

DETC2005-85111(DAC)

SYNTHESIS OF STRUCTURAL SYMMETRY DRIVEN BY COST SAVINGS

Tomonori Honda*

Engineering Design Research Laboratory
California Institute of Technology
Pasadena, CA 91125
email: thonda@caltech.edu

Fabien Nicaise

Engineering Design Research Laboratory
California Institute of Technology
Pasadena, CA 91125
email: fabien@caltech.edu

Erik K. Antonsson, Ph.D., P.E.

Engineering Design Research Laboratory
California Institute of Technology
Pasadena, CA 91125
email: erik@design.caltech.edu

ABSTRACT

An engineer presented with a design challenge often creates a symmetric solution. For instance, consider a table (front-back and left-right symmetry), a car (left and right symmetry), a bridge (front-back and left-right symmetry), or the space shuttle (left-right) symmetry. These examples may not be 100% symmetric, but their overriding features are remarkably similar. The reasons for the design of symmetric structures is not always clear. In some cases, like the table, symmetry may be a tradition. Similarly, the symmetry may be for aesthetic reasons.

However in automated design algorithms, especially stochastic techniques, the output is often largely asymmetric. One reason for this is that fitness functions are not rewarded for symmetry. A possible resolution to this is to add a reward function for symmetry. Unfortunately, this approach is computationally intractable as well as arbitrary.

In this paper a Genetic Algorithm based method is pre-

sented that rewards re-use of parts. The method is applied to a simple, idealized situation as well as to real design case. The results show that in some situations, symmetry naturally emerges from the synthesis, but that it does not provide clear performance advantages over asymmetric configurations.

1 Introduction

The aim of this paper is to demonstrate the possibility of (at least partially) automating the procedure of designing complex mechanical structures, and to demonstrate the possibility of creating symmetry. An example motivation for developing this capability is the need for a procedure that automates the design of the truss structure for the Descent Stage of NASA's Mars Science Laboratory (Mars 2009), schematically illustrated in Figure 1. Although the project is in the early stages of design, current solutions for this truss structure include 10 truss elements joining together at a node, making it difficult and expensive to build.

*Corresponding author.

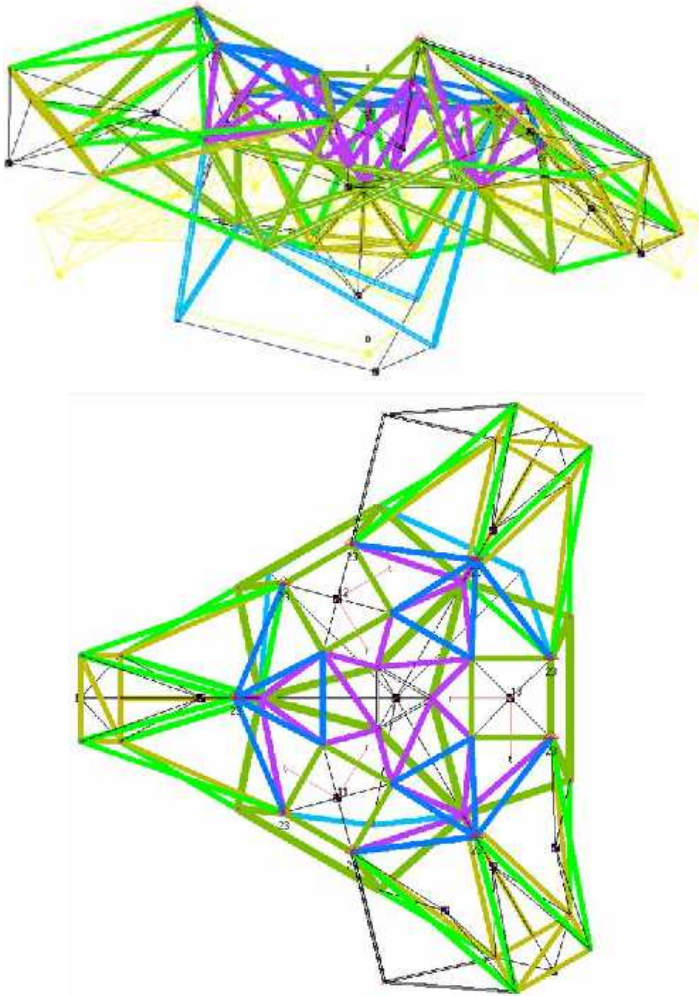


Figure 1. Two views of a Sample Descent Stage Truss Structure.
 Figures courtesy of NASA's JPL

Thus, the automation of this design task must consist of *topology synthesis* as well as optimization of node locations and cross-sectional areas. In addition, there is a need to be able to generate valid solutions to this design challenge as requirements change.

The use of stochastic search techniques to determine optimal solutions to design tasks has become routine. The design problem can be formulated into an optimization problem by using a utility or performance function to quantify performance associated with each particular design. For example, the Method of Imprecision (M_{OI}) [1] uses aggregated preference to represent preference for overall performance. In this paper, a technique is presented and demonstrated in which a Genetic Algorithm (GA) is used

for the synthesis of a truss structure that establishes a balance between mass, stiffness, and stress while avoiding the geometric constraints in form of “keep out” zones. (The “keep out” zones create more local optima in the fitness landscape, requiring an increase in the exploration needed.)

The main difference between designs generated manually and by computer is the use of symmetry. In general, human designers exploit symmetry as much as possible, while computer generated designs using stochastic search tend to lack symmetry. What value do symmetric structures have that stochastic search overlooks? One benefit of the having symmetry in structures is cost savings through repeated elements. Thus, the possibility of encouraging symmetry in stochastic truss synthesis through cost savings as additional parameter is explored here.

2 Background

Since the 1960's, there has been a developing interest in mimicking natural evolution to solve optimization problems. This has led to the development of Evolutionary Computation (EC) by Goldberg [2] which later developed into Genetic Algorithms (GA), Genetic Programming (GP), and Evolutionary Strategies (ES) [3–6]. The strength of these evolutionary techniques is that they can identify global optima, compared to more traditional convex optimization that searches for local optima. This distinction makes evolutionary algorithms less dependent on the initialization, compared to other optimization techniques.

Evolutionary techniques have been previously applied to structural configuration and optimization problems, however, the results are frequently highly asymmetric structures. One of the goals of this work is to synthesize symmetric structures by including a benefit for symmetry in the fitness landscape, rather than seeding (initializing) the evolution with symmetric structures.

The central idea behind Genetic Algorithms is to have a population of possible candidates for the solution and evolve this population by stochastic processes to make the population “better”. (An overview of a Genetic Algorithm can be seen in Figure 2.) First, using the knowledge of the design space and goals, the designer creates a gene (binary or real array) to represent or encode locations in the design space. (This paper assumes that real number encoding is used to describe the design space.) A particular position in the design space is called an individual. Offspring are cre-

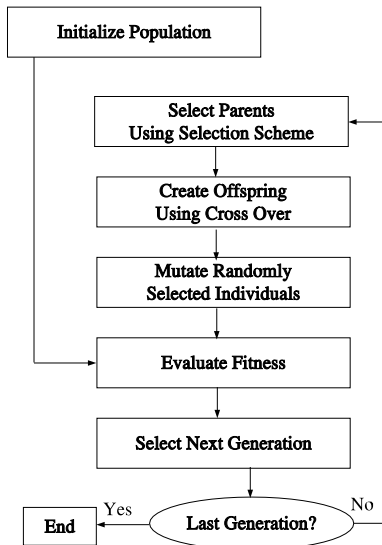


Figure 2. Basic Traditional Genetic Algorithm Loop.

ated from pairs of individuals using a cross-over operator that combines two genes to create another gene. In order to compare the individuals, the fitness function is constructed that represents the relative difference in preference based on the predicted performance of the individual. In addition, spontaneous mutation of a gene is programmed to occur. Finally, the next generation of the population is determined from the pool of all the individuals existing using a “survival of fittest” (“generation selection”) scheme.

The basic GA as a synthesizer has the following characteristics: cross-over mainly provides the exploitation of the “good” neighborhood, where the fitness is known to be high; mutation provides a means to explore a new neighborhood. After many generations of evolution of the population, individuals will tend to cluster around one or more fitness peak(s) (maxima).

Many papers have been published in the area of optimization of truss structures using GA’s. However, most of these optimizations focus on optimizing cross-sectional area of each truss element (*e.g.*, [7]). There are a few prior publications focusing on the configuration of truss structures that allow locations of the joints to move (*e.g.*, [8]). Finally, there are a small number of publications that address optimization of topology (number of joints and members) of the truss structure [9–14]. Typical methods that allow for topological variation, initially input a maximum

number of truss joints, and allow genes to turn off unconstrained joints. Thus, current truss optimization using GA’s are limited to searching for truss structures limited to a maximum number of joints prescribed in advance. Similar study on beam manufacturing cost and assembly cost using this method is recently published by K. Saitou and N. Lyu [15]. Because these methods assume that the designers have previously developed some particular topological configurations (could be more than one), it makes this a process more of optimization than synthesis of new designs. In this paper, the GA is modified so that it searches through variable topological truss space efficiently as described in next section. These topological changes are needed to allow the GA to produce different symmetric and nonsymmetric truss structures, given the locations of base nodes and loads.

3 Method

To implement a modified Genetic Algorithm to synthesize truss structures, three major elements must be defined: a Genetic structure (encoding), the fitness function/landscape, and optimization algorithms using these genetics.

3.1 Encoding

Each truss is encoded using two arrays, one for the node locations and the other for connectivity properties. Each node location consists of a 3-D vector of real numbers representing the Cartesian coordinates of the node. The node identification number for each node is implicitly kept by the index. The connectivity array keeps data on the links connecting the nodes: the two end node numbers, cross-sectional area, and the material number. (There is separate, fixed data structure that stores the material properties for an assortment of different materials.) The quantity of nodes and links is allowed to vary from individual to individual and is not limited. (The number of nodes is controlled using survival of fittest, *i.e.*, it will stabilize in the GA run to an optimal number of nodes.)

In addition, there are several data structures that store information common to all individuals. One of these is the list of candidate materials already mentioned. Another array stores the constraints imposed on each node as well as imposed loads. Finally, there is an array to track all the keep-out zones imposed by the problem statement. The input file for the code includes these prescribed nodes, con-

straints, force vectors, material properties, and keep-out zones.

3.2 Fitness Landscaping

The principal concern regarding the fitness landscape of a Genetic Algorithm is providing an incentive (selection pressure) to converge to at least one set of feasible solutions. A feasible solution to the design problem here is any rigid truss that satisfies geometric constraints (in form of keep-out zones) and supports given loads. To achieve this fitness landscape, the fitness level is divided into four regions:

1. Any truss that is not self-supporting.
2. Any self-supporting truss, but does not support all the requisite loads
3. Any self-supporting truss, which supports the load but violates the keep-out constraints.
4. Any truss that meets all the necessary criteria.

As the fitness of the truss improves, and ascends through the levels, the truss becomes closer to a feasible truss. By separating the fitness into four levels, there is a selection pressure at all points of the fitness landscape.

If truss falls in the first level, it is given a fitness based on how close it is to becoming a structure:

$$\text{fitness} = 0.25 * \frac{\# \text{ of dof. removed}}{\text{total \# of dof.}} \quad (1)$$

where dof. is degree of freedom. Then, if the truss achieves a perfect score at this measure, it moves on to level 2 of the evaluation. At this stage, it is checked to determine whether it passes through all of the prescribed load nodes. If it does not, then the fitness for the truss will be given by:

$$\text{fitness} = 0.25 + 0.25 * \frac{\# \text{ of load supportable}}{\text{total \# of loads}} \quad (2)$$

Then, when a truss carries all of the requisite loads, it is checked to determine whether the geometric constraints, in form of keep-out zones, are satisfied. The fitness score is then based on the number of violations:

$$\text{fitness} = 0.5 + 0.25 * \frac{\# \text{ of link violating KO Zone}}{\text{total \# of links}} \quad (3)$$

Once a truss has satisfied these constraints, the performance, such as natural frequency and factor of safety of the truss, will be evaluated. The reason these are held until last is that they tend to be computationally intensive tasks, and the first three levels ensure that only useful trusses are simulated.

Finally, in the last level fitness is measured with a shifted aggregated preference defined in the Method of Imprecision [1, 16] rather than the strict measures above. The shift compensates for the fact that the fitness is measured in the range of [0.75, 1.0] since a truss at this stage of fitness evaluation is at least a rigid, load-supporting truss satisfying the geometric constraints.

The fitness for the truss is given by:

$$\text{fitness} = 0.75 + 0.25 * \mu_{agg}(\text{truss}) \quad (4)$$

where μ_{agg} is the aggregated preference.

Several performance variables are calculated for each truss: cost, mass, lowest vibrational frequency, and factor of safety for stress. The most important performance variable implemented is the manufacturing cost for truss. One of the benefits of having symmetric parts is the cost saving by use of repeated elements, (*i.e.*, repeated production of the same part is usually cheaper than production of different parts). Theoretically, this could induce symmetry in the truss. The other criteria are also commonly used by JPL when evaluating truss design. To be consistent with JPL's standard, MSC Nastran is used to calculate the lowest vibrational frequency and factor of safety. An example of the particular preferences and aggregation of preferences is shown in the next section.

3.3 Algorithm Design

It is well known that the convergence of Genetic Algorithms depends on the smoothness of the fitness landscape. Because of choice of the encoding, a small change in end nodes values of the connectivity array could lead to drastic changes in the fitness value. This leads to poor convergence unless some measure is taken to improve the convergence. Thus, the following three-loop approach has been developed [17], in which each subloop will have a better possibility of convergence than brute force optimization over the global fitness landscape. (See Figure 2.) The inner most loop only optimizes over the cross-sectional area

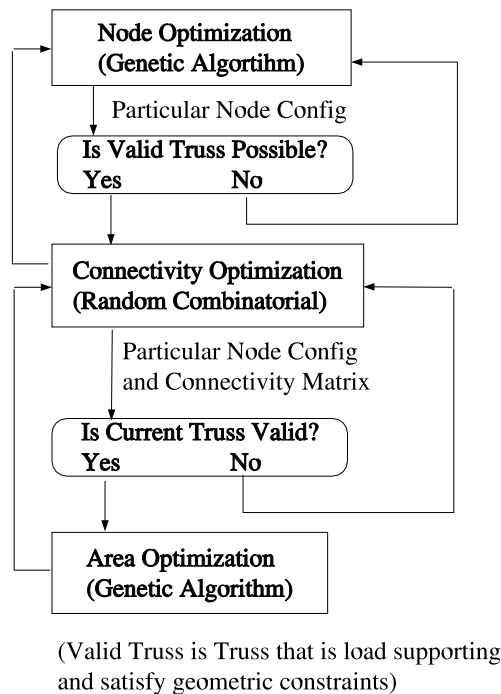


Figure 3. Truss Optimization Flow Diagram

of the truss elements if the truss satisfies the minimum requirement, (*i.e.*, fitness is at least 0.75.) The middle loop optimizes the connectivity of truss. The outer loop optimizes over the node locations. Thus, the smoothing of the fitness landscape is intended for design regions where the truss is at least rigid, load-supporting, and satisfies the geometric constraints. (See Figure 3.)

At the inner loop stage, the objective (fitness) function is a function of cross-sectional area only. As long as the preference function is continuous with respect to the performance variables, this objective function will be a continuous function of area. To optimize this area, a simple Genetic Algorithm is used with the following mutation and crossovers. Area mutation is achieved by choosing new areas from a Gaussian distribution centered around the current area with a standard deviation of 20% of the current area. Cross-sectional area cross-over is achieved by randomly selecting the area for the child link from one of the two parents. The population size for area optimization is usually 5 to 10, and typically iterates for 4 to 8 generations. The population and generation size is small because

the main function of this inner loop is to reduce the chance of eliminating a truss because of poor area choices. Also, when the outer loops start converging, this inner loop will be less necessary.

At the middle loop stage, pure random combinatorial optimization is used to optimize the connectivity matrix. However, before the connectivity is optimized, a new truss with all links not in violation of the keep-out zones is created. This truss is used to determine if, given the nodal configuration, it is possible to create a load-supporting truss that satisfies the geometric constraints. If it is not possible, then the connectivity optimization is skipped for that particular truss. If it is possible, then several links are randomly turned on or off to create new trusses. The best one is chosen from 5-10 trusses generated.

In the outer loop, node locations are optimized. If the two inner loops synthesize good-quality alternative configurations, then the fitness landscape should be continuous with respect to the nodal coordinates. To optimize the node locations, a Genetic Algorithm with following mutation and crossovers is used. Node mutation first randomly selects how many nodes to add or subtract using a discretized normal distribution with mean of 0 and standard deviation of 1. When a node is to be removed, it is randomly selected from the free nodes (nodes that are neither base nodes nor load nodes.) When a new node is to be added, it is randomly chosen from the entire space. Finally, all nodes are allowed to shift location with probabilities obtained from a Gaussian distribution with appropriate parameters. Crossover is performed by random selection of free nodes from each of the two parents.

4 Examples

4.1 Example 1

As a first example, a truss is synthesized with the following inputs. The base nodes are located at $\{(0,0,0), (10,0,10), (-10,0,10)\}$ and load nodes are located at $\{(0,9.5,-0.5), (0.5,9.5,0.5), (-0.5,9.5,0.5), (0,10,0)\}$. The pyramid defined by the load nodes is input as a rigid body. The force vector, $(0,-1e5,0)$, is applied at the node located at $(0,10,0)$. There are two independent geometric constraints. The first is that any part of the truss cannot pass

through the intersection of two solid cones defined by:

$$\{(x_1, x_2, x_3) \in R^3 | (x_2 - 9)^2 \leq \frac{16}{9}(x_1^2 + x_3^2) \leq (x_2 - 1)^2\} \quad (5)$$

The other constraint consists of a larger cone, and is considered to be a keep-in zone, and is defined by:

$$\{(x_1, x_2, x_3) \in R^3 | (x_2 - 17)^2 \leq \frac{9}{16}(x_1^2 + x_3^2)\} \quad (6)$$

Figure 4 shows the starting configuration for truss synthesis.

To implement truss synthesis, the preference for each performance variable must be defined. The preference for mass is given by:

$$\mu_m(m) = \begin{cases} 1 - \frac{\sqrt{m}}{\sqrt{M}} & \text{for } m < M \\ 0 & \text{for } m > M \end{cases} \quad (7)$$

where μ_m is the preference function for mass, m is the mass of the truss being evaluated, and M is maximum mass cut-off. For the first example $M = 80$ is used. The preference for natural frequency has the form:

$$\mu_f(f) = \begin{cases} 0 & \text{for } f < f_{\min} \\ \frac{\log_{10}(f) - \log_{10}(f_{\min})}{\log_{10}(f_{\text{des}}) - \log_{10}(f_{\min})} & \text{for } f_{\min} < f < f_{\text{des}} \\ 1 & \text{for } f > f_{\text{des}} \end{cases} \quad (8)$$

where μ_f is the preference function for natural frequency, f is lowest frequency mode of the truss, f_{\min} is minimum requirement for frequency mode, and f_{des} is desired lowest frequency mode. In this example, f_{\min} is set to 10^{-2} and f_{des} is set to 10^3 . Also, the preference for stress is defined using factor of safety and has the similar form:

$$\mu_s(S_f) = \begin{cases} 0 & \text{for } S_f < 1 \\ \frac{\log_{10} S_f}{\log_{10} S_d} & \text{for } 1 < S_f < S_d \\ 1 & \text{for } S_f > S_d \end{cases} \quad (9)$$

where μ_s is the preference function for stress, S_f is factor of safety of the truss, and S_d is the desired factor of safety.

S_d is set to be 10 for this example. (MSC Nastran is used to calculate the natural frequency and factor of safety.)

The final preference defined is for cost. This preference function uses the concept that it is cheaper to produce repeated links than nonduplicate links. Thus, any repeated links will be discounted. (For simplicity, the cost of joints are ignored, but cost of joint proportional to complexity could be added.) Each dissimilar link will cost 1 monetary unit to produce. As a discount, for a set of repeated links within 5% of each others dimensions, the links will cost 0.5 units, except for first one which will still cost 1 unit to produce. The total cost for the truss is calculated by following equation:

$$\mu_c(C) = \begin{cases} 1 - 0.5 * \left(\frac{C}{10}\right)^2 & \text{for } C \leq 10 \\ \frac{9}{8} * \left(\frac{C}{40} - 1\right)^2 & \text{for } 10 < C \leq 40 \\ 0 & \text{for } C > 40 \end{cases} \quad (10)$$

where μ_c is the preference function for cost and C is the total cost for all of the links of the truss.

Finally, these preferences must be aggregated to final preference using formula developed in [16]:

$$\mu_{agg} = \left(\frac{w_m \mu_m^s + w_f \mu_f^s + w_s \mu_s^s + w_c \mu_c^s}{w_m + w_f + w_s + w_c} \right)^{1/s} \quad (11)$$

where μ_{agg} is the aggregated preference, w are the respective weights, and s is the degree of compensation [16]. Variations in these parameters leads to different Pareto Optimal solutions. For this particular example, $w_f = 0.1$, $w_c = 1$, $w_s = 1$, $w_m = 1$, and $s = -1.0$ are chosen.

For each run, the following population size and iteration are used for each loop. For the cross-sectional area loop, a population size of 12 and 5 iterations are used. For the connectivity loop, a population size of 12 and 5 iterations are used. For the node loop, a population size of 40 and 100 iterations are used.

As shown in the figures (5 and 6), the results of the truss synthesis can be categorized into symmetric and non-symmetric results. Because the Genetic Algorithm operates stochastically, the best symmetric and non-symmetric solutions from 15 different runs are used to reach conclusions. The best symmetric solution has a fitness of 0.935, and the best non-symmetric solution has a fitness of 0.947.

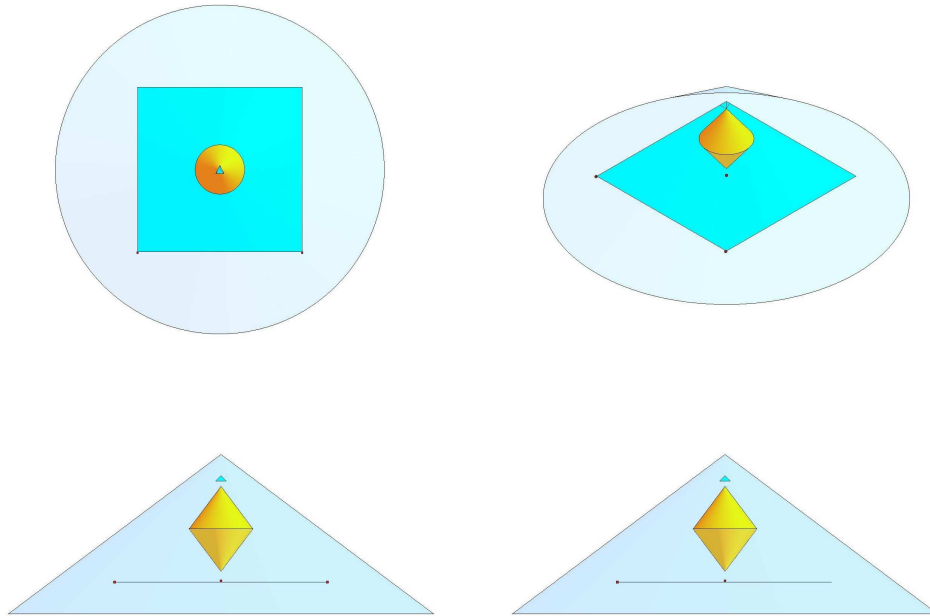


Figure 4. Basic set-up of the first test case, shown in top, front, side and isometric views (counterclockwise from the upper left).

On average, symmetric solutions have an average fitness of 0.920, and non-symmetric solutions have an average fitness of 0.938. This implies that even with cost savings for repeated elements, non-symmetric trusses have better preference (and hence, better performance) than symmetric trusses. The main causes are that symmetric trusses have lower (and hence, less desirable) vibrational frequency than non-symmetric trusses. Also non-symmetric trusses can use repeated links in a non-symmetric manner.

4.2 Example 2

The second example is a more realistic, albeit still idealized design challenge. It is the design of the descent stage for the Mars Science Laboratory. The purpose of the Descent Stage is to cradle the rover from launch, through entry, and safely deposit the rover on the surface of the planet. Once it delivers the rover to the surface, the Descent Stage needs to then crash to the surface in an area that will prevent damage to the rover.

For this example, the Descent Stage design is taken purely as a mechanical engineering, structural design task. All of the thrusters, propellant tanks, and interface hard-

ware are assumed fixed. (A future goal of the task is to add is some configurational freedom.) Figure 7 shows the overall configuration and structure for the Descent Stage. There are several portions to the design:

- o The entire design is housed in an aeroshell to protect it during entry into the Mars atmosphere. In this simplified version, it is represented by two 45° cones. These have not been rounded or sculpted as the real aeroshell certainly would be.
- o Near the top of one cone is the Interface to the cruise stage and the launch vehicle. This is the main load carrying member for the entire vehicle. It is equivalent to the groundplane in Example 1 above. For the Descent Stage it is modeled as thin torus near the back of one cone.
- o Near the bottom of the opposite cone is payload delivered to the surface, in this case a rover. For the purpose of this exercise the rover is shown smaller than real life to ensure a challenging solution space. The rover is represented by a large box.
- o Along the intersection line of the two cones lie the thrusters. They are placed at the widest diameter to

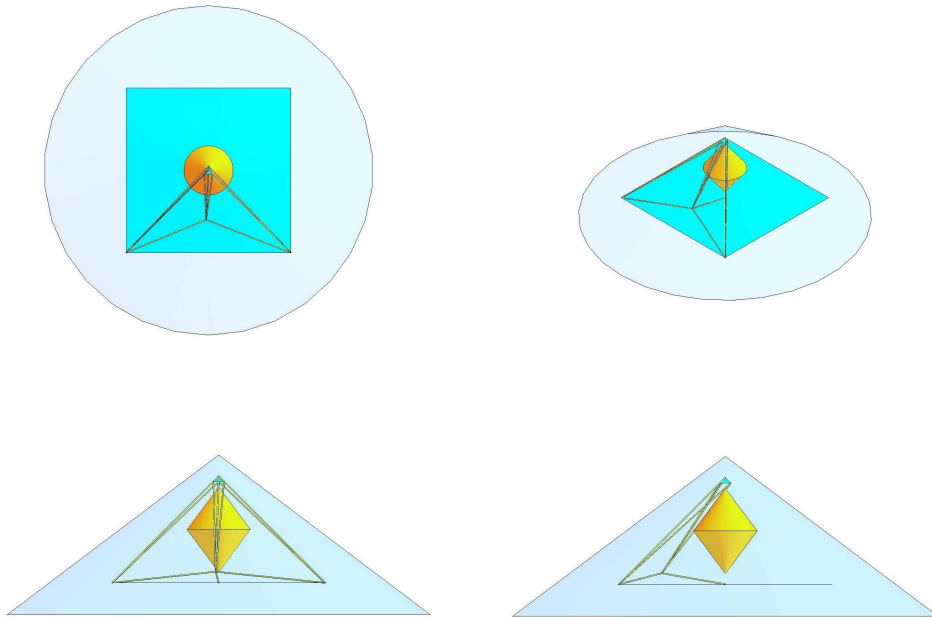


Figure 5. Example of Symmetric Result.

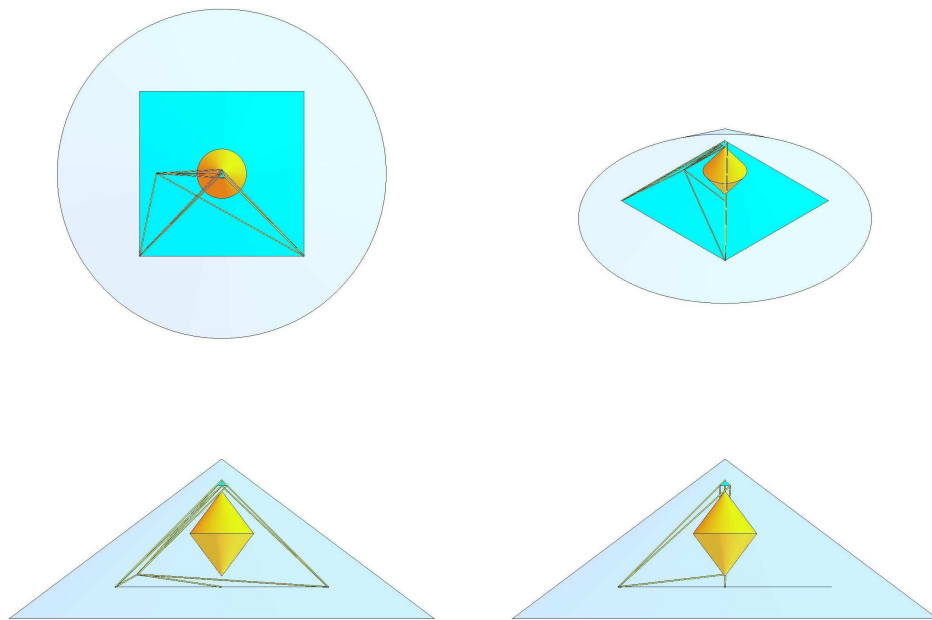


Figure 6. Example of Nonsymmetric Result.

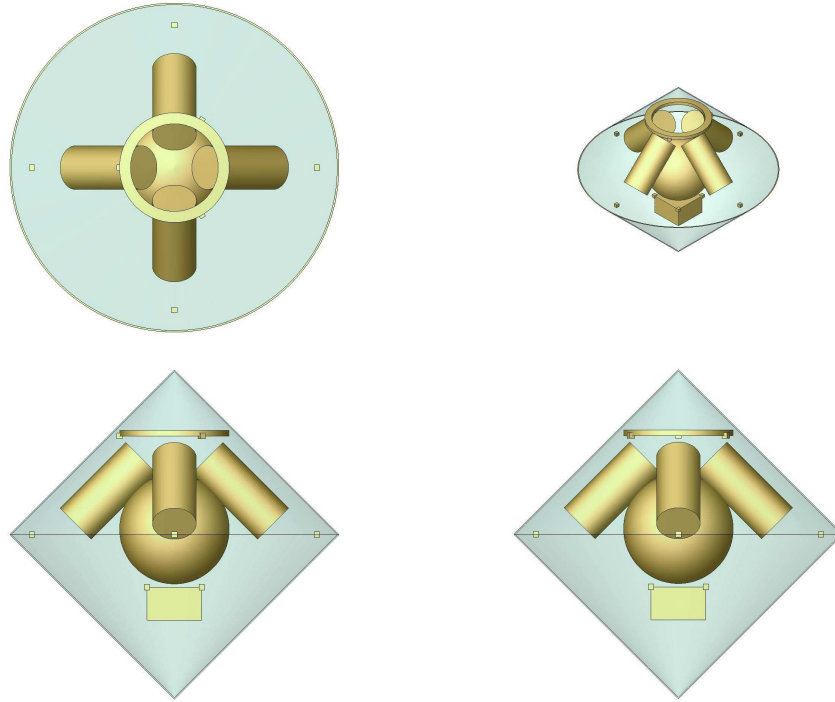


Figure 7. Setup for MSL Descent Stage Example.

maximize the stabilizing force as well as keep the plume clear of any important hardware. There are a total of 6 thrusters, placed 90° away from each other in a 1-2-1-2 pattern. In the figure, they are represented by a small box.

- o There are four large propellant tanks that provide the majority of the decelerating force. These are placed near the thrusters and almost define *arms* for the design. They are represented by rounded off cylinders.
- o There is a large spherical oxidizer tank to mix with the propellant. In this example, the tank is a large sphere halfway between the interface plate and the rover.

Note that the actual design of the vehicle has progressed further and no longer matches this description.

Because the dimensions and complexity of the feasible structure are different from Example 1, some preference functions are modified. Because preferences for vibrational mode and stress can be considered independent of the complexity of the truss, both preferences will be kept same as in Example 1. The preference for mass is modified by changing the value of M to 80. Finally, the cost preference will

be adjusted to:

$$\mu_c(C) = \begin{cases} 1 - 0.5 * \left(\frac{C}{20}\right)^2 & \text{for } C \leq 20 \\ 2 * \left(\frac{C}{40} - 1\right)^2 & \text{for } 20 < C \leq 40 \\ 0 & \text{for } C > 40 \end{cases} \quad (12)$$

where the variables are the same as before. The same number of iterations will also be used.

Example of symmetric and nonsymmetric configuration is given in figures 8 and 9. As in example 1, 15 different run for this example is performed also. 10 run did not find any feasible symmetric truss, so only best results will be mentioned only. The best symmetric structure has a fitness value of 0.9498, and the best nonsymmetric structure has a value of 0.9644. Thus, even with more realistic situation, there are distinct benefit of nonsymmetry if vibrational frequency is in design criteria.

5 Conclusions

This paper has demonstrated an approach to synthesizing novel structural configurations, including an initial approach to generating symmetric structures. With the ability

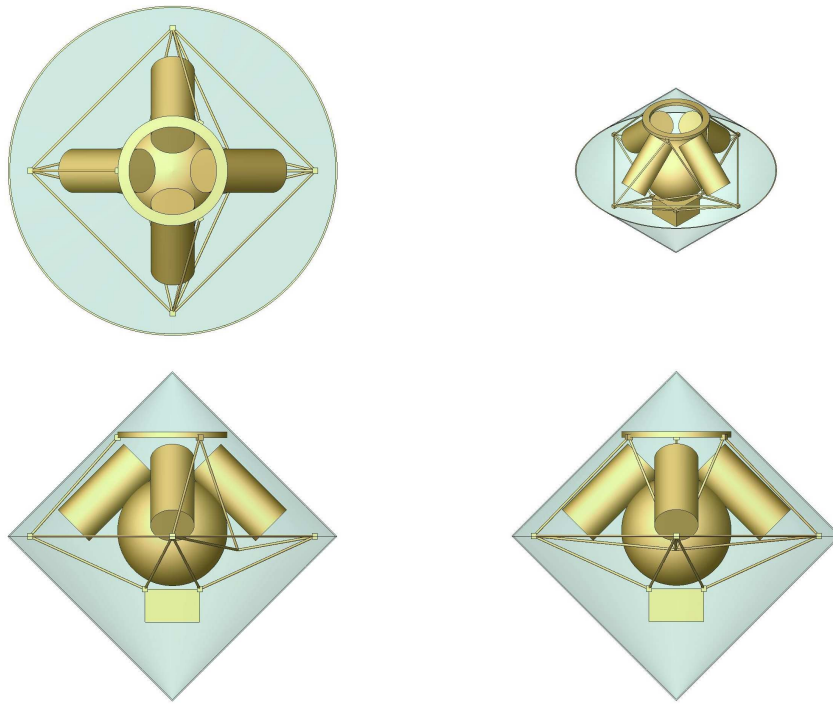


Figure 8. Sample Symmetric Result for MSL Descent Stage Example.

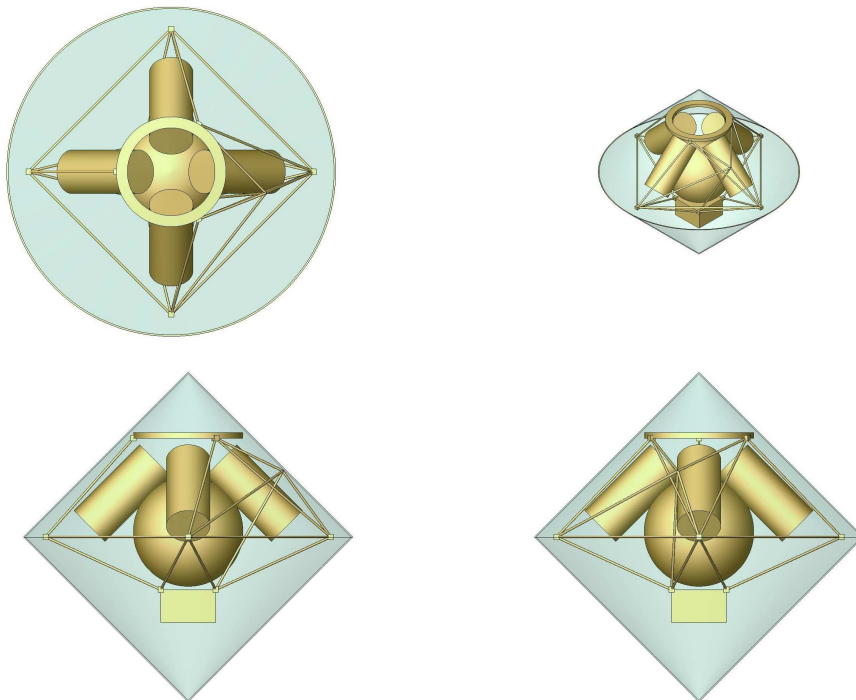


Figure 9. Sample Nonsymmetric Result for MSL Descent Stage Example.

to create and destroy nodes in the structure, the character of the search space has been greatly expanded over other evolutionary approaches to structural synthesis. The requirement to predetermine the complexity of the solution, or restrict the solution space to previously conceived configurations is removed. This allows the natural evolution of the approach to determine the optimal topology and details of the structure, giving more freedom to the range of solutions synthesized and evaluated.

The approach of incorporating cost into the overall aggregated measure of performance of the structure was utilized, along with a discount for repeated use of near-identical elements. The results are not conclusive: even deep discounting for cost did not produce symmetric structures, and those that were produced did not provide higher performance than asymmetric structures. Cost savings may not be the most critical motivation for developing symmetric structures; other factors such as aesthetics, packaging, and simplicity of design and analysis may play a more significant role in motivating symmetric design. With modern computational design synthesis and analysis methods, such as evolutionary approaches, the value of simple symmetric structures may be reduced, and the performance benefits of asymmetric structures (e.g., load sharing, vibrational response) may favor non-symmetric designs.

6 Acknowledgment

This material is based upon work supported, in part, by: NASA's Jet Propulsion Laboratory as part of "MSL Descent Stage Genetic Algorithm Optimization Study" and a Moore Fellowship supported by the Gordon and Betty Moore Foundation. Any opinions, findings, conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the sponsor(s).

REFERENCES

[1] Antonsson, E. K., and Otto, K. N., 1995. "Imprecision in Engineering Design". *ASME Journal of Mechanical Design*, **117(B)**, June, pp. 25–32. Invited paper. Special Combined Issue of the Transactions of the ASME commemorating the 50th anniversary of the Design Engineering Division of the ASME.

[2] Goldberg, D. E., 1989. *Genetic Algorithms in Search.*

Optimization and Machine Learning. Addison-Wesley, Reading, Mass.

[3] Antonsson, E. K., and Cagan, J., eds., 2001. *Formal Engineering Design Synthesis.* Cambridge University Press, Cambridge, U.K.

[4] Back, T., 1996. *Evolutionary Algorithms in Theory and Practice.* Oxford University Press, New York, N.Y.

[5] Lee, C.-Y., 2002. "Efficient Automatic Engineering Design Synthesis through Evolutionary Exploration". PhD thesis, California Institute of Technology, Pasadena, CA, June.

[6] Mitchell, M., 1996. *An Introduction to Genetic Algorithms.* MIT Press, Cambridge, Mass.

[7] Erbatur, F., Hasancebi, O., Tutuncu, I., and Kitc, H., 2000. "Optimal Design of Planar and Space Structures with Genetic Algorithm". *Computers and Structures*, **75**, pp. 209–224.

[8] Galante, M., 1996. "Genetic Algorithms as an Approach to Optimize Real-World Trusses". *International Journal for Numerical Methods in Engineering*, **39**, pp. 361–382.

[9] Deb, K., and Gulati, S. Kangal report no. 9901. Tech. rep.

[10] Rajan, S. D., 1995. "Sizing, Shape, and Topology Design Optimization of Trusses Using Genetic Algorithm". *Journal of Structural Engineering*, **121(10)**, pp. 1480–1487.

[11] Rajeev, S., and Krishnamoorthy, C. S., 1997. "Genetic Algorithms-Based Methodologies for Design Optimization of Trusses". *Journal of Structural Engineering*, **123(3)**, pp. 350–358.

[12] Bendsoe, M. P., and Soares, C. A. M., 1993. *Topology Design of Structure.* Kluwer Academic Publisher, Dordrecht, Netherlands.

[13] Bendsoe, M. P., and Sigmunds, O., 2003. *Topology Optimization: Theory, Method and Application.* Springer, Berlin, Germany.

[14] Li, Q., Steven, G. P., and Xie, Y. M., 2001. "Evolutionary Structural Optimization for Connection Topology Design of Multi-Component Systems". *Engineering Computations*, **18(3)**, pp. 460–479.

[15] Lyu, N., and Saitou, K., 2005. "Topology Optimization of Multicomponent Beam Structure via Decomposition-Based Assembly Synthesis". *Journal*

of Mechanical Design, **127**(2), pp. 170–183.

- [16] Scott, M. J., and Antonsson, E. K., 1998. “Aggregation Functions for Engineering Design Trade-offs”. *Fuzzy Sets and Systems*, **99**(3), pp. 253–264.
- [17] Antonsson, E. K., Nicaise, F., and Honda, T., 2005. “Configuraton Synthesis of a Variable Complexity Truss”. In 15th International Conference on Engineering Design, The Design Society.