9

Further Work and Conclusions

9.1 Introduction

This thesis has described the investigation and creation of a computer program capable of the generic evolutionary design of solid objects. A wide and far-reaching review of existing literature related to this topic was performed, with the conclusion that no such computer system currently exists. All aspects of the system were then described in depth: the new solid object representation used to define designs, the genetic coding and novel genetic operators used to allow the system to modify designs, the variety of new techniques used within the genetic algorithm used to allow acceptable designs to be generated, and the library of new analysis software used to evaluate designs. The abilities of this generic evolutionary design system were then demonstrated by using the system to evolve solutions to fifteen different design problems.

In the introduction to this thesis, a list of the intended capabilities of the system that was created as part of this research was given. This chapter summarises the actual capabilities of the system, as observed through experimentation, in terms of this list. Following this, a discussion of how the capabilities of the system could be extended and expanded is given, with suggestions for future research following on from this work. Finally, conclusions are drawn and the new and significant advances made by this research project are summarised.

9.2 Summary of System Capabilities

In Chapter One, it was stated that the generic evolutionary design system should have five principal capabilities. Explicitly, it was desired that the system should have the capability to:

1. **Evolve solid object designs from purely random beginnings, or from a combination of random and user-specified initial values.**

As was shown in the previous chapter, the system is indeed capable of evolving highly fit designs (as judged by the evaluation software) from initial populations of entirely random individuals. Beginning with such free-form 'blobs' permitted the system to create new and sometimes unconventional designs from scratch. These designs used a range of different design concepts, which were all 'discovered' independently by the system. Alternatively, by defining some values and initialising the remaining with random values, the system was able to evolve designs around fixed, inflexible skeletons, evolve new designs from previously evolved components, or to re-evolve selected portions of designs.

2. **Evolve a range of different types of solid object design.**

The generic nature of the new solid-object representation developed for this work, and the robust nature of the genetic algorithm have both helped to ensure that the evolutionary design system is generic. These general design capabilities have been demonstrated by the successful evolution of designs for fifteen different solid-object design problems by the system. Of these problems, the optical prism tasks were found to be the most 'difficult' for the system. However, by increasing the accuracy of the problem specification in the evaluation software, the system was able to evolve good solutions even for these 'hard' or deceptive problems.

3. **Allow the simple specification of a range of different solid object design tasks.**

The use of modular evaluation software permits new design applications to be very quickly specified. In order to define a new design problem, the user simply has to pick which modules should be used in combination, and specify some desired parameter values. By maintaining a library of reusable modules, this means that a new module of evaluation software need only ever be created once. In this way, the amount of additional software development required for new design applications is minimised. Moreover, the performance of the system is not dependent on the fine-tuning of system parameter values. Almost all of the designs were evolved using the system with its default settings, i.e. mutation rates, population sizes, life spans, percentage of parents selected, and so on, were not changed. In addition, the new SWGR multiobjective ranking method used within the system entirely removes the difficult problem of weighting separate criteria to allow them to be treated equally (and to allow explicit relative importance values).

4. Successfully evolve designs guided only by evaluation software during the evolution process.

All designs evolved by the system shown in the previous chapter were judged only by the selected modules of evaluation software. Human interaction is limited to selecting whether evolution should be prematurely terminated or whether the system should be used to re-evolve selected portions of designs. This removes the potential restriction of 'conventional wisdom' from preventing the system from evolving unusual and unexpected solutions to design problems. Indeed, as was seen with the 'tables' problem, the system is capable of creating highly unusual and distinctive designs that a human designer would be unlikely to originate.

5. Evolve useful and innovative designs.

As stated in Chapter One, the long-term goal of this research is to produce a design system capable of evolving truly useful solutions to real-world design problems. These designs could either be used directly, or could be used by human designers for inspiration. For this project, it was intended that the system should have the ability to successfully evolve acceptable and potentially innovative solutions to model design tasks, created to test the major faculties of the system. As was illustrated by the designs evolved for the fifteen design tasks, the system is indeed capable of evolving such solutions to a range of different problems.

9.3 Further Work

This thesis has described pioneering research in the evolutionary design of general solid objects from scratch. Because this was a new area of research, the main aim of this project was to demonstrate the feasibility of using a computer to evolve solid-object designs. Having now shown that this is possible, some potential future avenues of research following on from this project can be discussed.

9.2.1 Additions to the Solid Object Representation

The new spatial-partitioning representation created as part of this work, allows the definition of a wide range of solid object designs using as few parameters as possible. Designs with flat surfaces (e.g. heat sinks, prisms) can be defined with a high degree of precision, but this representation is unable to accurately represent curved surfaces. Although the 'streamlined' set of design problems demonstrated that the system is capable of producing designs with good approximations of curved surfaces, it seems likely that the designs could be substantially improved if curved surfaces could be directly defined by the representation.

As was described in Chapter Four, most existing surface representations that allow curves to be defined (e.g. Hermite, Fourier, Parametric, or Bézier surfaces) typically require numerous parameters, and would be difficult to use in combination with the existing solid-object representation. However, work recently published demonstrates the use of 'reparameterised superquadrics' with a GA to allow the evolution of solids with curved surfaces (guided by a human evaluator). This representation was shown to be evolveable by a GA and to be capable of the accurate definition of a wide range of solids with curved surfaces, using relatively few parameters (Husbands, Jermy, McIlhagga, & Ives, 1996).

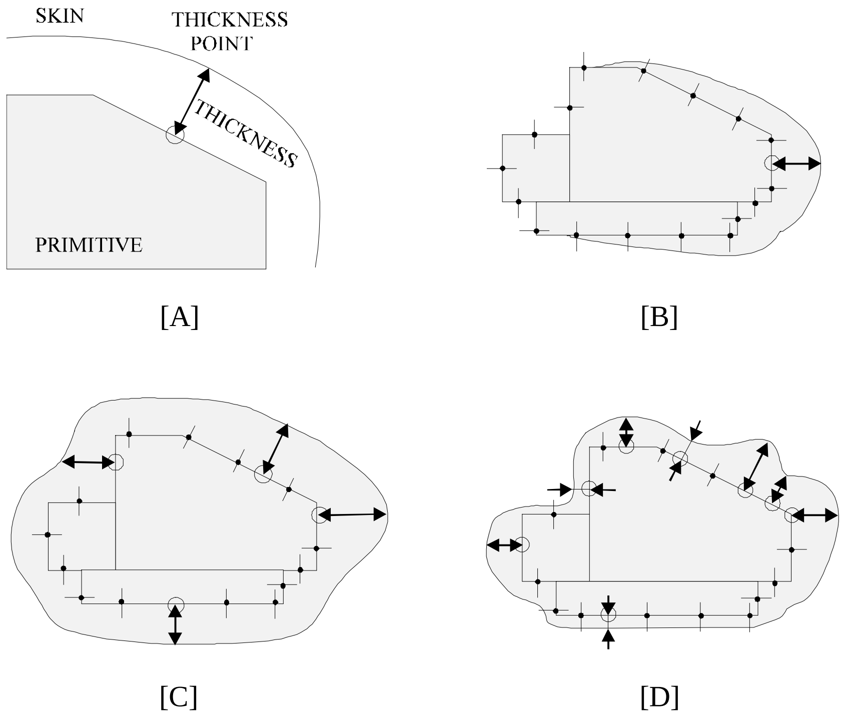

Fig. 9.1 Using a 'skin' stretched over primitives to define curved surfaces.

Alternatively, a suitable alternative could be to use the idea of a 'skin' that would be 'stretched' over a design defined using the current spatial-partitioning representation. This 'skin' could consist of a Parametric surface extrapolated from a variable number of evolveable control points defined on any part of the underlying design. These points would define the distance between the design and the surrounding surface, see fig. 9.1 [A]. Any part of a 'skin' with no 'thickness points' would have, by default, an extrapolated grid of virtual thickness points specifying diminishing thicknesses, thus potentially defining parts of the surface to be tight against the primitives of the spatial-partitioning representation, see fig. 9.1 [B]. (The thickness at these virtual points would decrease by a pre-defined rate of descent between thickness points, until they reached zero.) The greater the thickness, the further from the underlying primitives the 'skin' becomes, allowing true curves, their shape based on the internal primitives, to be defined, see fig. 9.1 [C]. Likewise, the more thickness points, the more undulations in the surface that could be defined, see fig. 9.1 [D]. In this way, complex curves could be defined, using only a very small number additional parameters (and corresponding genes in the genotype).

Other additions to the solid object representation could also be made. For instance, designs made from non-homogeneous materials could be simply defined by allowing each primitive of the design to have a 'material' identifier. This would then allow the system to evolve not only the shape, but also the materials used within each design. (Alternatively, to keep the search-space continuous, a 'density' variable could be used to indirectly define the material.) Moreover, the surface appearance of designs (e.g. colours and textures) could also be added to the representation in the same way, and be evolved by the system. (Indeed, Todd and Latham added evolveable textures to their evolved art, resulting in surfaces with intricate patterns and colours; Todd & Latham, 1992).

Another addition to the representation that could be made is *movement*. Most designs will only perform their function because of the internal motion of components (i.e. stretching, bending, rotation or translation). It is possible that separate primitives or groups of primitives could be allocated certain behaviours in the representation (e.g. rotate clockwise, or expand by 10%) which would then be modified by the system to allow the motion as well as the shape of components in designs to be evolved. This is similar to the approach used by Sims to evolve the swimming and walking motions of his evolved 'creatures' (Sims 1994a,b) and the approach used by Chakrabarti to generate conceptual designs of simple mechanical devices (Chakrabarti, 1995).

However, strictly speaking, all motion is created because of external forces (or energies) exerted on shapes made from specific materials. For example, a clockwork mechanism generates many rotational movements because of an initial rotational force applied to a coiled piece of metal (the spring). An electronic circuit generates a wide variety of output (the flow of electrons, photons, or the emission of various types of electromagnetic radiation) because of a flow of electrons applied to it, which is then channelled through a complex shape composed of many different types of material (to offer conductivity, resistance, capacitance, and so on). This means that the motion of designs (or the motion of anything within designs, e.g. electrons within circuits, light within prisms) should be generated as part of the evaluation process, and should not be an element of the representation. In other words, primitives in designs should not have fundamental behaviours associated with them - they represent inanimate objects. Any forces or energies that are required to make the design function should be *applied* at the time of the evaluation of that design. If it is desired that the nature of the forces or energies is alterable by evolution, then this can be easily achieved using a separate chromosome (or even a separate coevolving 'species'; Husbands 1994), independent of the representation.

The only exception to this are the types of substances that continuously exert forces or emit radiation because of their composition (e.g. magnetised materials, radioactive materials, or electric cells). These 'special' substances could have the magnitude of their 'behaviours' legitimately specified as part of the representation and hence evolved by the system, although this is merely an extension to the 'material' identifier discussed earlier.

9.2.2 Artificial Embryology

A simple form of artificial embryology was used in the generic evolutionary design system to allow illegal genotypes (those that defined designs with overlapping primitives) to be mapped to legal phenotypes. In addition, simple genotypes (defining partial designs) were mapped onto symmetrical phenotypes. As was described in previous chapters, this mapping process was used within the system to allow evolution to be unconstrained and to allow simple genotypes to define more complex phenotypes, and hence reduce the number of genes that needed to be considered by the system.

An extension to this mapping process would create a more realistic artificial embryology. Currently, symmetry is a user-defined external option, specified for all phenotypes being evolved by the system. By the addition of a 'symmetry' control gene to genotypes, to be used by the mapping process, this characteristic could be made evolveable by the system, allowing the degree of symmetry in designs to be automatically determined. Moreover, by using multiple symmetry control genes in a genotype to define which groups of primitives should be symmetrical in which planes, different portions of phenotypes could be made symmetrical independently of each other.

Nature commonly uses embryology in order to allow the definition of multiple duplicated parts of creatures, e.g. body segments of a caterpillar, limbs, eyes (Dawkins, 1976). This permits mutation to duplicate entire body parts or organs of creatures, removing the need to keep re-evolving the same thing (a feature used extensively in Todd's 'Form Grow' embryology, resulting in shapes with multiple 'ribs' or 'tentacles'; Todd & Latham, 1992). In the same way, 'duplicate' control genes could be added to the genotypes of designs in the system. These control genes would define which primitive or groups of primitives should be duplicated, how many times they should be duplicated, and where the duplicated primitive(s) should be placed during the mapping stage from genotypes to phenotypes. Just as in nature, this would permit complex structures to be duplicated during evolution and would probably be beneficial for many types of designs, e.g. the multiple flat surfaces of the 'steps' problem, or multiple upright 'slats' for the 'heat sink' problem.

9.2.3 Overspecification, Underspecification and Dominance

As was described in Chapter Five, Hierarchical Crossover is capable of generating meaningful offspring from two parent chromosomes with variable numbers of bits in genes, variable numbers of genes, and variable numbers of groups of genes. The generic evolutionary design system currently only uses mutation to vary the number of groups of genes in a chromosome (i.e. vary the number of primitives in a design). By adding another mutation operator to vary the number of genes per group of genes, deliberate overspecification, underspecification, and dominance would occur within the system. In other words, the number of alleles for a gene could be zero (underspecification), one, or more than one (overspecification).

The system will automatically use a default value if a gene is underspecified, or will use the first (dominant) duplicate allele found for an overspecified gene. Hierarchical Crossover preserves the order of alleles within chromosomes during crossover, so a child will inherit the order (and hence the dominance) of duplicated alleles (Bentley & Wakefield, 1996d). If another random mutation operator was added to change the ordering of alleles within chromosomes, the GA could not only evolve the values of genes and number of genes, but also which duplicate allele should be dominant for a gene.

This enhanced level of control over chromosomes is similar (but not as extensive) as that found in natural evolution (Paton, 1994). Researchers using similar techniques have discovered that such extensions can increase the diversity of individuals in populations and reduce the propensity towards premature convergence (Deb & Goldberg 1991, Smith & Goldberg 1992b).

A modification to the data structures of the system would allow the number of bits per gene to be variable. Theoretically, this could permit the GA to evolve the precision of genes in addition to their values. If single bits were also permitted to be underspecified, overspecified and dominant, the diversity of individuals could be increased further. To the author's knowledge, no GA with variable numbers of bits per genes has been developed as yet.

9.2.4 Multiple Design Solutions

The GA used in the generic evolutionary design system converges to a single design solution every time it is run. Although the range of solutions that the system will converge to is a small sub-set (determined by the 'importance' values) of the range of Pareto-Optimal solutions, because the system is seeded randomly, for most problems the system will evolve a different solution every time. This means that the only way to generate a number of alternative solutions to any problem is to repeatedly run the system.

However, there are certain types of GA that permit the concurrent co-evolution of multiple different solutions. These GAs use 'niching' to segregate the population into separate 'species' (Horn et. al. 1994). These separate and often non-interbreeding sub-populations of individuals are then evolved by the GA, each independently converging on a solution. This results in multiple solutions to a single problem being evolved every time the GA is run.

The addition of some form of niching to the GA used in the system would not be difficult, and would permit a number of alternative designs to be evolved in each run - a potentially useful feature. If co-operation was permitted between the solutions in separate species (e.g. a distributed GA; Whitley and Starkweather 1990), more than one type of design could be evolved at once (e.g. tables and chairs, or interlocking gears). This could also increase the performance of the GA and quality of the solutions (Levine, 1994). Moreover, the use of parallel processors to run such a GA (i.e. a parallel GA; Levine 1994) would enable the increased computation to be performed in an acceptable time.

9.2.5 New Applications

Having demonstrated that a number of different types of design problem can be adequately specified using modular evaluation software, and that the system is capable of evolving good solutions to these problems, the obvious next step would be to apply the system to some real-world design problems. It seems likely that the most suitable types of problem for the system are those that do not have tightly constrained solutions, i.e. there should be a wide variety of alternatives available to the system. Moreover, acceptable solutions must be evolveable by the system, i.e. the system must be able to begin with a very poor solution and slowly optimise it in a continuous series of steps, to produce the final solution.

Both these requirements are evident from using the system to evolve the different 'model' design problems. For example, the 'optical prisms' problems were the most tightly constrained of all the design tasks, with many types having only a single acceptable solution. Not only that, but some problems (e.g. the rhomboid prism) were very difficult to evolve from random beginnings, since every part of the design relied on every other part being correct before the design could perform its function at all. Hence, for these problems, the system could only generate a single type of solution, and additional problem-specific knowledge was required in order to help guide evolution. In comparison, the 'steps' problem was less constrained, but still had much of its shape precisely determined (e.g. the three desired steps at specific positions, having specific sizes), and had a desired function that was difficult to perform (i.e. support a very heavy mass on each step, whilst remaining stable). This reduced the number of alternative designs that the system could evolve. However, the 'tables', 'heat sinks' and 'streamlined' problems all had a large number of possible good solutions, and all could be evolved from random beginnings. Because of this, the system was able to evolve a wide variety of unusual and sometimes unconventional designs for these problems.

Consequently, examples of perhaps the most suitable type of design applications are true aerodynamic or hydrodynamic shapes (using computational fluid dynamics in evaluation) such as racing-cars, submarines, propeller blades, and aircraft. In addition, structural designs, such as bridges, transmission towers, electricity pylons. Alternatively, floorplanning design problems for factories, oil-rigs or shops (using primitives to define either individual rooms or the walls between rooms) could be attempted.

All of these applications are common in real life, all have multiple solutions, and it is highly desirable to find good solutions for them. Indeed, GAs are commonly used to optimise existing solutions for most of the problems listed (as was described in Chapter Two).

An alternative type of suitable application could be the evolution of designs based on their aesthetics. A simple modification to the system would allow the user to be presented with a number of individual designs from each population, and give scores to each of them based on how attractive the user finds them. This is the approach used by Dawkins, and Todd and Latham to allow the evolution of shapes that resemble creatures or attractive shapes (Dawkins 1986, Todd & Latham 1992). The same method was also used by Furuta to allow the evolution of aesthetic bridges (Furuta et. al., 1995). However, throughout the development of the generic evolutionary design system, natural selection was favoured over the artificial selection of such systems, i.e. the aim of this work has always been to create a program capable of evolving new designs *without* human interaction. It is currently unknown whether it is possible or not to formulate a number of evaluation modules capable of adequately analysing how attractive a design is. Perhaps a potential solution could be to train an artificial neural network to 'recognise' aesthetically pleasing designs, and use that to determine the corresponding fitness scores.

Finally, for some new applications it may be desirable to interface to some existing analysis packages and use them as evaluation modules. Indeed, the solid object representation used for this work can be converted to the standard CSG representation very simply. However, some analysis packages are incapable of analysing objects of any shape, so at least for the early stages of evolution, no fitness scores could be calculated for the designs. By placing constraints on which shapes are allowable in the representation, this problem could possibly be overcome, but it seems likely that such restrictions would also limit the ability of evolution to evolve creative solutions. An alternative could be to perform evolution in two stages: first evolve designs from scratch using a simple 'guiding' evaluation module, then fine-tune these partially evolved designs (that are now acceptable for the analysis software) by evolving them using the analysis software. However, it seems probable that the best results would be obtained by re-developing such analysis software, to allow it to evaluate designs of any shape.

9.2.6 Enhancements to the User-Interface

The current user-interface to the system permits the overall control of the system and allows the best current design to be displayed during evolution. However, all input commands and parameters must be defined in one initialisation file. A substantially more user-friendly alternative would be the use of a 'point-and-click' interface to allow options and commands to be specified by simply clicking on the appropriate control in a window.

Perhaps the biggest improvement that could be made to the user-interface would be a graphical display used to define the desired input parameters for evaluation modules. Currently, to define for example, the outer and inner 'size' extents, the user must type twelve appropriate parameter values into the initialisation file. A far easier method would be to allow the user to draw the two outer and inner cubes using a mouse, and use the corresponding dimensions for the desired input parameters of the evaluation module, see fig. 9.2.

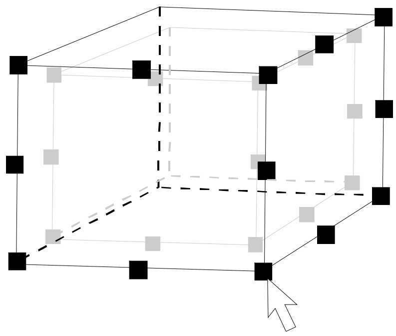

Fig. 9.2 Using a mouse to draw desired inner and outer extents for a design.

Likewise, if a flat upper surface was required, the user could delineate its desired size and position on the screen. If the user wished to initialise the size or position of any primitives, they could simply use the mouse and draw their positions or sizes. Finally, this method would allow a user to easily select portions of an evolved design to be re-evolved by selecting the acceptable primitives or groups of primitives with the mouse and defining them to be fixed or inflexible.

9.4 Conclusions

This thesis has presented a new way of using computers in design. It has shown that it is possible, feasible and useful to produce a generic evolutionary design system capable of successfully creating new and original designs of a range of solid objects.

The following specific conclusions can be drawn from the research described in this thesis:

- The genetic algorithm is a highly suitable type of search algorithm to form the core of the system.

- The novel spatial-partitioning solid-object representation: 'Clipped Stretched Cubes' allows a wide range of different solid object designs to be defined and manipulated using very few definition parameters.

- The novel concept of a 'semantic hierarchy' (i.e. tree of meaning) of a chromosome in conjunction with the new 'hierarchical crossover' operator ensures that meaningful offspring are always produced from two parent chromosomes of different lengths.

- The 'split' and 'delete' mutation operators allow the generic evolutionary design system to vary the number of primitives in phenotypes, and, when used with hierarchical crossover, to optimise this number in designs.

- The novel multiobjective ranking method SWGR, which makes use of the new concepts of range-independence and importance, has been shown to generate consistently a user-defined subset of acceptable solutions in the Pareto-Optimal range, without laborious fine-tuning of weights.

- The use of an explicit genotype to phenotype 'mapping stage' in the GA allows a small number of unconstrained genes to define more complex, legal phenotypes.

- The use of steady-state-replacement with preferential selection in the GA ensures that very fit solutions will not be lost from populations and helps to reduce the number of evaluations needed before the system converges to an acceptable solution.

- The novel technique of using a maximum 'life span' in the GA, prevents 'lucky' individuals from becoming immortal and corrupting evolution.

- A range of different solid object design tasks can be adequately and quickly defined by the use of different combinations of reusable evaluation modules, picked by a user from a library of such modular software.

This research has proved the concept of a generic evolutionary design system. This was demonstrated by evolving consistently acceptable designs for fifteen different design problems. Designs evolved by the system were based on sound conceptual ideas, 'discovered' independently by the system. The shapes of designs were optimised in order to ensure that they performed the desired function accurately. The system evolved a range of conventional and unconventional designs for all problems presented to it. The less constrained the design problems were, the wider the variety of alternative, and sometimes highly unusual design solutions the system evolved. The use of 'special features' such as symmetry, fixed and inflexible primitives in designs, increased the ability of the system to evolve various types of design solution.

In conclusion, evolutionary design has been performed in nature for millennia. This research has made the first steps towards harnessing the power of natural evolutionary design, by demonstrating that it is possible to use a genetic algorithm to evolve designs from scratch, such that they are optimised to perform a desired function, without any human intervention.

References

Adachi, N. & Kazuhiro, M. (1992) Ecological Dynamics of Strategic Species in Game World. *FUJITSU Scientific Technical Journal v28:4*. 543-558.

Adeli, H. (ed), (1994). *Advances in Design Optimization.* Chapman & Hall Pub.

Adeli, H. & Cheng, N., (1994a). Integrated Genetic Algorithm for Optimization of Space Structures. *ASCE Journal of Aerospace Engineering* v6:4, 315-328.

Adeli, H. & Cheng, N., (1994b). Augmented Lagrangian Genetic Algorithm for Structural Optimization. *ASCE Journal of Aerospace Engineering* v7:1, 104-118.

Adeli, H. & Cheng, N., (1994c). Concurrent Genetic Algorithms for Optimization of Large Structures. *ASCE Journal of Aerospace Engineering* v7:3, 276-296.

Albin, P. (1992). Approximations of cooperative equilibria in multi-person Prisoner's dilemma played by cellular automata. *Journal of Mathematical and Social Sciences* v24:2/3, 293-319.

Anwei, L. et al. (1989). HYBRID -- A Solid and Surface Modeling System Based on Database. In *CAD and Computer Graphics '89*, Beijing, (pp.222-227).

Axelrod, R. (1987). The Evolution of Strategies in the Iterated Prisoner's Dilemma. In *Genetic Algorithms and Simulated Annealing*, (London, Pitman) Davis, L. (ed.), 32-41.

Bäck, T. (1993). Optimal Mutation Rates in Genetic Search. In *Proceedings of the fifth internation conference on Genetic Algorithms*, Urbana-Champaign, USA. Morgan Kaufmann Pub. (pp. 2-8)

Beasley, J. E. & Chu, P. C. (1994). A Genetic Algorithm for the Set Covering Problem. In *Applied Decision Technologies*, London (late insert).

Bedau, M. A., Ronneburg, F. & Zwick, M. (1992). Dynamics of Diversity in an Evolving Population. In *Parallel Problem Solving from Nature 2*, Brussels, Belgium. Elsevier Science Pub. (pp. 95-104)

Bentley, P. J. and Thorn, M. T. (1994). An Affordable Virtual Reality Training Aid for Anaesthetists. In Proceedings of the International State of the Art Workshop: *Application and Exploitation of VR Systems in Surgery and Medicine*, 23 April 1994, Leeds.

Bentley, P. J. & Wakefield, J. P. (1995a). The Evolution of Solid Object Designs using Genetic Algorithms. In *Applied Decision Technologies* (ADT '95), April 1995, London, (pp. 391-400). Further published as:
Bentley, P. J. & Wakefield, J. P. (1996a). The Evolution of Solid Object Designs using Genetic Algorithms. In Rayward-Smith, V. (ed) *Modern Heuristic Search Methods*. Ch. 12, John Wiley & Sons Inc., 199-215.

Bentley, P. J. & Wakefield, J. P. (1995b). The Table: An Illustration of Evolutionary Design using Genetic Algorithms. In the first IEE/IEEE Int. Conf. on *Genetic Algorithms in Engineering Systems: Innovations and Applications* (GALESIA '95), Sept. 1995, Sheffield, (pp. 412-418).

Bentley, P. J. & Wakefield, J. P. (1996b). Generic Representation of Solid Geometry for Genetic Search. *Microcomputers in Civil Engineering* 11:3, Blackwell Publishers, 153-161.

Bentley, P. J. & Wakefield, J. P. (1996c). Overview of a Generic Evolutionary Design System. In *Proceedings of the 2nd On-line Workshop on Evolutionary Computation* (WEC2), Nagoya University, Japan, (pp. 53-56).

Bentley, P. J. & Wakefield, J. P. (1996d). Hierarchical Crossover in Genetic Algorithms. In *Proceedings of the 1st On-line Workshop on Soft Computing* (WSC1), Nagoya University, Japan.

Bentley, P. J. & Wakefield, J. P. (1996e). Conceptual Evolutionary Design by Genetic Algorithms. *Engineering Design and Automation Journal v2:3,* John Wiley & Sons, Inc.

Bentley, P. J. & Wakefield, J. P. (1996f). An Analysis of Multiobjective Optimisation within Genetic Algorithms. Submitted to *Evolutionary Computation*.

Bouchard, E.E., Kidwell, G. H., and Rogan, J. E. (1988). The Application of Artificial Intelligence Technology to Aeronautical System Design. In *AIAA-88-4426, AIAA, AHS and ASEE, Aircraft Design, Systems and Operations Meeting*, Atlanta, Georgia, (pp.7-9).

Boyd, I. D., (1994). Constrained Gas Network Pipe Sizing with Genetic Algorithms. (Submitted to) *Parallel Problem Solving From Nature.*

Boxer, S. (1996). Who Needs Nature? Article in *The Daily Telegraph 16/4/96,* UK, 8-9.

Bramlette, M. F. & Bouchard, E. E. (1991). Genetic Al;gorithms in Parametric Design of Aircraft.. Ch. 10 in *The Handbook of Genetic Algorithms*. Van Nostrand Reinhold, New York, 109-123.

Brown, E. B., (1966). *Modern Optics* (Second Edition). Reinhold Pub. Corp., Chapman and Hall Ltd., London.

Cai, J. & Thierauf, G. (1996). Structural Optimization of a Steel Transmission Tower by using Parallel Evolution Strategy. In Proceedings of *Adaptive Computing in Engineering Design and Control - '96*, (pp. 18-25).

Chakrabarti, A., (1995). FuncSION: A Software for Synthesis of Mechanical Designs. *Engineering Computing Newsletter 59*, Nov. 1995, p. 2.

Chambers, B. et. al. (1995). Application of genetic algorithms to the optimisation of adtaptive antenna arrays and radar absorbers. In *Genetic Algorithms in Engineering Systems: Innovations and Applications* (GALESIA '95), Sheffield, (pp. 94-99).

Chuan Jun Su, Mayer, R. J., Browne, D.C. (1991). Generalised CSG: A Solid Modeling Basis for High Productivity CAD Systems. In *Autofact '91*, (pp.17/35-17/43).

Cliff, C.; Husbands, P.; Meyer, J.; Wilson, S.W. (eds) (1994). *From Animals to Animats 3*. Proceedings of the third international conference on simulation of adaptive behaviour, MIT Press.

Culley, S. J. and Wallace, A. P., (1994). Optimum Design of Assemblies with Standard Components. Proceedings of *Adaptive Computing in Engineering Design and Control - '94*, (pp.163-168).

Dasgupta, D. and McGregor, D. R. (1992). Nonstationary Function Optimization using the Structured Genetic Algorithm. In *Parallel Problem Solving from Nature 2*, Brussels, Belgium. Elsevier Science Pub. (pp. 145-154)

Davis, L. (ed.) (1987). *Genetic algorithms and simulated annealing*. London, Pitman.

Davis, L. (1991). *The Handbook of Genetic Algorithms*. Van Nostrand Reinhold, New York.

Dawkins, R., (1976). *The Selfish Gene*. Oxford University Press.

Dawkins, R, (1982). *The Extended Phenotype*. Oxford University Press.

Dawkins, R. (1986). *The Blind Watchmaker*. Longman Scientific & Technical Pub.

Dawkins, R. (1995). *River Out of Eden*. BasicBooks (HarperCollins Pub., Inc.).

Darwin, D. (1859). *The Origins of Species by Means of Natural Selection.* Penguin Classics, London.

Deb, K. (1991). Binary and Floating Point Function Optimization using Messy Genetic Algorithms. *Illinois Genetic Algorithms Laboratory* (IlliGAL), report no. 91004.

Deb, K. & Goldberg, D. E., (1991). mGA in C: A Messy Genetic Algorithm in C. *Illinois Genetic Algorithms Laboratory* (IlliGAL), report no. 91008.

Deb, K. & Goldberg, D. E., (1992). Sufficient Conditions for Deceptive and Easy Binary Functions. *Illinois Genetic Algorithms Laboratory* (IlliGAL), report no. 92001.

Deb, K. et. al., (1993). Multimodal Deceptive Functions. *Complex Systems* 7:2, 131-153.

Deb, K. & Goldberg, D. E., (1993). Analyzing Deception in Trap Functions. In *Foundations of Genetic Algorithms 2*, Morgan Kaufmann Pub.

de Garis, H., Iba, H. & Furuya, T. (1992). Differentiable Chromosomes, the genetic programming of switchable shape-genes. In *Parallel Problem Solving from Nature 2*, Brussels, Belgium. Elsevier Science Pub. (pp. 489-498)

De Jong, K. A., (1975). *An Analysis of the Behaviour of a Class of Genetic Adaptive Systems*. (Doctoral dissertation, University of Michigan), Dissertation Abstracts International.

Dickinson, S. J. & Bradshaw, A., (1995). Genetic Algorithm Optimisation and Scheduling for Building Heating Systems. In *Genetic Algorithms in Engineering Systems: Innovations and Applications* (GALESIA '95), Sheffield, (pp. 106-111).

Dixon, J. R., Simmons, M. K., (1984). An architecture for application of artificial intelligence to design. In proceedings of *ACM IEEE 21st Conference on design Automation*, Albuquerque, N.M., (pp. 634-640).

Donne, M. S., Tilly, D. G. and Richards, C. W., (1994). Adaptive Search and Optimisation Techniques in Fluid Power System Design. Proceedings of *Adaptive Computing in Engineering Design and Control - '94*, (pp.67-76).

Dowsland, K. A. (1995). Simulated Annealing Solutions for Multi-Objective Scheduling and Timetabling. In *Applied Decision Technologies*, London, 205-219.

Dyer, M. Flower, M. and Hodges, J., (1986). 'EDISON': an engineering design invention system operating naively. *Artificial Intelligence 1*, 36-44.

Dym, C. L., & Levitt, R. E., (1991). *Knowledge-Based Systems in Engineering*. McGraw-Hill Inc.

Fisher, K. A., (1995). The Application of Genetic Algorithms to Optimising the Design of an Engine Block for Low Noise. In *Genetic Algorithms in Engineering Systems: Innovations and Applications* (GALESIA '95), Sheffield, (pp.18-22).

Fogel, L. J., (1963). *Biotechnology: Concepts and Applications.* Englewood Cliffs, NJ: Prentice Hall.

Fogel, D. B., (1995). *Evolutionary Computation: Towards a New Philosophy of Machine Intelligence*. IEEE Press.

Foley, J., van Dam, A., Feiner, S., Hughes, J. (1990). *Computer Graphics Principles and Practice (second edition)*. Addison-Wesley.

Fonseca, C. DM, & Fleming, P. J. (1995a). An Overview of Evolutionary Algorithms in Multiobjective Optimization. *Evolutionary Computation v3:1*, 1-16.

Fonseca, C. M, & Fleming, P. J. (1995b). Multiobjective Genetic Algorithms Made Easy: Selection, Sharing and Mating Restriction. *Genetic Algorithms in Engineering Systems: Innovations and Applications* (GALESIA 95), Sheffield (pp. 45-52).

Foy, M. D., et al. (1992). Signal Timing Determination using Genetic Algorithms. *Transportation Research Record #1365*, National Academy Press, Washington, D.C., 108-113.

French, M. J., (1994). *Invention and Evolution: Design in Nature and Engineering*, 2nd Edition. Cambridge University Press.

Furuta, H., Maeda, K. & Watanabe, W. (1995). Apllication of Genetic Algorithm to Aesthetic Design of Bridge Structures. In *Microcomputers in Civil Engineering v10:6*. Blackwell Publishers, MA, USA, 415-421.

Garipov, V., Diakov, E., Semenkin, E. and Vakhtel, S., (1994). Adaptive Search Methods in Spacecraft Systems Optimal Design. Proceedings of *Adaptive Computing in Engineering Design and Control - '94*, (pp.194-201).

George, F. (1994). Hybrid Genetic Algorithms with Immunisation to Optimise Networks of Car Dealerships. *Edinburgh Parallel Computing Centre*.

Gero, J. S. & Louis, S. J. (1995). Improving Pareto Otimal Designs Using Genetic Algorithms. *Microcomputers in Civil Engineering v10:4*, Blackwell Publishers, MA, USA, 239-247.

Glaskin, M., (1995). Architects build on Dawinism. Article in *Sunday Times 3/12/95*, UK.

Goldberg, D. E., (1989). *Genetic Algorithms in Search, Optimization & Machine Learning*. Addison-Wesley.

Goldberg, D. E., (1991a). Real-coded Genetic Algorithms, Virtual Alphabets, and Blocking. *Complex Systems 5*, 139-67.

Goldberg, D. E., (1991b). Genetic Algorithms as a Computational Theory of Conceptual Design. In Proc.of *Applications of Artificial Intelligence in Engineering 6*, (pp.3-16).

Goldberg, D. E., & Rudnick, M. (1991). Genetic Algorithms and the Variance of Fitness. *Complex Systems 5*, 265-278.

Goldberg, D.E., Deb, K., Horn, J., (1992a). Massive multimodality, deception and genetic algorithms. *Parallel Problem Solving from Nature 2,* 37-46.

Goldberg, D. E., Horn, J., Deb, K., (1992b). What Makes a Problem Hard for a Classifier System? *Illinois Genetic Algorithms Laboratory* (IlliGAL), report no. 92007.

Goldberg, D. E. et al. (1992a). Genetic Algorithms, Noise, and the Sizing of Populations. *Complex Systems 6*, 333-62.

Goldberg, D. E. et al. (1992b). Accounting for Noise in the Sizing of Populations. In *Foundations of Genetic Algorithms 2,* Morgan Kaufmann Pub., (pp.127-140).

Goldberg, D. E. (1993). A Wright-Brothers Theory of Genetic-Algorithm Flight. システム/制御/情報 v37:8, 450-58.

Goldberg, D. E. (1994). Genetic and Evolutionary Algorithms Come of Age. *Communication of the ACM*, v.37:3, 113-119.

Greffenstette, J. J. (1991). Strategy Acquisition with genetic Algorithms. Ch. 12 in *The Handbook of Genetic Algorithms*. Van Nostrand Reinhold, New York, 186-201.

Hameyer, K. & Belmans, R. (1996). Stochastic Optimisation of Mathematical Models for Electric and Magnetic Fields. In Proceedings of *Adaptive Computing in Engineering Design and Control - '94*, (pp. 139-144).

Harp, S. and Samad, T. (1992). Genetic Synthesis of Neural Network Architecture. In Davis, L. (ed.), *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, 202-221.

Harris, R. A., (1994). An Alternative Description to the Action of Crossover. In Proceedings of *Adaptive Computing in Engineering Design and Control - '94*, (pp. 151-156).

Harvey, I., (1992). The SAGE Cross: The Mechanics of Recombination for Species with Variable-length Genotypes. *Parallel Problem Solving from Nature 2,* North-Holland, (pp. 269-278).

Harvey, I., Hasbands, P., Cliff, D., (1994). Seeing the Light: Artificial Evolution, Real Vision. In *Third International Conference on Simulation of Adaptive Behaviour* (Animals to Animats 3), (pp. 392-401).

Hawkes, N. (1995). Nothing Succeeds Like Symmetry. Article in the *Times 10/11/95*, London.

Hitching, F. (1982). *The Neck of the Giraffe, or Where Darwin Went Wrong.* London: Pan.

Holland, J. H., (1973). Genetic Algorithms and the optimal allocations of trials. *SIAM Journal of Computing* 2:2, 88-105.

Holland, J. H., (1975). *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, Ann Arbor.

Holland, J. H., (1992). Genetic Algorithms. *Scientific American*, 66-72.

Horn, J., (1993). Finite Markov Chain Analysis of Genetic Algorithms with Niching. In Proceedings of the *Fifth International Conference on Genetic Algorithms*, Morgan Kaufmann Pub., (pp. 110-17).

Horn, J. & Nafpliotis, N. (1993). Multiobjective Optimisation Using the Niched Pareto Genetic Algorithm. *Illinois Genetic Algorithms Laboratory* (IlliGAL), report no. 93005.

Horn, J. et. al., (1994). Implicit Niching in a Learning Classifier System: Nature's Way. *Illinois Genetic Algorithms Laboratory* (IlliGAL), report no. 94001.

Horner, A. & Goldberg, D. E. (1991). Genetic Algorithms and Computer-Assisted Music Composition. In *Fourth Internation Conference on Genetic Algorithms*, (pp.437-41).

Husbands, P. (1994). Distributed Coevolutionary Genetic Algorithms for Multi-Criteria and Multi-Constraint Optimisation. In *Evolutionary Computing, AISB Workshop Selected Papers*, Fogarty, T. (ed.), Springer-Verlag, Lecture Notes in Computer Science Vol. 865, (pp. 150-165).

Husbands, P., Harvey, I, Cliff, D., Thompson, A. & Jakobi, N. (1996). The Artificial Evolution of Robot Control Systems. In *Proc. of the 2nd Int. Conf. on Adaptive Computing in Engineering Design and Control - '96*. University of Plymouth, UK, (pp. 41-49).

Husbands, P., Jermy, G., McIlhagga, M., & Ives, R. (1996). Two Applications of genetic Algorithms to Component Design. In *Selected Papers from AISB Workshop on Evolutionary Computing*. Fogarty, T. (ed.), Springer-Verlag, Lecture Notes in Computer Science, (pp. 50-61).

Joy, K. I. (1991). Utilizing Parametric Hyperpatch Methods for Modeling and Display of Free-Form Solids. In *Symposium on Solid Modeling Foundations and CAD/CAM Applications*, Rossingnac, J. & Turner, J. (eds), (pp.245-254).

Kanal, L. and Cumar, V. (Eds) (1988). *Search in Artificial Intelligence*. Spring-Verlag Pub.

Kant, E. (1988). Interactive problem solving using task configuration and control. *IEEE Expert*, 36-49.

Kargupta, H. et al. (1992). Ordering genetic algorithms and deception. *Parallel problem Solving from Nature 2*, 47-56.

Kargupta, H., (1993). Information Transmission in Genetic Algorithm and Shannon's Second Theorem. *Illinois Genetic Algorithms Laboratory* (IlliGAL), report no. 93003.

Keane, A. J. & Brown, S. M., (1996). The Design of a Satellite Boom with Enhanced Vibration Performance using Genetic Algorithm Techniques. In Proceedings of *Adaptive Computing in Engineering Design and Control - '96*, (pp. 107-113).

King, E. G. et al., (1993). The Use of a Genetic Algorithm for Minimum Length Nozzle Design: A Process Overview. In Proc. of *Third Workshop on Neural Networks: Academic/Industrial/NASA/Defence*, (pp. 556-563).

Klein, M. V., (1970). *Optics*. John Wiley and Sons, Inc., London.

Koakutsu, S. et al., (1992). Floorplanning by Improved Simulated Annealing Based on Genetic Algorithm. *IEE Transactions of the Institute of Electrical Engineers of Japan,* Part C v112-C:7, 411-417.

Koza, J. R., (1990). Genetic Programming: A paradigm for genetically breeding populations of computer programs to solve problems. Technical Report STAN-CS-90-1314, Stanford University.

Krishnakumar, K. & Goldberg, D. E. (1992). Control System Optimization using Genetic Algorithms. *Journal of Guidance, Control and Dynamics* v15:3, 735-740.

Kuethe, A. M. & Schetzer, J. D. (1959). *Foundations of Aerodynamics*. John Wiley & Sons Inc., London.

Lansbury, J. E., Wozniak, L., & Goldberg, D.E. (1992). Optimal Hydrogenerator Governor Tuning with a Genetic Algorithm. *IEEE Transactions on Energy Conversion*, v7:4.

Levine, D, (1994). *A Parallel Genetic Algorithm for the Set Partitioning Problem*. D. Phil dissertation, Argonne National Laboratory, Illinois, USA.

Libes, D., (1990). EXPECT: Curing those uncontrollable fits of inaction. In Proceedings of the *Summer 1990 USENIX Conference*, Anaheim, California.

Linkens, D. A. & Nyongesa, H. O. (1993). A Distributed Genetic Algorithm for Multivariable Fuzzy Control. In *IEE Colloquium on Genetic Algorithms for Control Systems Engineering*, Digest No. 199/130, (pp.9/1 - 9/3).

Lomborg, B. (1991). The Structure of Solutions in the Iterated Prisoner's Dilemma. *Institute of Politcal Science*, University of Copenhagen.

Maher, M. L., Zhao, F. and Gero, J., (1989). An approach to knowledge-based creative design. In *Preprints of Engineering Design Research Conference*, Amherst, Mass., (pp. 333-346) (National Science Foundation).

Mahfoud, S. W. (1993a). Finite Markov Chain Models of an Alternative Selection Strategy for the Genetic Algorithm. *Complex Systems 7*, 155-70.

Mahfoud, S. W. (1993b). Simple Analytical Models of genetic Algorithms for Multimodal Function Optimisation. In *Fifth international Conf. on Genetic Algorithms*, (pp.643).

Marett, R. & Wright, M. (1995). The Value of Distorting Subcosts When Using Neighbourhood Search Techniques for Multi-objective Combinatorial Problems. In *Applied Decision Technologies*, London, (pp.189-202).

McMahon, C. A., Xianyi, M. and Sims Williams, J. H., (1994). A Network Approach to System Integration in Design Optimisation. In Proceedings of *Adaptive Computing in Engineering Design and Control - '94,* (pp. 109-113).

Meyer, J. & Guillot, A. (1994). From SAB90 to SAB94: Four Years of Animat Research. In *Third International Conference on Simulation of Adaptive Behaviour* (Animals to Animats 3), (pp. 2-11).

Meyer-Arendt, J. R., (1989). *Introduction to Classical and Modern Optics* (Third Edition). Prentice-Hall International, Inc.

Michielssen, E.; Ranjithan, S.; Mittra, R., (1992). Optimal multilayer filter design using real coded genetic algorithms. *IEE Proceedings-J (Optoelectronics)* v139:6, 413-420.

Morris, J. and Martin, E. (1996). Multivariate Methods in process Monitoring, Modelling and Control. In *Proc. of the 2nd Int. Conf. on Adaptive Computing in Engineering Design and Control - '96.* University of Plymouth, UK, (pp. 26-40).

Mühlenbein, H. (1991). Darwin's continent cycle theory and its simulation by the Prisoner's Dilemma. *Complex Systems 5*, 459-478.

Obavashi, S. and Takanashi, S., (1995). Genetic Algorithm for Aerodynamic Inverse Optimsation Problem. In *Genetic Algorithms in Engineering Systems: Innovations and Applications* (GALESIA '95), Sheffield, (pp.7-12).

Oppacher, F. & Deugo, D. (1995). The Evolution of Heirarchical Representations. In *'Advances in Artifical Life, Proceedings of the Third European Conference on Artificial Life.*(pp. 302-313), Springer-Verlag, Berlin, Germany.

Pahl, G. and Beitz, W., (1988). *Engineering Design: A Systematic Approach*. The Design Council, Springer-Verlag, London.

Parmee, I C (Ed.) (1994). *Proceedings of the Adaptive Computing in Engineering Design and Control -'94*, PEDC, Plymouth, September 1994.

Parmee, I C & Denham, M J, (1994). The Integration of Adaptive Search Techniques with Current Engineering Design Practice. In Proc of *Adaptive Computing in Engineering Design and Control -'94*, Plymouth, (pp. 1-13).

Parmee, I C (Ed.) (1996). *Proceedings of the Second Inernational Conference on Adaptive Computing in Engineering Design and Control -'94,* PEDC, Plymouth, 26-28 March 1996.

Paton, R. (1994). Enhancing Evolutionary Computation using Analogues of Biological Mechanisms. In *Evolutionary Computing, AISB Workshop.* (pp. 51-64). Springer-Verlag, Germany.

Pham, D. T. & Yang, Y., (1993). A genetic algorithm based preliminary design system. *Journal of Automobile Engineers* v207:D2, 127-133.

Pope, A. & Harper, J. J. (1966). *Low-Speed Wind Tunnel Testing.* John Wiley & Sons, Inc., London.

Radcliffe, N. J. (1992). Non-linear Genetic Representations. In *Parallel Problem Solving from Nature 2*, Brussels, Belgium. Elsevier Science Pub. (pp. 259-268)

Radcliffe, N. J. & George, F. A. W. (1993). A Study in Set Recombination. In *Proceedings of the fifth internation conference on Genetic Algorithms*, Urbana-Champaign, USA. Morgan Kaufmann Pub. (pp. 23-30)

Radcliffe, N. J. & Surry, P. D., (1994a). Formal Memetic Algorithms. *Edinburgh Parallel Computing Centre*.

Radcliffe, N. J., Surry, P. D. (1994b). Co-operation through Hierarchical Competition in Genetic Data Mining. (Submitted to) *Parallel Problem Solving From Nature*.

Rayward-Smith, V. (Stream ed.), (1995). *Applied Decision Technologies: Modern Heuristic Search Methods*. Conf. Proc., London, 3-5 April 1995.

Rechenberg, I., (1973). *Evolutionsstrategie: Optimierung Technisher Systeme nach Prinzipien der Biologischen Evolution.* Stuttgart: Fromman-Holzboog Verlag.

Reeves, C. Steele, N. and Liu, J., (1994). Tabu Search and Genetic Algorithms for Filter Design. Proceedings of *Adaptive Computing in Engineering Design and Control - '94*, (pp. 117-121).

Rosenman, M. A., (1996). A Growth Model for Form Generation Using a Hierarchical Evolutionary Approach. *Microcomputers in Civil Engineering* 11:3, 163-174.

Rosvany, G. I. N., & Zhou, M., (1994). Optimality criteria methods for large discretized systems. In Adeli, H. (ed) *Advances in Design Optimization.* Ch. 2, Chapman & Hall, 41-108.

Ryan, C. (1994). Pygmies and Civil Servants. Advances in Genetic Programming. In Kinnear, K. (ed), *Advances in Genetic Programming*, MIT Press.

Savic, D. A. and Walters, G. A., (1994). Evolution Programs in Optimal Design of Hydraulic Networks. In Proceedings of *Adaptive Computing in Engineering Design and Control - '94*, (pp. 146-150).

Schaffer, J. D. (1984). *Some experiments in machine learning using vector evaluated genetic algorithms*. PhD dissertation, Vanderbilt University, Nashville, USA.

Schaffer, J. D., (1985). Multiple Objective Optimization with Vector Evaluated Genetic Algorithms. In *Genetic Algorithms and Their Applications: Proceedings of the First International Conference on Genetic Algorithms*, (pp. 93-100).

Schaffer, J. D. and Eshelman, L. (1995). Combinatorial Optimization by Genetic Algorithms: The Value of Genotype/Phenotype Distinction. In Proc. of *Applied Decision Technologies* (ADT '95), April 1995, London, (pp. 29-40).

Schnecke, V. and Vornberger, O., (1995). Genetic Design of VLSI-Layouts. In *Genetic Algorithms in Engineering Systems: Innovations and Applications* (GALESIA '95), Sheffield, (pp.430-435).

Scibor-Rylski, A. J. (1975). *Road Vehicle Aerodynamics.* Pentech Press, London.

Shankar, N. & Hajela, P., (1991). Heuristics Driven Strategies for Near-Optimal Structural Topology Development. In Proc. of *Artificial Intelligence and Structural Engineering*, Oxford, (pp. 219-226).

Sims, K. (1994a). Evolving Virtual Creatures. In *Computer Graphics*, Annual Conference Series, (SIGGRAPH '94 Proceedings), July 1994 (pp. 15-22).

Sims, K. (1994b). Evolving 3D Morphology and Behaviour by Competition. In *Artificial Life IV Proceedings*, Brooks, R. & Maes, P. (ed.s), MIT Press (pp. 28-39).

Smith, S. F., (1984). Adaptive Learning Systems. In Forsyth, R. (ed.), *Expert Systems: Principles and case studies*. New York: Chapman and Hall, 169-189.

Smith, R. E. & Goldberg, D. E. (1992a). Reinforcement Learning with Classifier Systems. *Applied Artificial Intelligence v6*, 79-102.

Smith, R. E. & Goldberg, D. E. (1992b). Diploidy and Dominance in Artificial Genetic Search. *Complex Systems 6*, 251-285.

Sriranganathan, S., Bull, D. R. and Redmill, D. W., (1995). Design of 2-D Multiplierless FIR Filters Using Genetic Algorithms. In *Genetic Algorithms in Engineering Systems: Innovations and Applications* (GALESIA '95), Sheffield, (pp.282-286).

Srinivas, N. & Deb, K. (1995). Multiobjective Optimzation Using Nondominated Sorting in Genetic Algorithms. *Evolutionary Computation*, v2:3, 221-248.

Sun, Y. & Wang, Z. (1992). Interactive Algorithm of Large Scale Multiobjective 0-1 Linear Programming. In *Sixth IFAC/IFORS/IMACS Symposium on Large Scale Systems, Theory and Applications*, (pp.83-86).

Syswerda, G. (1989). Uniform Crossover in Genetic Algorithms. In Schaffer, D. (ed.), *Proc. of the Third Int. Conf. on Genetic Algorithms*. Morgan Kaufmann Pub.

Syswerda, G. & Palmucci, J. (1991). The Application of Genetic Algorithms to Resource Scheduling. *Genetic Algorithms: Proceedings of the Fourth International Conference,* Morgan Kaufmann (pp. 502-508).

Tennant, A. and Chambers, B., (1994). Adaptive Optimisation Techniques for the Design of Microwave Absorbers. In Proceedings of *Adaptive Computing in Engineering Design and Control - '94*, (pp. 44-49).

Todd, S. & Latham, W., (1992). *Evolutionary Art and Computers*. Academic Press.

Thompson, D., 1961. *On growth and form*. Abridged Edition (J. T. Bonner, Ed.) Cambridge University Press.

Thompson, A. (1995). Evolving Fault Tolerant Systems. In *Genetic Algorithms in Engineering Systems: Innovations and Applications*, IEE Conf. Pub. No. 414, (pp. 524-529).

Tong, S.S., (1992). Integration of symbolic and numerical methods for optimizing complex engineering systems. *IFIP Transactions (Computer Science and Technology)* vA-2, 3-20.

Tributsch, H., (1982). *How Life Learned to Live*. MIT Press, Cambridge, MA.

Ulrich, K. T. and Seering, W. P., (1987). Conceptual design as novel combination of existing device features. *Advances in design automation* - Thirteenth Annual Conference, Boston, Mass.,(ASME), (pp. 295-300).

Vincent, T. L. (1992). Game theory as a Design Tool. *Journal of Mechanisms, Transmissions & Automation in Design* v105, 165-70.

Watabe, H. & Okino, N. (1993). A Study on Genetic Shape Design. In *Proceedings of the fifth internation conference on Genetic Algorithms*, Urbana-Champaign, USA. Morgan Kaufmann Pub. (pp. 445-450)

Whitley, D. & Starkweather, T., (1990). GENITOR II: a distributed genetic algorithm. *Journal of Experimental and Theoretic Artificial Intelligence* v2:3, 189-214.

Williams, B. C., (1990). Visualising potential interactions: constructing novel devices from first priciples. In proceedings of *Eighth National Conference on Artificial Intelligence* (AAAI-90), Boston, Mass.

Yamada, T. & Nakano, R. (1995). A genetic algorithm with multi-step crossover for job-shop scheduling problems. In *Genetic Algorithms in Engineering Systems: Innovations and Applications*, IEE Conf. Pub. No. 414, (pp. 146-151).

Appendix A

PRIMITIVE EXTRACTION ALGORITHM

This algorithm is used to extract the positions of the sides and vertices of a primitive from the nine definition parameters within a phenotype. It is used within some phenotype information modules (described in Chapter Seven) and to allow the graphical display of the primitives defined by phenotypes (see end of Chapter Seven).

Use (*x, y, z, width, height, depth*) to generate the 8 vertices;

(the 8 vertices define 6 sides and 12 edges)

Calculate the new vertices (if any) generated by the plane (*angle1, angle2, distance*) intersecting the 12 edges.

Calculate the distance of each vertex from the plane, if the value is positive, remove that vertex. (Remove any vertices that are on the outside of the plane.)

Divide vertices into groups of coplanar vertices - each group defines a side

(each vertex will be shared amongst three different groups)

Sort the vertices for each side into the correct order for displaying

(i.e. all vertices of a side must be connected by edges, but no 2 edges should intersect)

HIERARCHICAL CROSSOVER ALGORITHM

STAGE 1

Establish point(s) of similarity (POS) between parent 1 and parent 2:
(i.e. pick a viable hierarchical crossover point for both parents)

```
start at roots of both parents

loop until finished
{       pick at random a child node common to both parents (not previously picked)
        if successful then
        {       store node as a POS
                traverse down to node in both parents
                if node is a leaf of both parents then
                {       finished
                }
                if node is a leaf of only one parent, but not the other then
                {       reject node as a POS
                        traverse back up to parent of node
                }
        }
        else (no nodes in common or all previously picked)
        {       if at root then
                {       parents have no point of similarity and cannot reproduce
                        give up (finished)
                }
                else
                {       reject node as a POS
                        traverse back up to parent of node
                }
        }
}
```

STAGE 2

Perform hierarchical crossover at POS to generate new offspring:
(*Replacing with (node < POS) here will make hierarchical crossover equivalent to uniform single-point crossover, but a purely random choice will mix parents' genes more thoroughly.)
(**Using normal crossover here instead will mix any fixed length genes or blocks of genes.)

```
start at roots of both parents
find/create new roots for children

loop until finished
{       pick from parent 1 the first child node ¹ POS[current depth] not picked before
        if successful then
        {       pick from parent 2 the first corresponding child node not picked before
                if toss_a_coin equals heads* then
                {       attach child node (with all connected sub-nodes) from parent1 to
                          current node of child1
                        attach child node (with all connected sub-nodes) from parent2
                          (if it exists) to current node of child2
                }
                else (if it equals tails)
                {       attach child node (with all connected sub-nodes) from parent1 to
                          current node of child2
                        attach child node (with all connected sub-nodes) from parent2
```

```
                     (if it exists) to current node of child1
            }
        }
        else (parent 1 only has the POS node left)
        {   pick from parent 2 the first child node ¹ POS[current depth] not picked before
            if successful (parent 2 has an extra node) then
            {   if toss_a_coin equals heads* then
                {   attach child node (with all connected sub-nodes) to current
                      node of child2
                }
                else (if it equals tails)
                {   attach child node (with all connected sub-nodes) to current
                      node of child1
                }
            }
            else (parent 2 also only has the POS node left) then
            {   traverse down to child node in parent1 and parent2
                if the remaining two nodes are leaves then**
                {   if toss_a_coin equals heads
                    {   attach leaf from parent1 to current node of child1
                        attach leaf from parent2 to current node of child2
                    }
                    else (if it equals tails)
                    {   attach leaf from parent1 to current node of child2
                        attach leaf from parent2 to current node of child1
                    }
                    finished
                }
                else
                {   empty and extract both nodes from parents
                    attach one new empty node to each child
                    traverse down to empty node in child1 and child2
                }
            }
        }
}
```

MULTIOBJECTIVE FITNESS RANKING ALGORITHMS

Method 1: Sum of Weighted Objectives (SWO)

```
for every solution in the population
        solution_fitness  =  0
        for every criterion fitness_value 'n' of the solution
                solution_fitness = solution_fitness+weight [n]*fitness_value[n]
```

where weight [] is an array of pre-defined weights.

Method 2: Non-Dominated Sorting (NDS)

```
current_rank = 1
set every solution_rank = unranked
repeat
        for every solution in the population
                if solution_rank = unranked and
                   solution is not dominated by any unranked solutions
                        solution_rank = current_rank

        current_rank = current_rank + 1
until all solutions have been ranked
```

Method 3: Weighted Maximum Ranking (WMR)

```
for every objective in problem
        form a list of the fitness of each solution and pointer to this solution
                for current objective
        sort fitness_list into order of fitness

set every solution.rank = 99999

for every ranking position 'p' in population
        for every objective 'o' in problem
                if (fitness_list for 'o' [p] -> solution.rank > p / importance [o]
                then fitness_list for 'o' [p] -> solution.rank = p / importance [o]
```

where importance [] is an array of pre-defined 'importance' weights.

Method 4: Weighted Average Ranking (WAR)

```
for every objective in problem
        form a list of the fitness of each solution and pointer to this solution
                for current objective
        sort fitness_list into order of fitness

set every solution.rank = 0

for every ranking position 'p' in population
        for every objective 'o' in problem
                fitness_list for 'o' [p] -> solution.rank += p * importance [o]
```

where importance [] is an array of pre-defined 'importance' weights.

Method 5: Sum of Weighted Ratios (SWR)

```
for every objective 'o' in the problem
        max_fitness [o] = worst fitness_value in current population
        min_fitness [o] = best fitness values in current population

for every solution in the population
        for every criterion fitness_value 'n' of the solution
                fitness_value [n] = (fitness_value [n] - min_fitness [n]) /
                                        (max_fitness [n] - min_fitness [n])

for every solution in the population
        solution_fitness  =  0
        for every criterion fitness_value 'n' of the solution
                solution_fitness += importance [n] * fitness_value [n]
```

where importance [] is an array of pre-defined 'importance' weights.

Method 6: Sum of Weighted Global Ratios (SWGR)

```
for every objective 'o' in the problem
        max_fitness [o] = worst fitness_value ever, of all populations
        min_fitness [o] = best fitness values ever, of all populations

for every solution in the population
        for every criterion fitness_value 'n' of the solution
                fitness_value [n] = (fitness_value [n] - min_fitness [n]) /
                                        (max_fitness [n] - min_fitness [n])

for every solution in the population
        solution_fitness  =  0
        for every criterion fitness_value 'n' of the solution
                solution_fitness += importance [n] * fitness_value [n]
```

where importance [] is an array of pre-defined 'importance' weights.

SYSTEM INITIALISATION COMMANDS

Name of system initialisation file: GADESIGN.INI
Any line beginning with the character '#' is ignored.

COMPOUND COMMANDS

GLOBAL:
EVALUATE . . . END
VALUES *primtive_num* . . . END

BOOLEAN COMMANDS

GLOBAL:
NO_IPLANES
X_SYMMETRY
Y_SYMMETRY
Z_SYMMETRY
INVARBLEPRIMS

WITHIN 'EVALUATE...END':
UNFRAGMENTED
MUSTHAVEVERTS
INTERSECTED

SINGLE PARAMETER COMMANDS

GLOBAL:

INT_POP	*#individuals*
EXT_POP	*#individuals*
MAX_GEN	*generations*
PRIMITIVES	*#primitives (-1 = random)*
LIFESPAN	*age*
BITMUTATE	*probability*
PRIMMUTATE	*probability*
PARENTS	*percentage*
INFLEXIBLE	*primitive_num*
TWODEE	*designwidth*

WITHIN 'VALUES...END':

XPOS	*value*
YPOS	*value*
ZPOS	*value*
WIDTH	*value*
HEIGHT	*value*
DEPTH	*value*
ANGLE1	*value*
ANGLE2	*value*

PLANEDIST *value*

WITHIN 'EVALUATE...END':
MASS *desired_value*
SURFACEAREA *desired_value*
SUPPORTIVENESS *mass*

MULTIPLE PARAMETER COMMANDS:

GLOBAL:
IMPORTANCE
i1 i2 i3 i4 i5 ...
FIXED *primitive_num*
b1 b2 b3 b4 b5 b6 b7 b8 b9

WITHIN 'EVALUATE...END':
SIZE
minleft minrght minbttm mintop minback minfrnt maxleft maxrght maxbttm maxtop maxback maxfrnt
FLATUPPERSURFACE
height width depth
FLATSURFACE
height xpos zpos width depth
RAYTRACING
xsrc ysrc zsrc xdrn ydrn zdrn xdest ydest zdest xrdrn yrdrn zrdrn A B C D xund yund zund refraction
PARTICLEFLOWSIM *#volumes*
left right back front bottom top xforce yforce zforce
...
left right back front bottom top xforce yforce zforce

Appendix B

The seven papers written as part of the research undertaken for this thesis follow:

1. Bentley, P J & Wakefield, J P (1996). The Evolution of Solid Object Designs using Genetic Algorithms. In Rayward-Smith, V. (ed) *Modern Heuristic Search Methods.* Ch. 12, John Wiley & Sons Inc., 199-215.

2. Bentley, P J & Wakefield, J P (1995). The Table: An Illustration of Evolutionary Design using Genetic Algorithms. In *Genetic Algorithms in Engineering Systems: Innovations and Applications* (GALESIA '95), Sept. 1995, Sheffield (pp. 412-418).

3. Bentley, P. J. & Wakefield, J. P. (1996). An Analysis of Multiobjective Optimisation within Genetic Algorithms. Submitted to *Evolutionary Computation*.

4. Bentley, P. J. & Wakefield, J. P. (1996). Conceptual Evolutionary Design by Genetic Algorithms. *Engineering Design and Automation Journal v2:3,* John Wiley & Sons, Inc.

5. Bentley, P. J. & Wakefield, J. P. (1996). Overview of a Generic Evolutionary Design System. In *Proceedings of the 2nd On-line Workshop on Evolutionary Computation*, (pp. 53-56), Nagoya University, Japan.

6. Bentley, P. J. & Wakefield, J. P. (1996). Hierarchical Crossover in Genetic Algorithms. In *Proceedings of the 1st On-line Workshop on Soft Computing* (WSC1), Nagoya University, Japan.

7. Bentley, P J & Wakefield, J P (1996). Generic Representation of Solid Geometry for Genetic Search. *Microcomputers in Civil Engineering 11:3,* Blackwell Publishers, 153-161.