# Object-based Design Modeling and Optimization with Genetic Algorithms

Nicola Senin, David R. Wallace[1], and Nick Borland

## Abstract

**DOME (Distributed Object-based Modeling Environment) is a software framework for product design system modeling where designers are distributed geographically and make use of different software tools. Designers develop their own local software components, and distributed object technology is used to integrate their services via the Internet to form an overall system model. Designers can then explore alternatives by making changes to local models or remote services while observing how the entire model responds.**

**This exploration is amenable to automated search, which involves both continuous parameters (changing the value of services) and discrete changes (selecting different objects to substitute entire local models). A genetic optimization object and appropriate direct representation genomes and operators were developed for this purpose. The effectiveness of several genetic algorithms was compared and a new variation of restricted tournament selection (RTS) was developed. The RTS variation, called the Struggle GA, most reliably located the global optima and the most local optima. Other crowding algorithms reliably located the global optima but were less successful identifying multiple local solutions. The global algorithms (simple and steady state) did not reliably locate the global optima in mixed variable problems. Finally, a realistic beverage container design example is presented.**

*Keywords: object-based design modeling, catalog-based design, genetic algorithms*

# 1. Introduction

## 1.1. Design scenario

This work builds upon a distributed object-based modeling formalism (Pahng, Senin et al. 1998). A beverage container design example will be used order to introduce key concepts. This scenario is illustrated in figure 1. In-house, the *design configuration* engineers select the main design parameters (materials, container type, dimensions) and perform the final evaluation of each design alternative (trading off cost, safety, capacity, and environmental impact). Geometric modeling and related computations are out sourced to an industrial design firm, while environmental impact assessment and life cycle modeling is provided by a consultant. Each design participant uses different software tools, models, and data, ranging from solid modelers to spreadsheets.

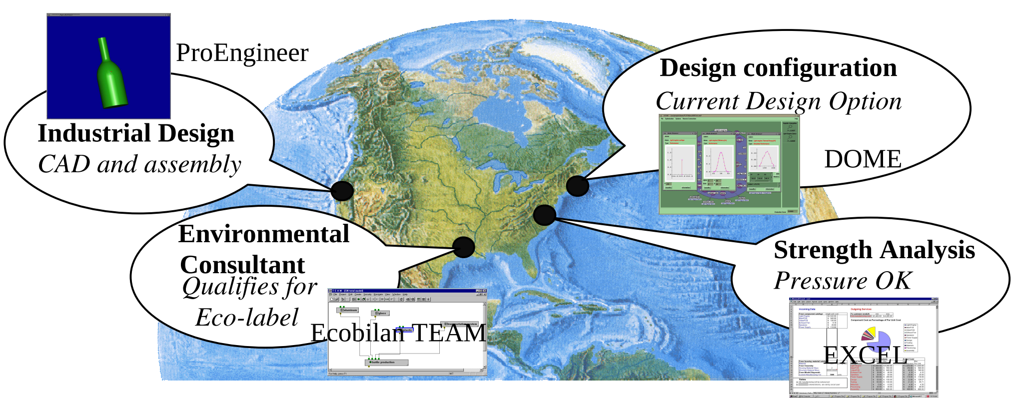



**Figure 1. Design scenario – Participants in the bottle design team are distributed geographically. Each presents their own viewpoint and software tools, but an integrated understanding of the complete product is needed.**

[1] Please address correspondence to this author, Room 3-435, 77 Massachusetts Avenue, Massachusetts Institute of Technology, Cambridge, MA, USA 02139. Email: drwallac@mit.edu, phone 617 253-2655, fax 617 253-9637

In developing new beverage containers, design options must be explored, trading off cost, styling, safety, material, and environmental considerations—all of which are highly interrelated but addressed by different participants. In current practice obtaining such an integrated view for just a single alternative is at best very time consuming, typically requiring months to coordinate and resolve. At worst, obtaining an integrated system view for making global tradeoffs is deemed intractable and decisions are left to intuition.

### 1.2. DOME formalism

DOME (Distributed Object-based Modeling Environment) is a software framework developed in the MIT CADLab to aid designers involved in distributed design modeling. DOME facilitates the creation of integrated models by allowing each design participant to embed their own software tools or models into *modules*. Modules are *distributed objects*, with their services mapped to the functionality of the software applications they encapsulate (Figure 2).

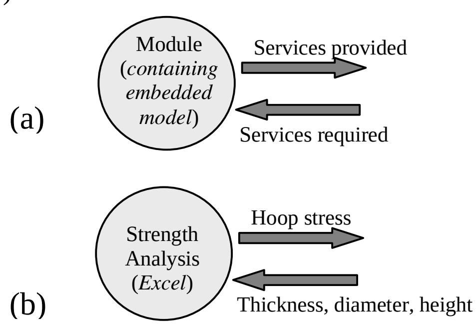



**Figure 2. (a) DOME modules and services. (b) The bottle strength analysis module computes hoop stress, but it needs to call services from other modules to obtain appropriate bottle parameters.**

These modules are CORBA compliant, and are made publicly available over the Internet to the other design participants. Thus each module works as a standalone software component, declaring in its public interface what services it can provide and what services must be provided to it. Designers build distributed models by locally defining mathematical relationships between services of different modules, thereby creating complex service exchange networks between DOME modules. Solving mechanisms ensure that local state changes propagate to all the other modules via service calls over the Internet, keeping the complete distributed model consistent with local design changes. In order to facilitate the service mapping process, basic engineering data types (probability distributions, intervals, functions, etc.) and mathematical relationship modeling are provided within the system.

The object-based modeling framework also provides tools for catalog-based design, allowing designers to store components or pieces of software (i.e., modules) in catalogs, from which they can be selected for use in a problem model. For example, in the bottle design problem the bottle geometric model, fluid contents, and different procedures for life-cycle assessment can be changed through catalog selections.

Further, decision-making support is provided by means of specialized evaluation modules (e.g. based on utility theory (Keeney and Raiffa 1976), or acceptability (Kim and Wallace 1997)).

### 1.3. Mixed variable search problem

Once a distributed object model has been created, each design participant can change the design variables (services) to which they have access: either real numbers or discrete catalog selections. Upon making a change, they can observe how it propagates to affect the model and the design's performance. For large problem models, many independent variables are under the control of many designers and finding combinations that correspond to acceptable performance may become a challenging task. When automated search techniques are applied to facilitate exploration, this becomes a *mixed variable* search problem involving both continuous parameters and discrete catalog selections.

### 1.4. Genetic Algorithm optimization module

The focus of this paper is to describe a genetic optimization object or module designed to operate on DOME models. However, the concepts are generally extensible to object-models. The services of the optimization module can be added to any DOME model. The optimization gains control over the distributed problem model by using services to manipulate design search variables, and using the services of decision support modules for objective formulation. Genome encoding schemes and a proper set of genetic operators is developed for continuous variables and catalog selections in the object-based modeling representation.

Several types of global and speciating algorithms are tested with the representation and a new crowding variation is developed.

## 2. Previous Work

Design problems can be modeled using a number of approaches. Heuristics or knowledge-based methodologies can be used (Borup and Parkinson 1992; Budaychevsky, Chertakov et al. 1993), or systems of equations representing relationships and constraints between design variables can be analytically solved (e.g. Design Sheet -Reddy, 1996). Extensions of traditional interval calculus can also be used to reduce the solution space by progressively eliminating infeasible solutions (Chen and Ward 1992; Lin and Ward

1992; Ward and Seering 1993; Ward and Seering 1993). Communicating object-based representations are amenable to large distributed model applications. GAs have been applied to many design applications where object-based models have been used, including catalog selection problems (Brown and Hwang 1993, Carlson, 1993 #19; Carlson and Pegg 1995; Carlson 1996). However, genome representations are typically developed on a problem specific basis rather than for the underlying object formalism.

*Crowding* (DeJong 1995) and *Fitness Sharing* (Goldberg and Richardson 1987) are two approaches for preserving diversity and maintaining multiple solutions in a population. In *Fitness Sharing* the population is forced to spread over the search space by de-rating the fitness of groups with a high concentration of similar individuals (i.e. overpopulated areas of the search space). Crowding techniques use the concept of competition only between offspring and similar individuals in the existing population. This paper focuses on different crowding techniques: *Deterministic Crowding* (DC) (Mahfoud 1995), a variation of De Jong's Crowding (DeJong 1995) where the offspring competes for survival with one the two parents; *Restricted Tournament Selection* (RTS) (Harik 1995), where the offspring competes for survival with the most similar individual from a subset which was randomly picked from the population; and Struggle GA (Grueninger and Wallace 1997), a variation of RTS.

# 3. Genetic Search Engine Implementation

## 3.1. Genome representation, operators, and similarity measure

Solutions for a DOME model are defined by a list of values for the independent search variables. Search variables are either real numbers defined over continuous intervals, integers, or discrete selections of modules from catalogs.
If *binary encoding* schemes are used the search variables are translated into a sequence of bits on the genome. A decoding process then translates each new offspring into the proper sequence of values to be fed to the problem model. However, *direct encoding* is used in this work. There is no conversion as the genome is a direct representation of the search variables. Thus, the genome is a hybrid mixture of real numbers (for continuous variables) and references to modules (for catalog selections).

Similarity measures are developed for use with crowding GAs and the genetic operators. Similarity is defined as the dimensionless ratio of the distance between two individuals and their maximum possible distance, defined by the boundaries of the search space. Assessing the distance between alternatives generically is often very difficult and subject to semantic interpretation. For binary encoding similarity can be evaluated by a pairwise comparison of bits (e.g., normalized Hamming distance) but the hybrid nature of a direct-coded genome requires a different similarity measure for each section of the genome. For real number pairs, the Euclidean distance was adopted (Figure 3-a), while for catalog selections, a distance measure based on the relative position of the two alternatives in the catalog was developed (Figure 3-b). The meaningfulness of module order in a catalog is judged by the designer. Ordered catalogs use this distance measurement while unordered catalogs treat all contents as equidistant. A similar approach is used for hierarchical catalogs, which are semantically interpreted as family trees.

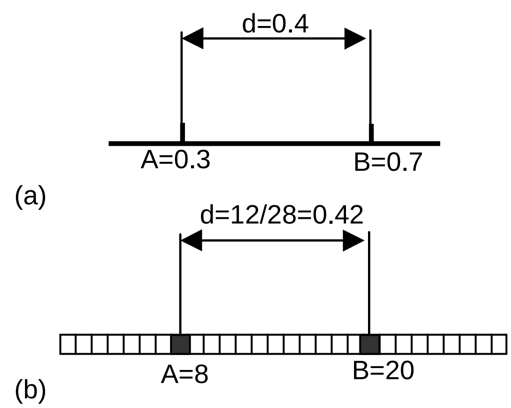



**Figure 3. Similarity measurements between two variables. (a) For real numbers the similarity is the absolute of the difference of the two values mapped in the interval 0-1. (b) For ordered catalogs (where relative position of the selections is meaningful) the similarity is the difference in the position of the two selections (distance) over the maximum distance. In the example, A and B occupy respectively the 8$^{th}$ and 20$^{th}$ position in a catalog of 28 modules.**

The similarity of each allele pair is aggregated through a weighted sum; the weights can be used by the designer to control the speciation process and emphasize differences in particular design variables. For example, in the bottle design problem the weights can be used to define bottle material and shape to be more significant similarity indicators than variables such as wall thickness.

With the distance and similarity measures defined, genetic operators are defined to manipulate the direct representation. For real number alleles, BLX crossover (Eshelman and Schaffer 1992) is applied to each pair of continuous variables (figure 4-a). The same operator is mapped to a discrete space for interpreting between two modules selected from catalogs (figure 4-b). This catalog crossover operator was also extended to hierarchical catalog trees.

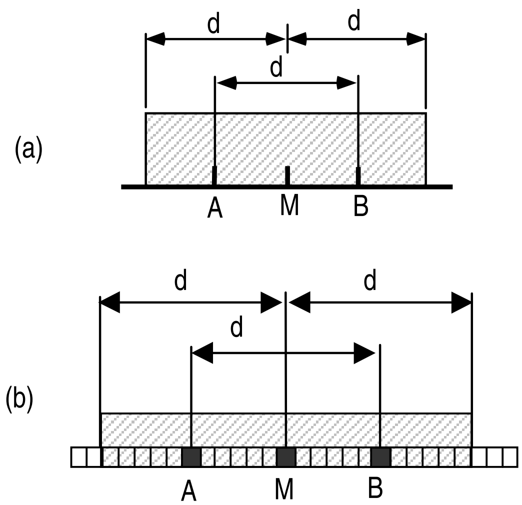



**Figure 4. (a) BLX operator for continuous variables, and its counterpart (b) for catalog selections. The same crossover mechanism is applied to a real interval for the pair of real numbers, and to the catalog discrete space, for a pair of catalog selections. The result of the crossover between A and B is randomly selected from a uniform (or Gaussian) distribution centered at the mean point M.**

## 3.2. Struggle GA

In addition to published algorithms, a variation of *Restricted Tournament Selection (RTS),* called the Struggle GA (STR), was developed in this work (Grueninger and Wallace 1997). In RTS, the offspring competes for survival with the most similar individual selected from a random subset (window) of the original population. Since the window does not encompass the whole population, there is the possibility that the new individual may not be competing with the most similar individual in the entire population. We have found that this *replacement error* can lead to loss of population diversity and reduce search reliability. The *Struggle* GA eliminates the population window—new solutions are compared to the most similar individual in the complete population as outlined below.

```
Randomly seed population of genomes
Repeat
       Select parents P1 and P2
       Cross P1 and P2 yielding offspring C
       Apply mutation with a probability p_mut
       on C, yielding C’
       Find the individual R that is most
       similar to C' in the entire population
       If fitnessScore(C’)>fitnessScore(R)
       replace R with C’
Until Stop Criterion
```

The Struggle GA (STR) uses uniform/random selection of parents. Chan (Chan 1997) developed a method for selecting parents where they are either uniformly chosen or the two most similar individuals are selected with a small probability (e.g. 0.2). This selection method (Struggle2 or Str2) accelerates convergence with negligible loss in search reliability or speciation.

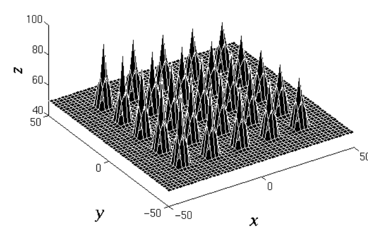
# 4. Implementation Results

In order to test the optimization module representation, several problem models were developed. Some of these problem models were standard test problems and other problems were real world engineering problems.

Results for a well known test problem (Scheckel’s Foxholes) and the bottle design application from the scenario are included in this paper. Both problems used the same GA settings: a crossover probability of 1.0 and a mutation rate of 0.03 were used for both continuous variables and catalog selections. Unless otherwise noted the population size was 20, as small populations are a must for computationally intensive real-world design applications. The window size for Restricted Tournament Selection was half of the population (10). The *Pmateclosest* parameter for the Str2 GA was 0.2. Runs were terminated after 300 generations (6000 design alternative evaluations) and all results are averaged over ten runs. The comparisons were run for Simple (S) GA, Stead-state (SS), Deterministic Crowding (DC), Restricted Tournament Selection (RTS), Struggle (Str), and Struggle2 (Str2). Fitness sharing algorithms are not presented as, even though they distributed the population, they were ineffective in converging on optima for both problems. Direct genome encoding with the proper set of genetic operators and distance measurement was used, unless otherwise specified.

## 4.1. Scheckel’s Foxholes results

Sheckel’s Foxholes is a continuous variable function from DeJong’s dissertation (DeJong 1995). The function is defined in Figure 5.




$$z_{(x,y)} = \cfrac{1}{\cfrac{1}{50} + \sum_{i=1}^{25} \cfrac{1}{1 + i + (x - a_i)^6 + (y - b_i)^6}}$$

$$b_i = 16\left[(i \div 5) - 2\right]$$

$$a_i = 16\left[(i \bmod 5) - 2\right]$$

**Figure 5. Sheckel’s Foxholes definition and surface plot.**

The foxholes function was modeled in DOME as a design problem where x and y are the continuous design parameters, and the function value is the objective. The object model is described in appendix A.
Figure 6 illustrates that all of the algorithms except the simple GA consistently locate the global optimum. Figure 7 illustrates the rate of convergence for the different algorithms.

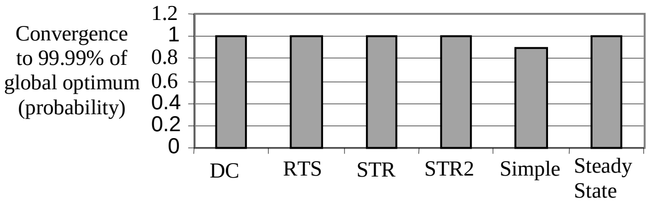



**Figure 6. Reliability finding Sheckels foxholes global optimum. Probability of converging within 99.99% of the known global optimum in 300 generations (averaged over 10 runs, popsize=20)**

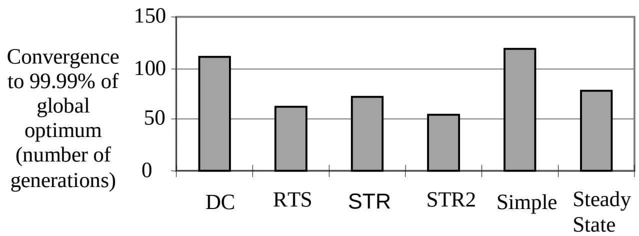



**Figure 7. Average number of generations to find Shekels foxholes global optimum (within 99.99% of known global optimum).**

Figure 8 provides an illustration of why the direct genome representation developed for the DOME formalism is preferred over a binary representation—success in locating the optima is very dependent upon the number of bits used in the binary representation.

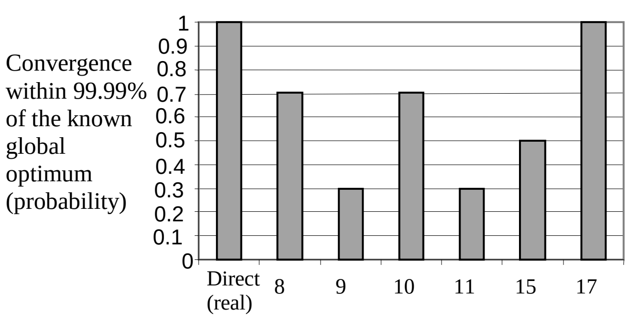



**Figure 8. Reliability finding Sheckels foxholes global optimum using direct or binary genome representations. Probability of converging within 99.99% of the known global optimum in 300 generations (Struggle GA, averaged over 10 runs, popsize=20)**

Additionally, the direct encoding required less than half of the evaluations needed for the high-bit binary representation to converge. A detailed explanation of these observations is provided in work by Gruninger and Wallace (Grueninger and Wallace 1997).

In addition to locating the global optimimum, effectiveness locating local optima was benchmarked in Figure 9. Variations of the struggle and restricted-tournament selection algorithms appear to be more effective than deterministic crowding—both algorithms reduce diversity loss through fewer replacement errors.

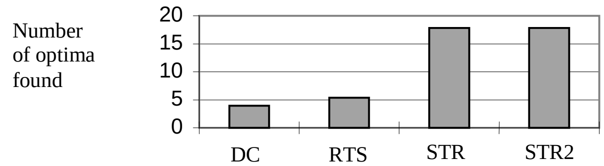



**Figure 9. Average number of Sheckels local optima found (within 99.99% of local peak). Averaged over 10 runs (popsize=20, number of local optima = 25)**

Figure 10 shows that with a population of 50 and after 500 generations all 25 local optima are identified and maintained using the struggle GA. None of the other algorithms were able to locate all optima in a single search.

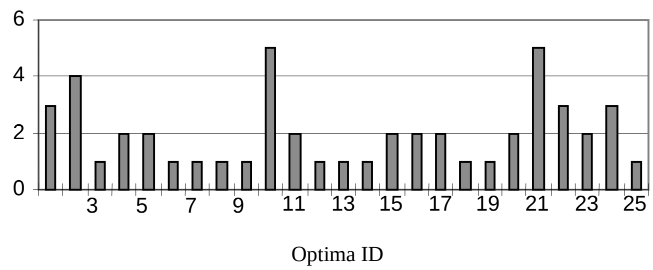



**Figure 10. Number of individuals on each local optima after 500 generations (within 99.99% of local peak) using struggle GA. Optima are ordered from best to worst. (population size = 50).**

## 4.2. Bottle design problem results

The bottle problem model is shown in Figure 11 from the engineering design configuration viewpoint GUI. The genetic *Optimization module* GUI is shown in Figure 12, during an optimization. There are three continuous search variables involved (container diameter, height, thickness), and one catalog selection (type of container—plastic bottle, aluminum can, and glass bottle). The overall problem model is distributed over four platforms and embeds three commercial applications (a CAD system, a spreadsheet, and an environmental impact assessment package). Evaluations on functionality, safety and environmental impact are computed within the model and contribute to the objective score.

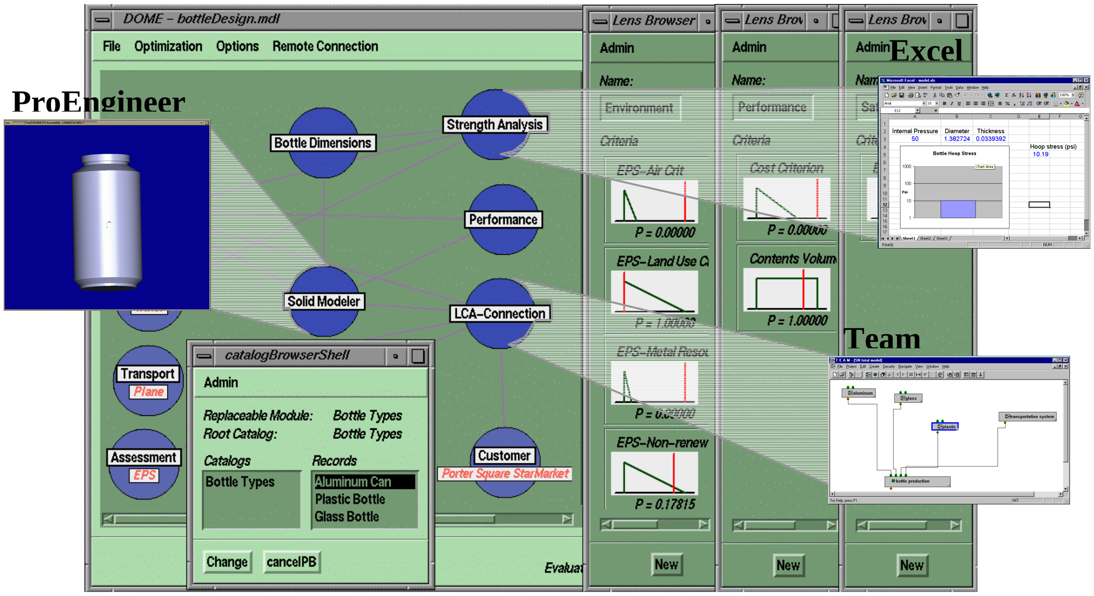



**Figure 11. The beverage model seen from the GUI for the engineering configuration module. The services from life-cycle assessment, strength analysis and industrial design are accessed through remote modules, which embed commercial applications. The aluminum can design configuration is being selected from a catalog**

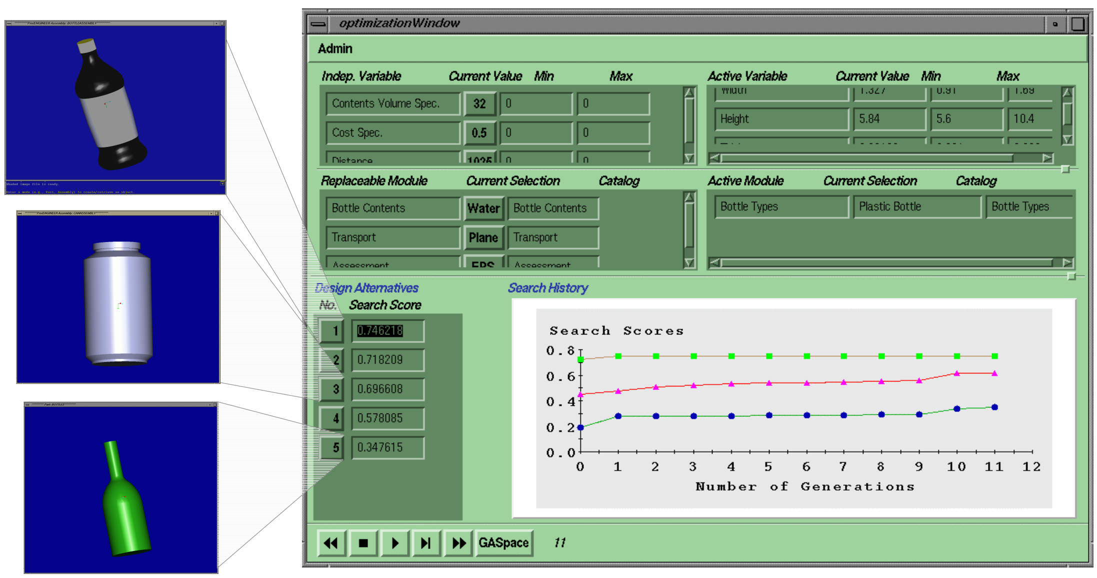



**Figure 12. Optimization results. Three locally optimal families of solutions correspond to the different bottle types. The plastic bottle performed best based upon the design goals.**

The bottom-center graph in Figure 12 shows the generation history of the optimization (best, average, and worst of generation). At the bottom left search results are listed, rank ordered by objective measure. Three major families of solutions were found each corresponding to a different bottle type.

## 5. Conclusions and Future Work

The DOME software framework provides an object-based modeling formalism and means to integrate commercial applications, in-house software and design-tailored tools for problem modeling and decision support into a single system. Large, distributed models of real-world engineering problems can be modeled and design variations can be explored.

A genetic optimization module was developed to aid designers in locating acceptable design configurations and to gain insight on the search space. Appropriate similarity measures and genome representations were developed and tested with several types of genetic algorithms. Tests on a standard search problem and on a real-world design problem showed that GAs which tend to maintain population diversity can be used to locate many alternative acceptable solutions in a search and also are more reliable in locating the global optima. In particular, the Struggle GA variation of RTS showed superior performance in terms of reliability and number of converged local optima.

Future work includes the study of collaborative optimization, where different optimization modules have control on subsets of the search variables and adapt their search strategy on the basis of the information exchanged between optimizations.

## 6. Acknowledgements

We are grateful for the financial support provided by the National Science Foundation Center for Innovation in Product Development (grant number EEC-9529140) and the Alliance for Global Sustainability.

## Bibliography

Borup, L. and Parkinson, A. (1992). “Comparison of four non-derivative optimization methods on two problems containing heuristic and analytic knowledge.” Advances in Design Automation DE-Vol. 44-1: 137-143.

Brown, D. R. and Hwang, K.-Y. (1993). “Solving Fixed Configuration Problems with Genetic Search.” Research In Engineering Design 5(2): 80-87.

Budaychevsky, I. A., Chertakov, S. S. et al. (1993). “Using a constraint-oriented knowledge-based system for optimum mechanical engineering design.” Artificial Intellingence in Engineering.

Carlson, S. E. (1996). “Genetic Algorithm Attributes for Component Selection.” Research in Engineering Design 8(1): 33-51.

Carlson, S. E. and Pegg, T. A. (1995). "Genetic Operators for Catalog Design". Proceedings of the Design Automation Conference, ASME.

Chan, K. K. Y. (1997). An investigation on speciating genetic algorithms in multimodal optimization. Bachelor Thesis, Mechanical Engineering. Cambridge, MA, Massachusetts Institute of Technology.

Chen, R. and Ward, A. C. (1992)."Introduction to Interval Matrices in Design". Proceedings of the Design Theory and Methodology Conference, ASME.

DeJong, K. A. (1995). "An analysis of the behavior of a class of genetic adaptive systems". Dissertation Abstracts International 36(10), 5140B; UMI 76-9381. Ann Arbor, MI, University of Michigan.

Eshelman, L. J. and Schaffer, J. D. (1992). "Real-Coded Genetic Algorithms and Interval-Schemata". Foundations of Genetic Algorithms 2. L. D. Whitly. San Mateo, Kaufmann Publishers: 187-202.

Goldberg, D. E. and Richardson, J. (1987). "Genetic Algorithms with sharing for multimodal function optimization". Proceedings of the Second International Conference on Genetic Algorithms, Hillsdale, NJ, Lawrence Erlbaum Associates.

Grueninger, T. and Wallace, D. R. (1996). Multimodal optimization using genetic algorithms. CADlab report 96-02, http://cadlab.mit.edu.

Harik, G. (1995). "Finding Multimodal Solutions Using Restricted Tournament Selection.". Proceedings of the Sixth International Conference on Genetic Algorithms, Morgan Kaufmann Publishers.

Keeney, R. L. and Raiffa, H. (1976). Decisions with Multiple Objectives: Preferences and Value Tradeoffs. New York, NY, John Wiley and Sons Inc.

Kim, J. B. and Wallace, D. R. (1997). "A Goal-oriented Design Evaluation Model". Design Engineering Technical Conference, DETC97/DTM-3878, ASME.

Lin, B.-T. and Ward, A. C. (1992). “Introduction to design inferences over sets of intervals.” ASME Design Theory and Methodology 42: 187-192.

Mahfoud, S. (1995). Niching Methods for Genetic Algorithms, University of Illinois at Urbana-Champaign, IlliGAL Report 95001.

Pahng, K. F., Senin, N. et al. (1998). “Distributed modeling and evaluation of product design problems.” Computer-Aided Design 6(30): 411-423.

Reddy, S. Y. and K. W. Fertig (1996). "Design Sheet: A System for Exploring Design Space, Application to Automotive Drive Train Analysis". Fourth International Conference on Artificial Intelligence in Design (AID'96), Standford Univeristy, CA.

Ward, A. C. and Seering, W. P. (1993). “The Performance of a Mechanical Design 'Compiler'.” Journal of Mechanical Design 115(3): 341-345.

Ward, A. C. and Seering, W. P. (1993). “Quantitative Inference in a Mechanical Design Compiler.” Journal of Mechanical Design 115(1): 29-35.

## Appendix A Object model representation of the Shekels Foxholes problem

| Module name | Module type | Services Provided | Services Required | Comments | Optimization Setup |
|---|---|---|---|---|---|
| Foxholes | Container | | | Contains the foxholes relation and all other modules in the problem | |
| X | Real | Value | | Independent variable | Optimize |
| Y | Real | Value | | Independent variable | Optimize |
| Z | Real | Value | X.Value<br>Y.Value | Z=foxholes(X,Y) | |
| P | Preference function | Preference function values | | Designer's preference structure on Z (requirement) | |
| D | Criterion (single attribute decision) | Zacceptability | Z.Value<br>Preference function values | Assessment based on acceptability decision model | Objective function |

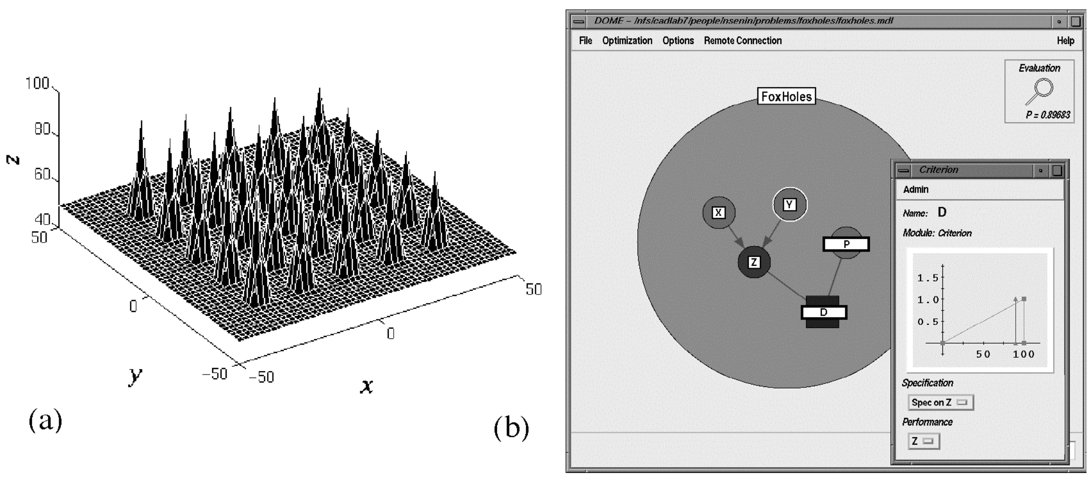



**Figure A1**: a) Function visualization, b) Object model visualization in DOME GUI