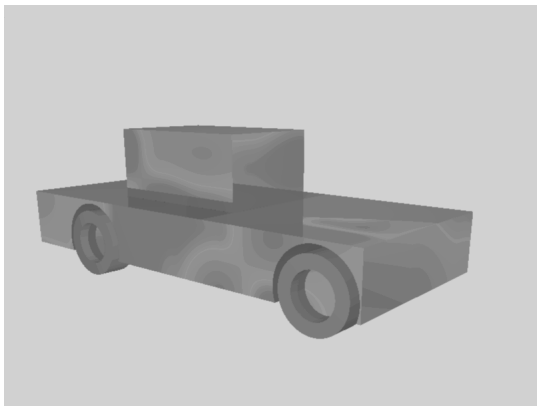
Fig. 8.17 Two fixed blocks form the skeleton, and two enclosing volumes are used to define the forces required on the saloon car designs.

Because it was required that primitive shapes should completely cover the fixed skeleton, leaving no gaps, it was necessary to initialise the position and size of all primitives to be used to define the body shape of the cars. If this was not done, because of the simple nature of the particle flow simulation, the system tended to position one large primitive at the front of the

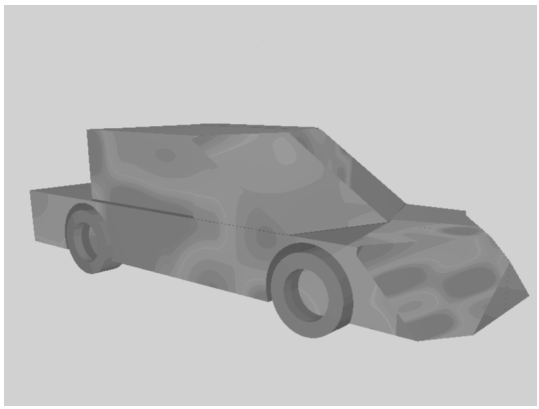
designs, optimise its shape to minimise the 'air' resistance, and then ignore everything behind it (producing designs looking like giant snow-ploughs). By seeding the initial population with designs with fixed skeletons evenly covered with fully evolveable primitives, the system is forced to consider and evolve the shape of the entire car, instead of just the front.

Since a standard requirement of most cars is symmetry (except perhaps for the exotic 'Indy' racing cars), all designs were defined to be symmetrical in $x = 0$. The system was then used, with equal importance values and all other parameters at their default values, to evolve a number of different designs of a saloon car, around the fixed skeleton. Figure 8.18 shows this skeleton and three evolved designs, rendered with *Polyray* using a 'steel' texture (with cloud reflections).

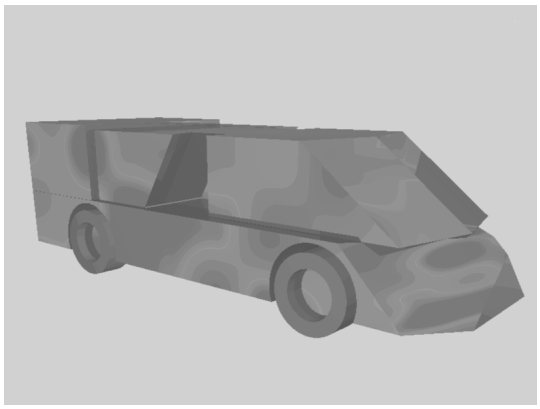
A range of very fit and acceptable designs were consistently evolved by the generic evolutionary design system. For example, fig. 8.18 [B] shows a ten-primitive car (using the system to evolve only the front half of this design) that has a curved, sloping 'windscreen' and 'bonnet', designed to reduce wind-resistance, and generate the required down-force and side-forces. The shape of this design resembles certain models of higher performance saloon cars currently available. Alternatively, fig. 8.18 [C] shows a twelve-primitive car that resembles a conventional transit van or 'people-carrier', with a combined sloping windscreen and bonnet, and two 'air-scoops' further back at the sides, evolved to provide some additional forces to the side of the design. Finally, fig. 8.18 [D] shows a fourteen-primitive car with a sloping windscreen and boot, and a more unconventionally shaped bonnet with an extra 'nose' on the top (composed of new primitives added by the system). This feature is used as a 'spoiler' in order to increase the down-force generated by the flow of 'air'.



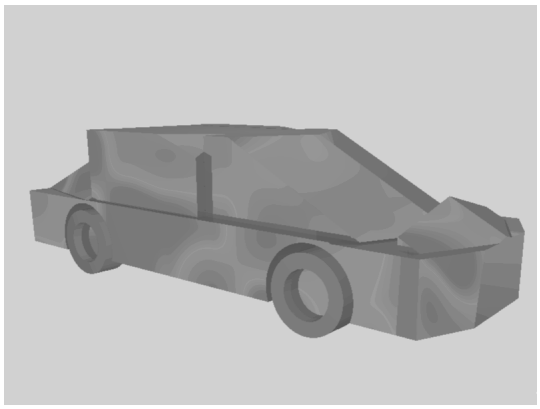
[A] Fixed, inflexible skeleton of saloon car



[B] Evolution of the front half of a saloon car



[C] Evolution of an entire saloon car



[D] Evolution of an entire saloon car

Fig. 8.18 Saloon cars evolved by the generic evolutionary design system (wheels added).

8.6.5 Evolving a Sports Car

The final design problem presented to the system was to evolve an aerodynamically shaped sports car. Again two fixed, inflexible primitives were used to define the skeleton of the car, but with a reduced height and increased width. The desired size of the sports car was defined to be the same height and width as the skeleton, but with a larger length. Since the 'size' module ignores inflexible primitives, this means that the system had to make up these dimensions using normal, evolveable primitives around the skeleton. Four enclosing volumes of space were used to define the forces required over each quarter of the design (i.e. over each wheel), see fig. 8.19. Forces of 0.0 were desired in the *x*-direction (zero wind resistance), small opposing forces on

each side of the design in the *z*-direction, and a specific down-force was desired over the wheels (more for the back wheels than the front).

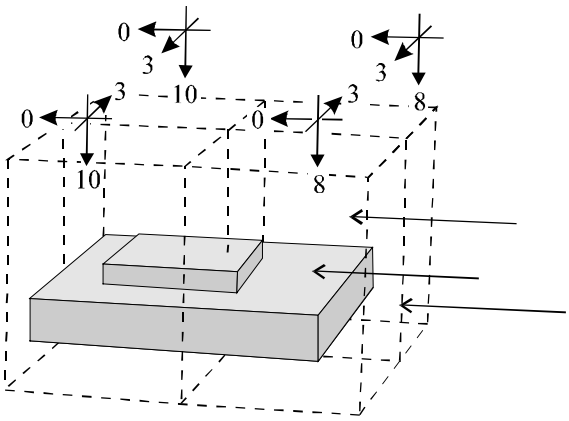
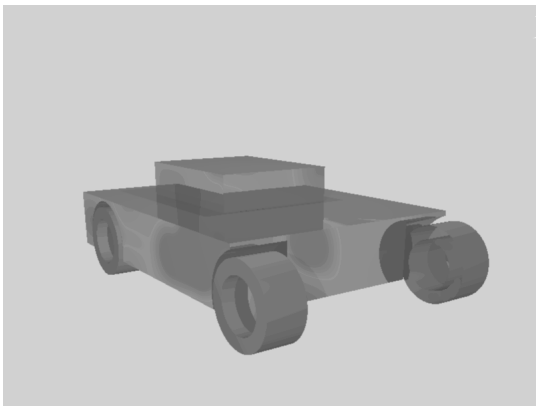
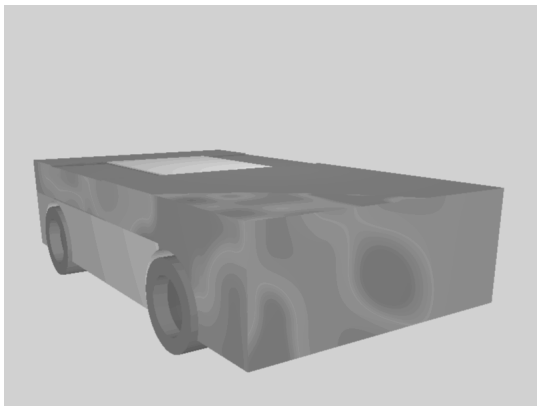


Fig. 8.19 Two fixed blocks form the skeleton, and four enclosing volumes are used to define the forces required on the sports car design.

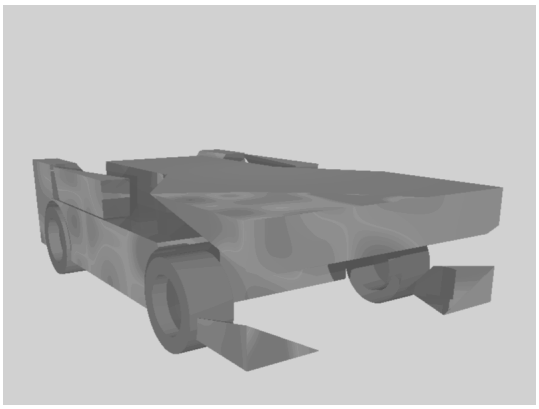
Primitives were positioned around the skeleton in the same way as for the previous example and the system (with default parameter values) was used to evolve a number of designs of sports cars. Figures 8.20 and 8.21 show the evolution of one such design, as it was optimised.



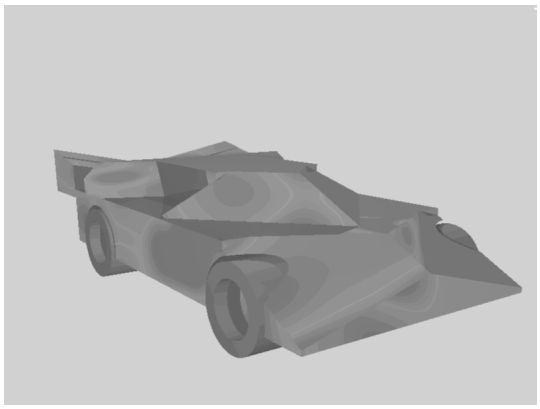
[A] Fixed, inflexible skeleton of sports car



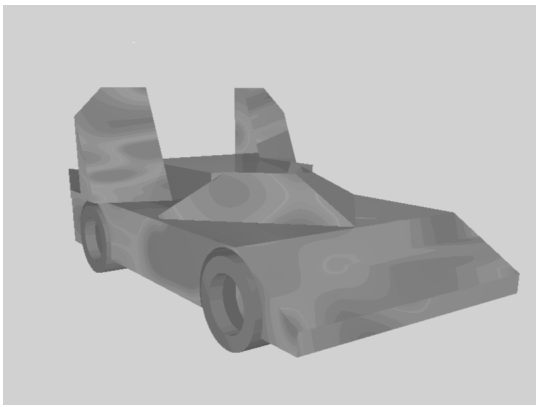
[B] Initialised primitives of sports car



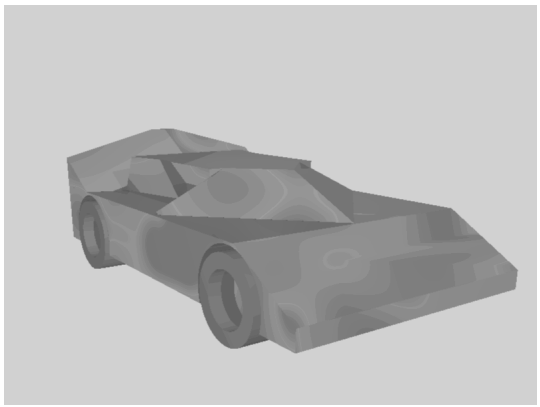
[C] A random initial design of sports car



[D] Best evolved sports car after 20 generations



[E] Best sports car after 100 generations



[F] Best sports car after re-evolving the rear

Fig. 8.20 The evolution of a sports car by the generic evolutionary design system (wheels added).

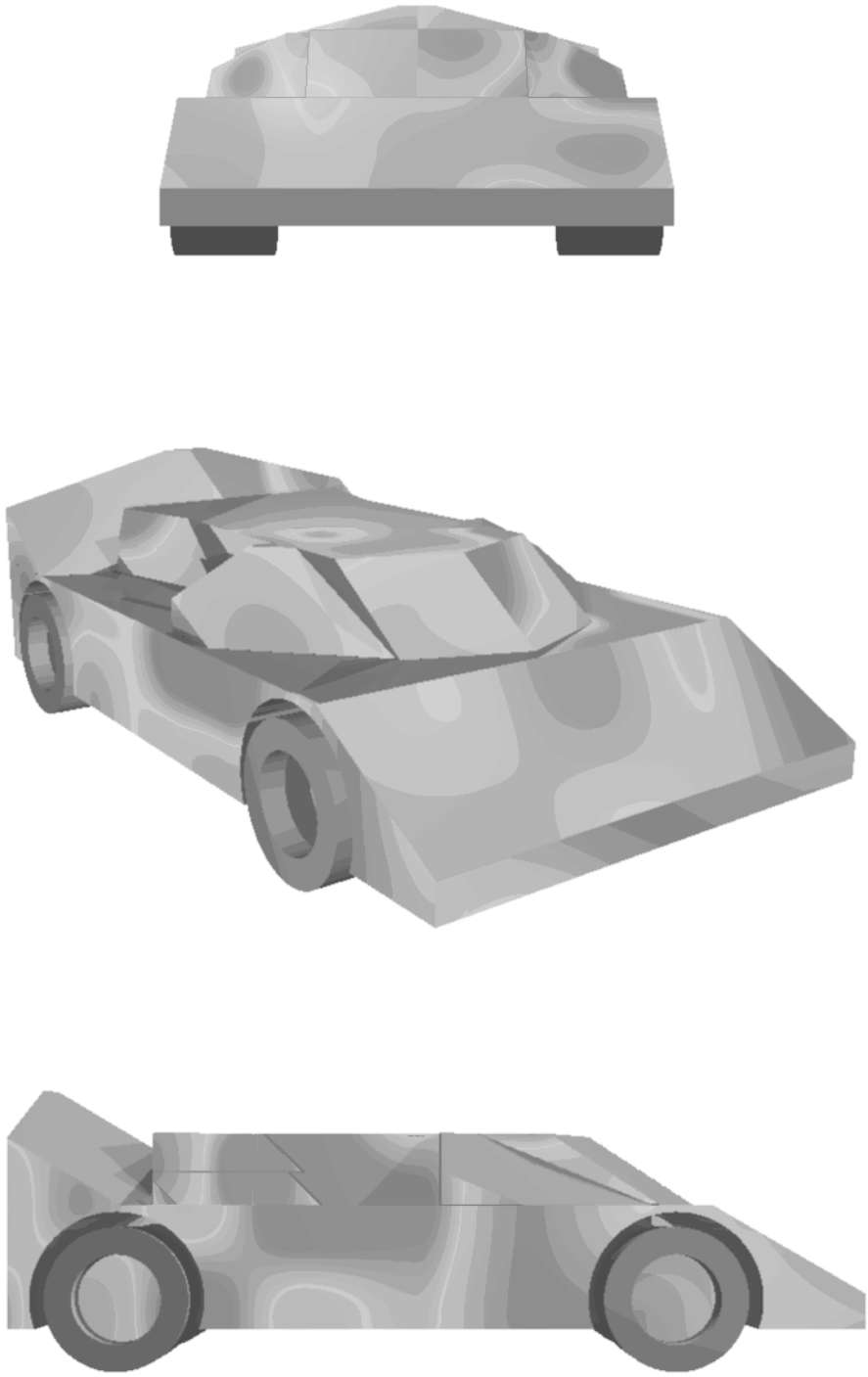


Fig. 8.21 The final evolved sports car design.

As fig. 8.20 shows, the system was initialised with a fixed, inflexible skeleton (fig. 8.20 [A]) and eight primitives were positioned around it (fig. 8.20 [B]). This resulted in typical initial designs (with random values for the intersecting planes of all evolveable primitives) after zero generations, that consisted of free-form 'blobs' positioned around a central skeleton, see fig. 8.20 [C]. Despite beginning from such randomised shapes (i.e. starting from scratch), after only twenty generations for this design, the recognisable shape of a sports car was evident, see fig. 8.20 [D]. Even at this early stage, the design has a streamlined sloping 'windscreen' and 'bonnet'. However, the rear of the car extended too far back (having had additional primitives added there) and was not yet guiding the flow of 'air' correctly.

After one hundred generations, the system had fine-tuned the slope of the bonnet, and had reduced the back of the car to the correct size. However, in order to generate the desired forces on the back wheels, the system had taken the unusual step of adding two large 'rabbit ears' to the back of the car, see fig. 8.20 [E]. These acted as twin 'spoilers' to alter the flow of 'air' in order to generate the additional down-force required. However, in the interest of aesthetics, it was felt that the system should be given a second opportunity to evolve the rear of the car - and hopefully find a more attractive alternative.

Consequently, the system was initialised with the evolved alleles that defined the front portion of the car design, and the rear was initialised randomly as before. Evolution was restarted, and the system quickly evolved a more conventional (and more attractive) back spoiler raised above the roof of the car, as well as two smaller side 'flaps', all designed to generate the desired forces on each part of the design, see fig. 8.20 [F]. Evolution was then permitted to continue for a further two hundred generations, during which the 'windscreen' was fine-tuned further by the system adding new primitives and increasing the curvature. (Another less apparent change was the splitting of the two side 'flaps' at the rear.) Figure 8.21 shows the final, highly fit design of the aerodynamic sports car, evolved by the generic evolutionary design system.

8.6.6 Summary of the Evolved Streamlined Designs

The evolved results for the five 'streamlined' design problems presented to the generic evolutionary design system have demonstrated the ability of the system to evolve a range of highly fit aerodynamic or hydrodynamic designs. Despite the simple nature of the 'particle-flow simulator' used, the system was consistently able to generate functionally and aesthetically acceptable designs to all problems.

These 'streamlined' design problems have illustrated the ability of the system to evolve designs in front of or around fixed, inflexible 'skeletons', to evolve complete designs or parts of designs, and to re-evolve selected parts of designs. The ability of the system to automatically optimise not only the position, size and shape of primitives in a design, but also the number of primitives was shown, by the system adding or removing primitives where needed. Moreover, the fact that the system was able to generate designs for this more representative and computationally expensive set of problems without any difficulty suggests that the techniques used may be scalable to permit the evolution of designs for even more realistic design problems in the future.

8.7 Summary

This chapter has described and illustrated the use of the generic evolutionary design system to evolve solutions for five different types of design problem. In total, fifteen different design tasks were presented to the system: tables, steps, heat sinks, right-angle prisms, roof prisms, derotating prisms, rhomboid prisms, penta prisms, abbe prisms, porro prisms, train fronts, boat bows, boat hulls, saloon cars and sports cars. Each of these tasks involved the evolution of a design with an entirely different shape, in order to allow that design to perform the desired function. Despite some of these problems being deceptive for the GA, this generic system was able to evolve not only fit, but acceptable designs (as judged by humans) for all fifteen problems.

Moreover, designs evolved by the system were based upon sound principles; for example the system independently 'discovered' that a table design could be made stable by lowering the centre of mass or using a large base. The system made use of complex concepts such as refraction and total internal reflection to correctly guide rays of light through designs. The system 'invented' methods without human guidance to streamline designs and to generate down-force (e.g. curved and sloping surfaces, 'spoilers'). Using no world knowledge, the system created designs of heat sinks, prisms, trains, boats and cars that closely resembled designs created by human designers, in addition to creating a whole range of highly original and unconventional alternatives.

These fifteen design problems all demonstrated significant aspects of the system. The use of symmetry, importance, and the advantages of the chosen multiobjective ranking method were shown. The ability of the system to accurately evolve alleles defining the position and size of primitives, and the orientation of their intersecting planes, and the ability of the system to vary the number of primitives was demonstrated and seen to be valuable. The various methods of seeding the initial population in the system to allow evolution from scratch, evolution of designs using smaller components, evolution around fixed, inflexible skeletons and re-evolution of selected portions of designs was shown. Finally, comparisons were made between the performance of early prototypes of the system and the complete generic evolutionary design system. Designs were evolved faster with the complete system and were just as fit or fitter than those evolved by the prototypes, showing that the extra techniques used in the complete system improve the performance, in addition to increasing the ease of use, without reducing the generality of the system.