

# The Evolution of Solid Object Designs using Genetic Algorithms

PETER J BENTLEY & JONATHAN P WAKEFIELD

DIVISION OF COMPUTER AND INFORMATION ENGINEERING,  
SCHOOL OF ENGINEERING, UNIVERSITY OF HUDDERSFIELD, QUEENSGATE,  
HUDDERSFIELD, WEST YORKSHIRE HD1 3DH, UK  
email: p.bentley@eng.hud.ac.uk

## ABSTRACT

This paper describes an attempt to enable computers to generate truly novel conceptual designs of three dimensional solid objects by using genetic algorithms (GAs). These designs are represented using spatial partitions of 'stretched cubes' with optional intersecting planes [1]. Each individual three-dimensional solid object has its functionality specified by an explicit objective function, which is utilised by GAs to evolve candidate designs.

## 1. INTRODUCTION

The genetic algorithm (GA) is a highly flexible and powerful search algorithm that uses principles of evolution observed in nature to direct its search [2,3]. GAs can tackle optimisation problems if these problems are formulated in terms of search, where the search space is the space containing all possible combinations of parameter values, and the solution is the point in that space where the parameters take optimal values. Indeed, GAs are commonly used to optimise designs, with some remarkable results [4]. However, in this paper we aim to demonstrate that GAs are able to do more than just optimise existing designs - we propose that they can create entirely new designs from scratch.

The area of design creation using genetic algorithms is a relatively unexplored area. Using GAs to create new designs has the potentially great benefit of new conceptual designs being automatically created in addition to optimal designs. Research has so far been performed in limited ways, such as the use of GAs to create new conceptual designs from high-level building blocks [5] although this often seems to be little more than the optimisation of connections between existing designs rather than true design by the GA. Genetic art is becoming more popular, with various voting systems now being on-line on the internet (e.g. John Mount's 'Interactive Genetic Art' at <http://robocop.modmath.cs.cmu.edu.8001>). Other art-evolution systems using humans as design evaluators have been created [6,7], but as yet very few, if any systems exist that can evolve a design from scratch with no human input during the evolution process. This paper describes an early prototype of an evolutionary design system, capable of doing just that.

## 2. REPRESENTATION

Evolving designs from scratch rather than optimising existing designs requires a very different approach to the representation of designs. When optimising an existing design, only selected parameters need have their values optimised (e.g. for a jet-turbine blade, such parameters could be the angle and rotation speed of the blade). To allow a GA to create a new design, the GA must be able to modify more than a small selected part of

that design - it must be able to modify every part of the design. This means that a design representation is required, which is suitable for manipulation by GAs. Many such representations exist, and some are in use by the evolutionary-art systems: a variant of constructive solid geometry (CSG) is used by Todd & Latham [7], others use fractal equations (e.g. John Mount - see the WWW address given above), and Dawkins [6] uses tree-like structures. For a system capable of designing a wide variety of different solid object designs, however, a more generic representation is needed.

A variant of spatial-partitioning representation has been developed for this work [1]. This representation uses spatial partitions of variable sizes, each one being a 'cuboid' with variable width, height, depth and position in space. Each primitive shape can also be intersected by a plane of variable orientation, but this refinement to the representation will not be considered in this paper. The main benefit of using such a variable-sized primitive shape as a spatial-partition is that few partitions are required to represent designs. Significantly, the fewer the partitions in a design, the fewer the number of parameters that need to be considered by the GA. The main disadvantage, however, is that it is possible to represent illegal designs using the representation, i.e. it is possible for two spatial-partitions to overlap each other causing redundancy and ambiguity.

Thus some form of correction is needed, to convert overlapping primitives into non-overlapping primitives. Since this correction process needs to be executed for every new candidate design in the population, every generation, it must be fast.

### 3. CORRECTING ILLEGAL DESIGNS

Various alternative methods for correcting illegal designs were examined:

#### METHOD 1

Intersecting primitives are located by detecting whether any corner of one primitive lies within another primitive, Fig. 1. If they are found to overlap, the positions of the primitives are examined relative to one another. If the difference in position between the two primitives is greatest in the x direction, the widths of the primitives are reduced, if the difference is greatest in the y direction, the heights are reduced, and for a greatest difference in the z direction, the depths are reduced, Fig 2. Using the positions to determine which sides to move ensures that the primitives are changed by the minimum amount possible. When reducing the dimensions, they are reduced until instead of overlapping, the appropriate sides are touching. To prevent too much unnecessary squashing, all other sides of the primitives are kept in the same place by moving the positions of the two primitives slightly.

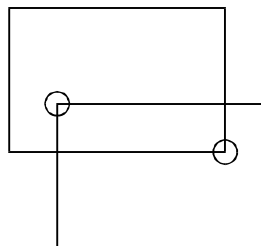


Fig. 1 Two primitive shapes are detected as overlapping:  
a corner of one primitive lies within the other.

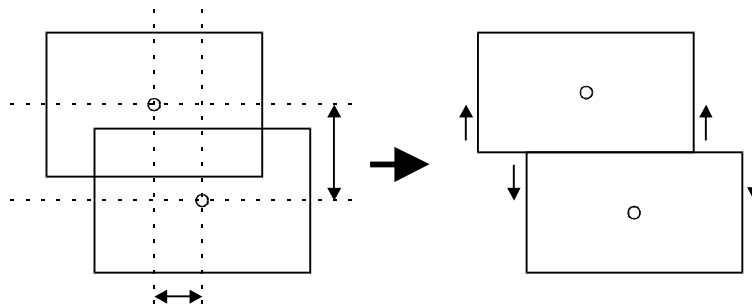


Fig. 2 Distance between overlapping primitives is greatest along y-axis: heights (and y positions) are changed.

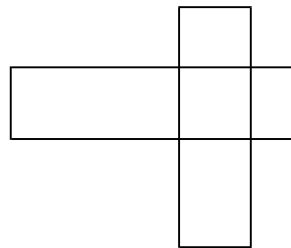


Fig. 3 Primitives not detected as intersecting: no corner of one lies within the other.

This method of squashing any overlapping primitives until they touch was found to be very successful. However, a flaw in the overlapping detection method became apparent: it is possible for two primitives to overlap without any of the corners of one being within the other, Fig. 3. This meant an alternative detection method was required.

#### METHOD 2

In this method, primitives begin 'life' as single points (specified by the centre positions of the primitives as described in the genotype). They are then grown in size by a small amount for a specific number of iterations. When two primitives grow to touch each other, the touching sides stop growing, otherwise the sides continue to grow until they reach the size specified by the genotype, Fig 4.

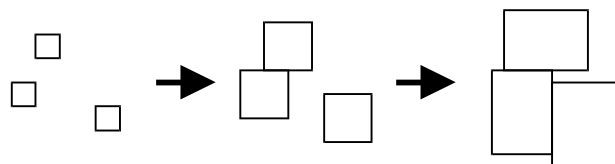


Fig. 4 Growing the primitives.

This method was found to work most of the time, and indeed mirrors nature attractively in that genotypes are mapped to phenotypes by a growth process. Another attraction was the way it was implemented - the time complexity was reduced from  $O(np^2)$  for the first method to  $O(np_c)$ , where  $n$  is the number of individuals in the population,  $p$  is the number of primitives in a design, and  $c$  is the constant (maximum) number of iterations the primitives were grown for.

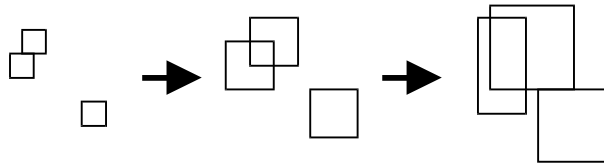


Fig. 5 Two primitives begin too close together and grow to overlap each other.

Unfortunately, the method did not work all of the time: it is unable to detect whether two primitives starting too close to one another and growing by too large an amount are touching or not, and so they grow to overlap each other, Fig. 5. A more serious problem though, was one of speed - despite being a potentially faster method for very high numbers of primitives in a design, it was actually a very slow method for usable numbers of primitives. Another method was needed.

### METHOD 3

This method of squashing overlapping primitives is really a combination of parts of the first two methods. Primitives, when detected as overlapping are squashed as described in the first method, Fig. 2. To detect whether the primitives were overlapping, an idea was taken and extended from the second method. This method examines the sides of the two primitives and if at least one side of one primitive is between two sides of the other primitive for all three axes, the two primitives are overlapping. The time complexity returns to  $O(np^2)$ , but because the detection is nothing more than a conditional check, (see Fig. 6) the process runs very quickly indeed.

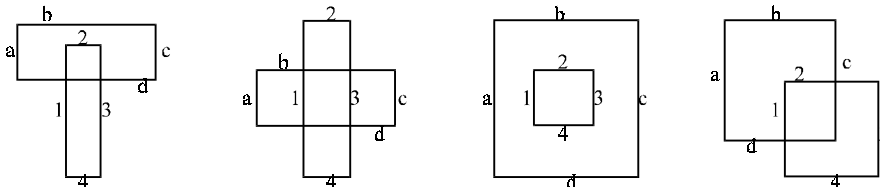


Fig. 6 If side 1 or 3 lies between a and c, or if side a or c lies between 1 and 3, and if side 2 or 4 lies between b and d or if side b or d lies between 2 and 4, the two primitives are overlapping.

So the third method provides an effective way to correct illegal designs. However, this creates the question: *where* should designs be corrected?

### WHERE TO CORRECT THE DESIGNS

Each candidate design (or phenotype) is defined by the representation, and coded as the genotype which is directly manipulated by the GA. The question is then, should incorrect designs be corrected in the genotype, in the phenotype, or should they be prevented from existing in the population at all?

To deal with the last, first: if every time the GA creates an illegal design, that design is discarded, the GA will take substantially longer to evolve designs, since illegal designs are often evolved. If the illegal designs are allowed to exist in a population, but are corrected in the genotype, the population of candidate designs becomes very static, with little evolution occurring. This is because a high proportion of all new designs evolved by the GA are illegal designs - if these are corrected every time, evolution is effectively being undone so the population stagnates.

So the solution must be to correct the phenotypes. This is achieved by allowing an illegal design (a design with spatial partitions overlapping) in the genotype, but correcting the design when it is mapped to the phenotype. In this way, the genotype no longer directly specifies the phenotype (or design), but instead gives directions that should be carried out if permissible. This method corresponds well with nature. For example, suppose that there is a single gene within a person's genotype that specifies how tall he should be (in reality it is probably a collection of genes). It might 'say' his ideal height should be 6'5" - but in real life he might only be 6'. Why? Because whilst growing, he was restricted by gravity. If he had grown up in space without the restriction of gravity, he may well have become 6'5".

Equally, the designs are restricted by the rules of the representation. Even though in the genotype, a primitive shape may be given a large width, when the genotype is mapped to the phenotype, that primitive may be squashed to a smaller size and new position by its neighbours. It is apparent that in nature, evolution takes into account such restrictions - our genes compensate for gravity when specifying our heights, so although the human genes may be asking for a height of 6'5" they 'know' that, because of gravity, this equates to 6'. Equally, as is shown by the experiments below, the evolution of the GA takes into account the restriction of the design representation, and compensates.

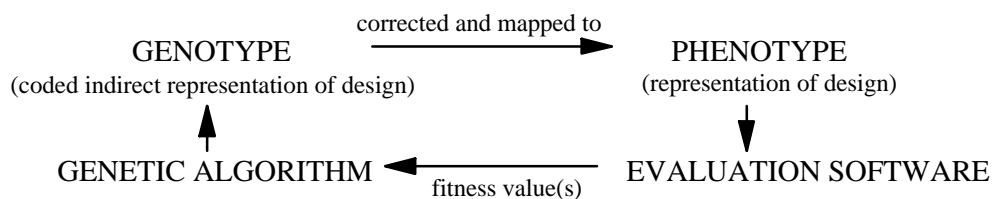


Fig. 7 Operation of the prototype evolutionary design system.

So, the GA evolves new designs by manipulating coded indirect representations in the genotypes, which are mapped to the direct representation of the phenotypes, Fig. 7. Some form of guidance is obviously necessary to direct the evolution of the GA. This is provided by simple evaluation software - effectively a software version of the design specification. There follows a series of experiments to demonstrate the way this evolutionary design system can evolve new designs from scratch, guided only by such evaluation software.

#### 4. THE EVOLUTION OF A TABLE

The genetic algorithm used for the following series of experiments remains unchanged throughout, apart from varying the population size and the number of generations it runs for. A basic canonical GA was used [8], with primitive 'roulette wheel' partner selection [3], real coding [9] and no special multi-objective function handling capabilities. Designs were all limited to five primitives and the population was initialised with primitives of random size and location at the beginning of every test run (i.e. starting from scratch). Evaluation software was utilised to describe the functionality of a table by the use of an objective function consisting of a weighted sum of a number of design objectives.

Five experiments were performed, with the evaluation software being made more sophisticated for each experiment by the addition of an extra design objective.

### *SIZE*

Perhaps the most basic requirement for a table is that of appropriate size: a table the size of a house or a matchbox is of no great use. The size of the design is specified by minimum and maximum extents for the left, right, back, front, top and bottom of the design. The fitness of a candidate design decreases the further it is from the ideal size. Using a population of 50 individuals, designs of the required size had evolved after 25 generations, Fig. 8.

### *MASS*

Another basic, but vital requirement is that of mass: a table weighing a ton would not be very practical. An ideal mass is defined, the fitness of the design decreasing the more the actual mass differs from the ideal mass. The ideal mass was defined as being a very low value. Designs with good fitnesses (the right size and very close to the ideal mass) had evolved after 100 generations, with a population of 50, Fig. 9.

The easiest way for the GA to create designs of low mass is to reduce the dimensions of the primitive shapes that make up the designs. This can produce fragmented designs however, where primitives become disconnected from the main part of the design. Since this work concentrates on the evolution of a single object at a time, such fragmented designs are heavily penalised. The fitness of fragmented designs is only based on the sum of the distance of each primitive from the origin (normally the centre of the design). This addition to the evaluation software means that in a population of fragmented designs, evolution will first create whole designs by drawing in the primitives towards the origin. In a population of whole designs, a fragmented design is very unfit and normally never reproduces.

### *STABILITY*

A more complex requirement is stability under gravity - the table must stand upright, and not fall over under its own weight. The stability of a candidate design is determined by calculating whether the centre of mass of the design falls within the outer extents of the primitive(s) touching the ground. (The ground is assumed to be at the lowest part of the design.) The more unstable the design is, the worse its fitness becomes. To help evolution, the population size was increased to 80 [10]. After 130 generations, some very fit designs had evolved (the right size, stable, and almost the right mass), Fig. 10 and 11. Strangely, evolution to good designs seemed remarkably improved with the addition of this new constraint.

### *FLAT SURFACE*

Perhaps the most important requirement of a table is its flat, upper surface - the table top. A good table should have a flat surface at an appropriate height from the ground, and covering an appropriate area. These criteria were evaluated by randomly sampling, five times, the top of the design within the required area of the table top, and checking how much the height at these points differed from the required height. (Analogous to five objects of negligible mass, e.g. feathers, being dropped onto the design within the area the flat surface is required. Should a feather be supported at too high or too low a height, or not be supported at all, the fitness of the design decreases proportionately.)

Since this evaluation is of a random nature, a design may be given a different fitness value every time it is evaluated (unless it fully satisfies the constraint). Not surprisingly, the GA found evolution to good designs a little harder to perform, so the population size was increased to 100, and the GA was allowed to run for 300 generations. The resulting designs were usually very good, but compromises were being made by the GA. It was

not always able to fully satisfy all of the constraints since, in order to have a large flat surface, the size of the primitives must be increased, but to have a low mass, the size of the primitives must be decreased. Results were often designs with excellent table tops, but too massive (Fig. 12), or designs with unusual multi-level table tops with the right mass (Fig. 13). All designs were still the right size and stable, however.

#### *SUPPORTIVENESS AND STABILITY*

One final requirement for a table is that it must be able to support objects on its flat upper surface. Although the tables evolved so far stand up on their own, if an object was to be placed on them, most would still topple over. Greater stability is therefore required. This is achieved by a simple extension to the 'stability under gravity' constraint, which requires the table to be able to support a heavy object placed at the very edges of its extremities without falling over. The mass of the object was deliberately set very high (twice the required mass of the table itself).

The GA was allowed to run for 500 generations, and produced some good designs. Again compromises were made - the GA quickly 'discovered' that an easy way to support a heavy object is to make the table even heavier, so it was necessary to increase the weighting of the 'low mass' constraint. Once this was done, the GA produced some remarkably fit designs, coming up with two main methods for increased stability: a very low centre of mass (Fig. 14) or a very wide base (Fig. 15), as well as producing designs the right size, nearly perfect flat surfaces, and almost the right mass.

#### *WHAT THE TABLES ARE NOT*

Although it should be apparent that the GA can indeed create novel, usable designs for tables, it is also clear that the tables do not resemble the sort of four-legged table most of us are used to. This should not really be surprising, since the GA has no world knowledge and so cannot be influenced by existing designs. (Also the small number of primitives combined with the initial random seeding of the population is a factor in producing some of the more unusual designs.) As the software stands, it is theoretically possible for the GA to evolve a four-legged table, but it is highly unlikely. It would require four primitive shapes to be precisely the same length so that they all touch the ground at once. Any primitive which was a fraction too long or too short would no longer behave as a leg, thus resulting in a poor design, so a mutation or crossover would have to produce all four legs in one unlikely jump.

However, it is not at all difficult to make the GA produce four-legged tables. By enforcing symmetry about the x and z axis, more traditional, symmetrical table designs can be evolved. (An alternative approach would be to seed the initial population with existing table designs and allow the GA to combine the best features of these, but this would no longer be designing from scratch.)

The tables produced are not evaluated for aesthetics, which would require human input, and no stress analysis takes place, so deficiencies in these areas are occasionally apparent.

## 5. CONCLUSIONS

The genetic algorithm is capable of more than design optimisation - it can evolve entirely new designs. To do this, a flexible representation is required, to allow the GA to modify every part of the design. As shown in this paper, the GA can automatically take into account the background correction of illegal designs (i.e. those not following the rules of the representation). It can cope with partially conflicting constraints (e.g. the low mass against the large table-top and greater stability), and randomly varying evaluations (e.g. the random sampling of the table-top evaluation) and still produce some excellent results.

In the 'evolution of a table' example, the multiobjective function defining the table was treated as a single objective function by weighting and summing the separate functions. Although the GA coped surprisingly well, this is not an ideal solution [3] and in the future the GA will be modified to evolve Pareto optimal solutions to such multicriteria design problems [11].

Future work will also be directed towards the creation of a system capable of evolving shapes other than 'cuboid' in appearance by allowing the GA to use the refinement to the representation, i.e. primitive shapes will be permitted to have portions 'sliced' off them. It is also anticipated that the GA can be made to evolve not only the position and dimensions of the primitive shapes, but also the number of primitives making up a design.

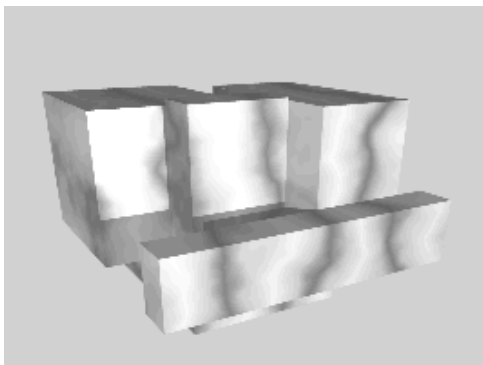


Fig. 8 Evaluation of size only

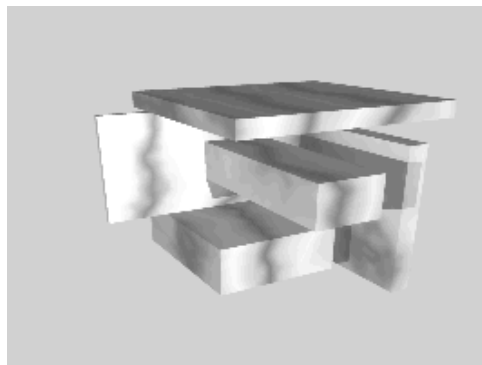


Fig. 9 Evaluation of size and low mass

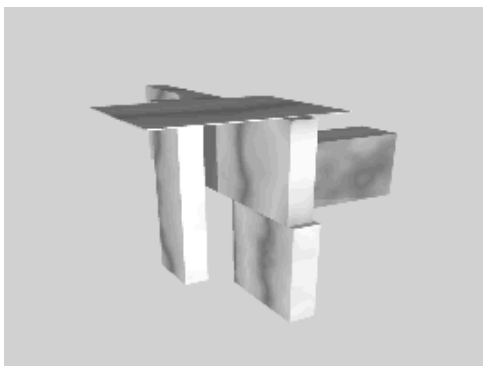


Fig. 10 Evaluation of size, low mass and stability

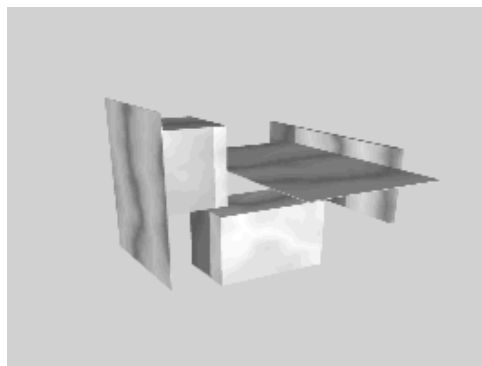


Fig. 11 Evaluation of size, low mass and stability

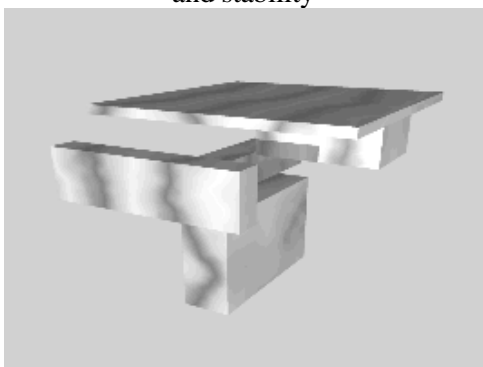


Fig. 12 Evaluation of size, low mass, stability and flat top

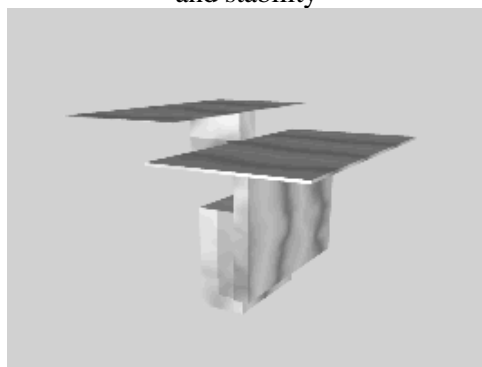


Fig. 13 Evaluation of size, low mass, stability and flat top



Fig. 14 Evaluation of size, low mass, flat top and greater stability

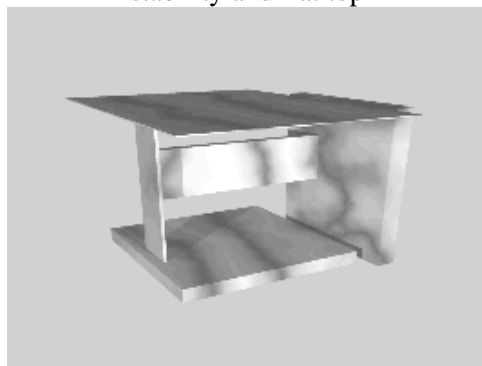


Fig. 15 Evaluation of size, low mass, flat top and greater stability

## 6. REFERENCES

- [1] Bentley, P J & Wakefield, J P (1994). Generic Representation of Solids for Genetic Search. To be published in *Microcomputers in Civil Engineering*.
- [2] Holland, J H (1975). *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, Ann Arbor.
- [3] Goldberg, D E (1989). *Genetic Algorithms in Search, Optimization & Machine Learning*. Addison-Wesley, USA.
- [4] Parmee, I C & Denham, M J (1994). The Integration of Adaptive Search Techniques with Current Engineering Design Practice. Proc from *Adaptive Computing in Engineering Design and Control -'94*, Plymouth, 1-13.
- [5] Pham, D T & Yang, Y (1993). A Genetic Algorithm based Preliminary Design System. *Journal of Automobile Engineers* v207:D2, 127-133.
- [6] Dawkins, R (1986). *The Blind Watchmaker*. Longman Scientific & Technical Pub.
- [7] Todd, S & Latham, W (1992). *Evolutionary Art and Computers*. Academic Press.
- [8] Mühlenbein, H (1992). Darwin's continent cycle theory and its simulation by the Prisoner's Dilemma, Conf. Proc. *1st European Conference on Artificial Life Towards a Practice of Autonomous Systems*. Paris, 236-244.
- [9] Goldberg, D E (1991). Real-coded Genetic Algorithms, Virtual Alphabets, and Blocking. *Complex Systems* 5, 139-167.
- [10] Goldberg, D E, Deb, K, Clark, J H (1992). Genetic Algorithms, Noise, and the Sizing of Populations. *Complex Systems* 6, 333-362.
- [11] Horn, J & Nafpliotis, N (1993). Multiobjective Optimization Using the Niche Pareto Genetic Algorithm. IlliGAL Report No. 93005, Illinois Genetic Algorithms Laboratory.