Optimization of Constructive Solid Geometry Via a Tree-Based Multi-objective Genetic Algorithm

Karim Hamza[1] and Kazuhiro Saitou[2*]

[1] Ph.D. Candidate, [2] Associate Professor, Mechanical Engineering Department
University of Michigan, Ann Arbor, MI 48109-2102, USA
{khamza, kazu}@.umich.edu

Abstract. This paper presents the multi-objective evolutionary optimization of three-dimensional geometry represented via constructive solid geometry (CSG), a binary tree of boolean operations of solid primitives. NSGA-II is extended for binary tree chromosomes with customized crossover and mutation operators tailored for the evolution of CSG trees and applied for two-objective shape optimization of indoor modular space truss joints. The results show success in generating a variety of shapes over the Pareto front. A selection of Pareto-optimal shapes are manufactured using a solid freeform fabrication process.

1 Introduction

Shape optimization of three-dimensional solid bodies is a challenging task that finds many applications across a multitude of disciplines such as machine design, structures, micro electro-mechanical systems, fluid mechanics and aerospace. Automated shape optimization of three-dimensional geometry often requires the computer representation of the solid geometry [1,2], such as i) voxel representation, ii) octree representation, iii) constructive solid geometry (CSG) and iv) boundary representation (B-rep). In voxel representations [1,2], a solid is represented as a collection of small volumetric cells in uniform sizes. The octree representation [1] is a structured variant of voxel representation based on a hierarchical subdivision, which allows the shape representation with locally varying details.

By far, the family of shape optimization methods, which appear in the literature most often, is known as topology optimization [3,4]. Topology optimization relies mostly on voxel representation and sometimes on the octree. A possible reason for the popularity could be the ease of encoding and interpreting the design variables in terms of material/no material decisions. Many algorithms have been applied for topology optimization, including gradient based, evolutionary, stochastic and hybrid techniques. However, the fact remains that the final output of topology optimization

* Corresponding Author

K. Deb et al. (Eds.): GECCO 2004, LNCS 3103, pp. 981–992, 2004.

is a rough grid of material/no material decisions. Such output must thus be *realized* into a shape that can be manufactured; otherwise the result will not be useful.

Constructive solid geometry (CSG) [1,2] is a rather unique representation of solid body geometry, in the sense that its building blocks are solid primitives such as a cube, cylinder or a sphere. Shapes of increasing complexity are *constructed* from the primitives via merging or subtracting the primitives or other CSG solids. As such, most solid bodies represented by CSG are relatively easy to manufacture because all the surface features have simple geometry. CSG representation has been used for direct optimization of shape [5] as well as a post processor for realizing the complex geometry resulted from topology optimization [6].

Boundary representation (B-rep) [1,2,7] represent the solid geometry as the intersection of half spaces represented by the surfaces of the solid. B-rep is used as the internal kernel for most commercial computer aided design (CAD) software as it offers the best flexibility and speed for interactive user editing of the shape. However, the boundary information storage system is often too complex to allow for shape optimization. To the best of the authors' knowledge, there have been no successful shape optimization studies via direct manipulation of B-rep.

Although the CSG representation has several advantages in terms of feature simplicity compared to topology optimization, it seems to be less used for shape optimization in the literature due to the difficulties associated with its tree structure. This paper presents the multi-objective optimization of CSG trees using an extended NSGA-II [8,9]. The following sections present the mathematical formulation of the problem, the details of the extended NSGA-II, and a case study on the shape optimization of indoor modular space truss joints. The paper concludes with a summary and a further work.

2 Problem Formulation

2.1 Notations

A *solid body* is a closed subset S of the three dimensional space (Fig. 1).

A *solid primitive* is a solid body S_p uniquely defined by a few parameters. The solid primitives considered in this paper are: i) box, ii) cylinder and iii) sphere (Fig. 1). These solid primitives can be uniquely defined by defined an origin point, axes orientation and one to three sizing dimensions.

A *Boolean operation* is a mapping from two solid bodies to a solid body (Fig. 2). The boolean operations considered in this paper are: i) *union* (Fig. 2-a) represented as '+' and ii) *subtraction* (Fig. 2-b) represented as '-'.

Constructive solid geometry (CSG) tree is a binary tree T with solid primitives at terminal nodes and Boolean operations at non-terminal nodes. While a CSG T can uniquely represent a solid body $S(T)$, the inverse is not true: for example, a solid body in Fig. 2-c can be represented by either of the two CGS trees in Fig. 3.

n_P denotes the number of primitive solids (leaflet nodes) in the CSG tree.

n_B denotes the number of boolean operations solids (inner nodes) in the CSG tree. It should be noted that $n_B = n_P - 1$.

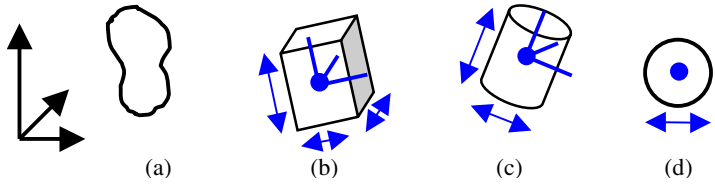

Fig. 1. (a) a general solid body is a closed subset of the three dimensional space. (b) a box can be uniquely defined by their origin location, orientation and three sizing dimensions. (c) a cylinder can be uniquely by its origin location, orientation and two sizing dimensions. (d) a sphere can be uniquely defined by its origin location and one sizing dimension.

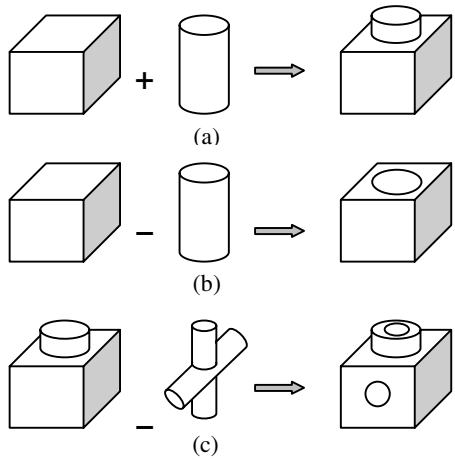

Fig. 2. (a) a box and a cylinder are combined via a union operation to form a padded box. (b) a box and a cylinder are combined via a subtraction operation to form a box with a hole. (c) Two solid bodies combine to form a more complex solid

2.2 Decision Variables

The decision variables are the information necessary to uniquely define a CSG tree: 1) number of primitive solids (n_p), 2) topology, 3) types of boolean operations at non-terminal nodes, and 4) types, origins, orientations and dimensions of solid primitives at terminal nodes.

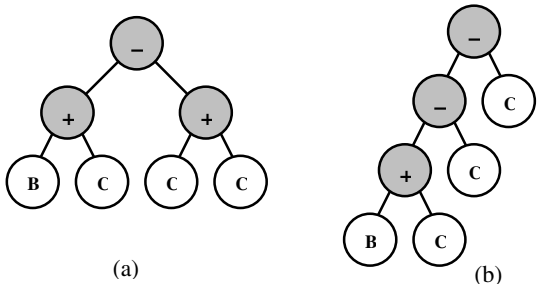

Fig. 3. Two CSG trees that can produce the solid body in Fig. 2-c. (a) the union of two cylinders are subtracted from the union of a box and a cylinder. (b) two cylinders are vertically and horizontally subtracted from the union of a box and a cylinder.

2.3 Objective Functions

Depending on the discipline that the shape optimization is intended for, there could be one or more objective functions. In the general case, the vector of objective functions can be written as:

$$\mathbf{f} = \langle f_1 \quad f_2 \quad \ldots \quad f_r \rangle \tag{1}$$

where $f_i = PerformanceMeasure_i(S(T))$, $i = 1$ to r

2.4 Constraints

Most applications require the final shape to be a single solid body. In other words, the solid represented by the CSG must be a *manifold* solid:

$$manifold(S(T)) = \text{true} \tag{2}$$

A manifold solid is a *connected set* in the three dimensional space. That is, any point within the solid body can be reached from any other point in the solid body, through some path that only traverses points within the solid body. Although making an exact check of the manifold condition for a general solid is a difficult task, there are algorithms that can quickly perform an approximate check with acceptable accuracy [7].

In terms of shape complexity, given the option for unlimited number of nodes in the CSG tree (combined with the sphere alone as a primitive), it is possible to get a near-zero representation of any solid shape in the three dimensional space. The proof to that is rather trivial, considering the limiting case of a sphere of a very small radius, the sphere approaches becoming a point. Given the option to do the union of unlimited number of points, it is possible to build-up any solid shape.

Practicality issues however, both in terms of computational resources as well as the manufacturability desire to decrease the geometric complexity (which is one of the

main drivers to use CSG, rather than other solid body representations), dictate that a cap should be placed on the maximum number of primitives to be considered in the CSG tree.

$$n_P \leq n_{P,\max} \tag{3}$$

The choice of $n_{P,\max}$ is up to the user and is essentially problem dependent. A large allowance on $n_{P,\max}$ would allow more flexibility in exploring more complex shapes. However excessively $n_{P,\max}$ is expensive in terms of computational resources and may have a negative effect of excessively increasing the size of the search space and thereby making the optimization task more difficult.

Aside from the manifold solid and maximum number of primitives, depending on the discipline of the problem, generic constraints can be incorporated into the problem. Generic constraints can have the functional form of:

$$feasible\,(S(T)) = \text{true} \tag{4}$$

3 Optimization Algorithm

3.1 Overall Flow

The overall flow of the optimization algorithm is similar to NSGA-II [8, 9] and is given as:

```
1.  Generate Initial Population
2.  Loop Until Termination: (Max. Number of Generations)
3.      Perform Pareto-Ranking of Current Population
4.      Preserve Elitism by passing copies of some top-rank members to
    new population
5.      Loop until New Population is full
6.          Select Parents for reproduction via binary tournament
7.          Perform Crossover and mutation then place the offspring in
            New Population
8.          When New Population is full, replace Current Population with
            the new one and repeat at Step #2.
```

Selection is performed via a binary tournament, in which two members are randomly selected from the current population. If one of the selected members has a better Pareto rank [9] than the other, the better member becomes a parent. If both members have the same rank, then one of them is randomly chosen to become a parent.

Elitism preservation is performed by passing copies of the top-ranked members to the new population prior to filling out the rest with the new offspring. If the number of top-ranked members in the current population is less than a threshold, their copies

are passed to the next generation. If there are more top rank members in the current population than a threshold, then only some of them are passed to the new population according to a nitching function that encourages the diversity of the Pareto front.

3.3 Chromosome Encoding and Evaluation

Encoding of the decision variables that define the CSG tree of a solid body in the linear format that is commonly used in multi-objective genetic algorithms seems inefficient. This is because the CSG is a variable-sized tree, which means that the length of the active portions of a linear chromosome are not guaranteed to be the same in two selected parents, which voids the use of many of the standard crossover techniques [10] for linearly encoded chromosomes. Furthermore, to adopt tree-structure crossover from a linearly encoded chromosome would mean continuous translation and de-translation between the chromosome and its equivalent tree, which is an unnecessary computational effort.

In this paper, the information of the decision variables is directly encoded as tree structure chromosome, which represents the CSG tree. This encoding scheme also has the advantage of allowing the chromosome is allowed to branch out and grow during the evolution of the CSG tree without a strict cap on the number of nodes or the topology of the tree.

Implementation of the CSG tree structure is performed via the C++ programming language, and is linked with the ACIS [7] solid modeling kernel, which facilitates translating the CSG tree into a format recognized by several computer aided design (CAD) software packages across several disciplines. The CAD packages could then report back the performance measures (problem specific objectives and constraints) of the solid shape.

3.4 Handling of Constraints

Keeping in mind that the evaluation of problem specific objectives and constraints using CAD packages might involve expensive computations such as finite element, dynamic and fluid mechanics simulations, the *constraints first – objectives later* approach [11] is adopted. Constraints are hierarchically listed staring with the manifold solid constraint (equation 2) then the cap on number of primitives (equation 3). Violating designs are given a *death penalty*. That is, any design that is infeasible in the first constraint will be dominated by any design that is feasible on that constraint, and so forth for the list of constraints. This approach of handling constraints removes the need to perform the expensive link with the CAD software for obviously bad designs as well as protect against CAD software instability, which can often happen if presented with a non-manifold solid or a solid of extreme complexity (reflected by too big a CSG tree).

3.5 Seeding of the Initial Population

Seeding of the initial population is an option, which the algorithm user might elect to activate. It allows the input of particular design guesses or previously known good designs into the initial population. When combined with elitism, seeding guarantees that the final outcome of the search will be no worse than the best of the previously known seeds. However, seeding might sometimes be inappropriate (especially for difficult feasibility problems), because it can cause excessive bias of the initial population and thereby lead to premature convergence of the search without much exploration beyond the given seeds.

3.6 Crossover and Mutation

Crossover and mutation operators are specially designed and are based on physical understanding of the problem represented by evolving the shape of a solid body.

– *Exchange of sub-trees:* This is a well established crossover technique in genetic programming [12, 13] for tree structures that present computer programs. Applying it to CSG trees corresponds to two parent solids exchanging geometric features. However, when the exchange is too randomly disruptive if it occurs too close to the root node of the tree, since it roughly corresponds to replacing half the solid with some other geometry that may not necessarily fit the first half. Therefore this form of crossover is used with medium overall probability.
– *Randomly Flipping a Boolean Operation:* This is a heavily disruptive mutation, which is used with low probability to induce diversity in the population.
– *Complete Randomization of a Primitive Solid:* This is another disruptive mutation, which is used with low probability to induce diversity in the population.
– *Slight Randomization of Origin/Orientation of a Primitive Solid:* This is used as a means of local adjustment/sampling of the positioning of primitive solids.
– *Short Local Search along Origin/Orientation of all Primitive Solids in Tree:* The short runs of local search (not pursued until reaching a local optimum) serve as a means of improving quality of the population members. However, local search should not be excessively used because employing it within the GA loop can lead to premature convergence as presenting extra computational cost.

4 Application Case Study: Indoor Modular Space Truss Joints

4.1 Problem Description

A schematic diagram of modular space trusses is shown in Fig. 4. A main advantage of space trusses is there efficient and light weight designs, yet a disadvantage is the extra complexity during manufacturing and erection. Producing space trusses in a modular form allows for mass production, which significantly reduces the manufacturing and erection cost. Large scale outdoor trusses generally have little restrictions

on the size considerations of the joints, which allows for sophisticated joints made of welded-plates, which are strong and light weight to be used. The indoor version of the modular space trusses are designed for quick erection/removal. Their main use is for hanging banners and other light weight objects. For the indoor version of the modular trusses, size and manufacturability considerations often dictate the shape of the modular joints to be simply spherical; a shape which can be challenged.

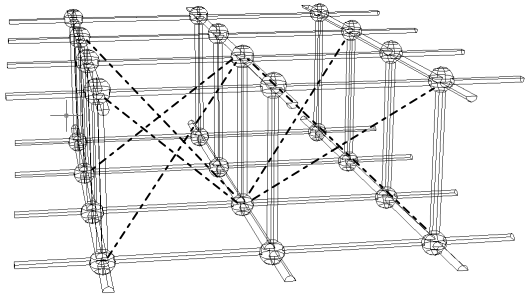

Fig. 4. A schematic diagram of a modular space truss (cross diagonal structural members removed from the diagram to allow better visibility)

The first objective is a measure of the strength of the joint. In this preliminary study, a fast rough calculation is sufficient. This is done by observing the material cross-section around load paths and using it as a measure of stiffness. The objective is to maximize the stiffness, but since the implemented algorithm is set for minimization, the first objective is given as:

$$\text{Min. } f_1 = -\sum_{1}^{\text{nLoadPathPoints}} \min(\mathit{MaterialDepth}(S(T))) \tag{5}$$

The second objective is minimization of weight or amount of material in the joint, which is given as:

$$\text{Min. } f_2 = \mathit{volume}\,(S(T)) \tag{6}$$

Aside from the manifold solid constraint and the cap on geometric complexity (maximum 20 primitive solids for this problem), the triple symmetry of the joint allows for reduction of the search space by limiting the search to one octant. The evolved geometry in the octant is then triple mirrored to produce a tri-symmetric joint. However to allow for this reduction, the evolved solid shape must be well connected to the three mirroring planes, or else the mirroring could produce a non-manifold solid. This is represented by the constraints:

$$\mathit{IntersectionArea}\,(S(T)) \geq 2\% \text{ of unit square, for XY, YZ and ZX planes} \tag{7}$$

It is important to note that the choice of objectives and constraints is highly application dependent. Although weight minimization is a common objective in many structural design problems, performance measures for strength are usually more difficult. In this particular study, knowledge about the function of the joint allowed for prior definition of the load paths. In general cases however, it would be necessary to link evolving geometry with automated meshing for finite element analysis (FEA) using CAD packages. For this study, evaluation of objectives typically takes less than two seconds on a 2.0GHz PC. Incorporating FEA would probably increase the computations ten-fold, thereby increasing typical time to perform one GA run from approximately one hour (as in this study) to about half a day, which is still quite feasible and could be pursued in future research.

4.2 Results and Discussion

Several runs of the implemented algorithm are performed for the truss joint. The main tuning parameters for the GA runs are listed in Table 1. A typical run produced the quasi-Pareto curve of Fig. 5. The main interest of this case study is observing how the shape of the joint changes along the trade-off between stiffness and weight. Therefore, the actual quantities of f_1 & f_2 are non-dimensionalized with respect to the highest and lowest found points. The CSG trees of the points in the Pareto-curve are exporting as Solid models via the ACIS library [7] and the rendered CAD models are displayed as small icons in Fig. 5. Furthermore, selections of the CAD models are input to a rapid freeform manufacturing machine. Photos of the actual manufactured models are displayed in Fig. 6.

Table 1. Tuning parameters of the GA runs for the indoor modular space truss joints case study

Population Size	100
Number of Generations	30
Threshold for number of elite members	20
Initial Population Seeding	i) 6 Seeds (solid cube, sphere, sphere with forking cylinders at 2 sizes), ii) Unseeded
Prob. Exchange of sub-trees	0.50
Prob. Flip Boolean Operation	0.20 / Number of nodes
Prob. Complete Randomize Primitive Solid	0.20 / Number of nodes
Prob. Local Positioning Shift	0.50 / Number of nodes
Prob. Short Local Search	i) 0.00, ii) 0.05

It is noted that the search space is large (even for the simple considered problem). A twenty-primitive solid tree would have 19 binary choices for the inner nodes (boolean operations), 20 discrete choices and approximately 20×6 continuous variables for the terminal nodes (primitive solids). It was observed in most runs however, that most of the higher fitness designs had few primitive solids providing material cover to the important zones in the truss joint. This observation permitted the use of relatively

small population size, while achieving consistent performance in most of the performed runs.

Other than the sample run used for manufacturing of the models, a total of twenty runs were performed, each five of which using different parameter settings in terms of population seeding and probability of performing local search as stated in Table 1. The top rank population members of each of those runs are collectively plotted in Fig. 7. It is observed in Fig. 7 that none of the runs with unseeded initial populations was able to reach the highest 10% joint stiffness region of Pareto-front. On the other hand, most of the seeded runs had several members in that region. This is understandably the effect of seeding, since one of the seeds is just a solid cube (filling all the search shape search space with material), which is obviously the Pareto-point that has the highest possible stiffness, yet heavy weight. In the low to mid level joint stiffness region however, the unseeded runs seem to be working better. This might indicate that seeding can be causing excessive bias to the higher stiffness region of the Pareto-front.

Although implementation of local search seems to make intuitive sense since a good proportion of the design variables are continuous, the effect of local search is hardly observable in the high and low joint stiffness regions. In fact, in the mid level region, it seems to have a negative effect. This could probably be attributed to premature convergence effects.

Overall, most of the runs performed well, which indicates consistency of success of the algorithm for the considered case study.

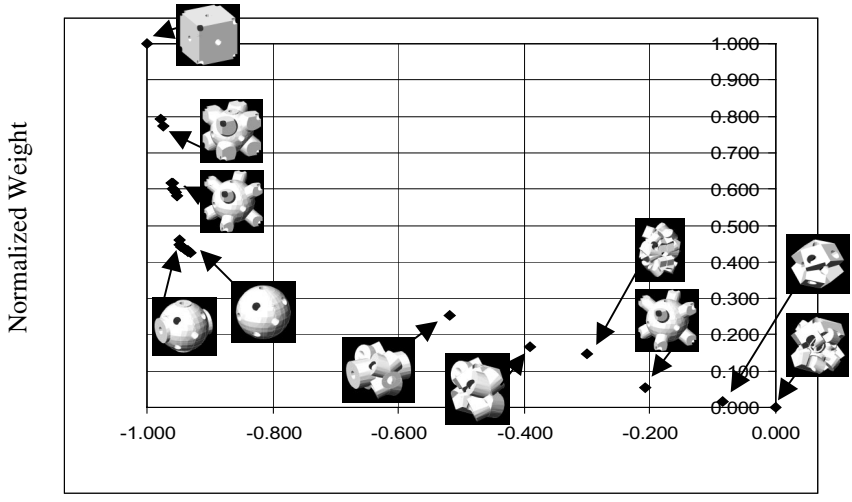

Fig. 5. Results of a typical run of the implemented algorithm showing the normalized trade-off between maximizing stiffness and minimizing joint weight. Also shown are CAD models of the first rank evolved CSG trees

Fig. 6. Photos of selected quasi-Pareto optimal shapes that were rapidly manufactured using selective laser sintering

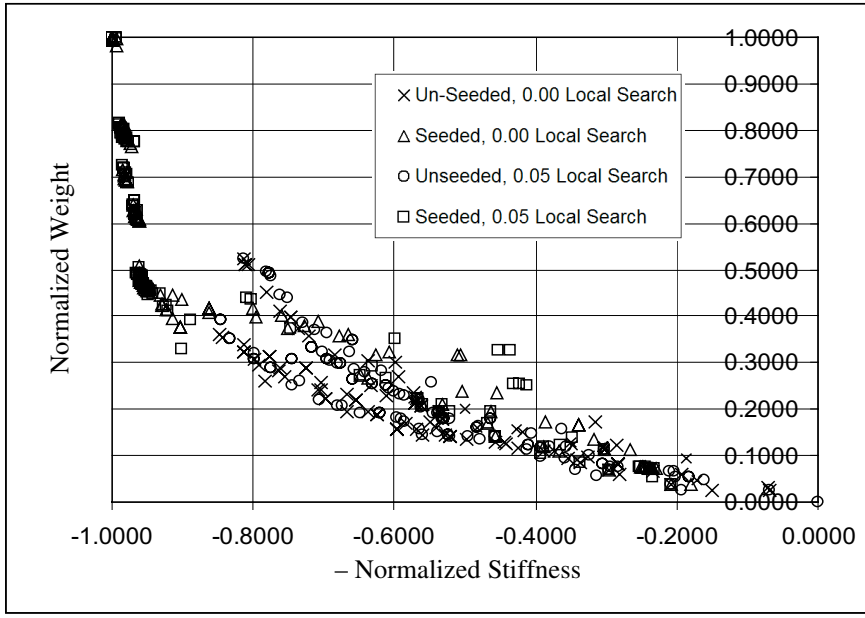

Fig. 7. Summary of top rank population members for twenty GA runs

5 Conclusion and Further Work

This paper presented an adaptation of NSGA-II for shape optimization of three dimensional solids by evolving CSG trees. The algorithm is set to directly evolve the tree structure by applying specially tailored crossover and mutation operators to a tree structure encoded chromosome. A simple case study involving shape optimization of indoor modular space truss joints is presented to demonstrate the algorithm. Models of selected quasi-Pareto optimal joint shapes were then manufactured via a rapid solid freeform fabrication technique.

Further work would include case studies of more complex components, possibly across several engineering disciplines. Also, more experimentation with the tuning parameters of the algorithm to determine the effects on the search efficiency is to be pursued.

Acknowledgements. The authors would like to extend special thanks to Prof. Suman Das from the University of Michigan, Ann Arbor. The manufacturing of the models of the modular truss joints was performed in the Solid Freeform Fabrication Lab as part of a course project on Solid Freeform Fabrication taught at the University of Michigan during the Fall semester of 2003.

References

1. Maentylea, M.: An Introduction to Solid Modeling. Computer Science Press, Inc., Rockville, Maryland (1988)
2. Lee, K.: Principles of CAD/CAM/CAE Systems. Addison Wesley, Reading, Massachusetts (1999)
3. Chapman, C., Saitou, K., and Jakiela, M., "Genetic Algorithms as an Approach to Configuration and Topology Design," Transactions of ASME, Journal of Mechanical Design, v. 116 (1994) 1005–1012
4. Bendsϕe, M. and Kikuchi, N. Generating optimal topologies in structural design using a homogenization method, Computer Methods in Applied Mechanics and Engineering, 71 (1998) 197-224
5. Kodiyalam, S., Kumar, V., Finnigan, P.: Constructive solid geometry approach to three-dimensional structural shape optimization. AIAA, 30 (1992) 1408–1415
6. Chirehdast, M., Papalambros, P.: Conversion of spatial-enumeration scheme into constructive solid geometry. Computer Aided Design, 26 (1994) 302–314
7. Corney, J, Lim, T.: 3D Modeling with ACIS. Saxe-Coburg Publications, Stirling, UK (2001)
8. Deb, K., Argawal, S., Pratab, A., Meyarivan, T.: A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II. Proceedings of the Parallel Problem Solving from Nature VI Conference, Paris, France (2000) 849-858
9. Coello, C., Van Veldhuizen, D., Lamont, G.: Evolutionary Algorithms for Solving Multi-Objective Problems. Kluwer Academic Publishers (2002)
10. Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs. 3rd edn. Springer-Verlag, Berlin Heidelberg New York (1996)
11. Kurpati, A., Azarm, S., Wu, J.: Constraint handling improvements for multiobjective genetic algorithms. Struct Multidisc Optim 23 (2002) 204–213
12. Koza, J.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press (1992)
13. Michalewicz, Z.: How to Solve it: Modern Heuristics. Springer-Verlag, Berlin Heidelberg New York (1999)