Optimization and parametric modelling to support conceptual structural design

Kirk Martini

Optimization and parametric modelling to support conceptual structural design

Kirk Martini

Abstract

The paper describes software combining parametric geometric modeling with a version of the harmony search method, modified to support multimodal structural optimization. Researchers have recognized the potential of population-based optimization methods, such as genetic algorithms, to support multimodal optimization: that is, generating a diverse range of good alternative solutions, rather than a single best solution. Among these methods is the harmony search method, which has been demonstrated to be efficient in many unimodal structural optimization problems. The paper describes a new version of the harmony search method, implemented as an assembly within Bentley's Generative Components, enabling high-level control of geometry. The new method is demonstrated on an bridge supported by two inclined parabolic arches, a structure where GC controls a complex geometry with a single variable. Comparative studies of the example find that the new method is more effective than conventional harmony search in consistently finding multiple good solutions.

1. INTRODUCTION

For a structural optimization system to support decisions during the conceptual stages of design, it should have two key features: first, it should allow high-level control of structural geometry; and second, it should produce a diverse range of good solutions, rather than a single best solution. Concerning geometric control, a great deal of structural optimization research has not considered geometry at all, but has instead focused on the issue of selecting member sizes for a framework with a fixed geometry [1-3]. Research that has considered geometry has commonly employed low-level control, often using the coordinate values of individual nodes in a structural model as decision variables [4,5]. Saka [6] did a notable study involving optimization of geodesic domes. Given the span of the dome, the geometry of the framework was determined by two variables: the height of the dome and the number of meridian rings. The coordinates of structural nodes were calculated by a special-purpose program based on those variables. While this approach was effective, it is not general. Writing a special-purpose program may be workable when node locations are limited to the surface of a sphere, but becomes impractical with more complex geometries. A more general approach is to incorporate parametric geometric modeling software such as Bentley's Generative Components (GC) [7] in the structural optimization cycle.

Concerning the issue of generating multiple solutions, optimization methods with this property are called *multimodal*. One advantage of multimodal methods concerns the issue of fitness functions, which numerically evaluate the quality of a design. These functions typically cannot account for considerations which are difficult to quantify, such as aesthetics. An optimization algorithm that produces a range of alternatives allows a human designer to assess those alternatives with respect to external considerations [5,8,9]. In addition, a multimodal method may identify design approaches that a designer had not initially considered, which is particularly important in conceptual design [8].

Some researchers have noted the potential of metaheuristic population-based methods to support multimodal optimization [5,8]. These methods include genetic algorithms, particle swarm optimization, ant colony optimization, and several hybrids of these and other methods. The fundamental approach is to begin with a randomly generated population of solutions (which may range from 10 to 1000, depending on the method and problem) and then use stochastic rules to generate better solutions from the current population. The rules are typically based on some metaphor, such as biologic evolution, or the movement of swarms of animals. Hasancebi et. al [10] gives a thorough overview and comparison of such methods.

One such metaheuristic method is the harmony search method, based on a musical metaphor explained below. Harmony search has a property that it generates one new solution per cycle rather than an entire new population, potentially making it more efficient than other methods. Some researchers have found that harmony search performs better in unimodal optimization than some other metaheuristic methods [1]. This paper presents results of a study where a version of the harmony search method, modified to support multimodal search, is integrated with GC and a structural analysis program to perform structural optimization.

The remainder of the paper is organized as follows. Section 2 provides a formal statement of the structural optimization problem. Section 3 then describes the conventional harmony search method, plus the modifications proposed in this study for harmony search to support multimodal optimization. Section 4 describes the implementation and flow of data among programs. Section 5 then applies the new method to an example problem, a simplified model of a pedestrian bridge supported by inclined arches. Section 6 then offers a summary and conclusions.

2. STATEMENT OF THE OPTIMIZATION PROBLEM

A particular design configuration is called a solution, and is represented as a vector **x** of decision variables x_i. The objective is to find a solution which minimizes a fitness function $f(\mathbf{x})$ subject to multiple constraints:

$$g_j(\mathbf{x}) \leq 0 \tag{1}$$

A variable *xi* may vary continuously (as for geometric dimensions), or may represent discrete symbolic values from a predefined list (as for standard structural cross sections). For a continuous variable *xi*, the value must lie between a specified upper bound (x_i^U) and lower bound (x_i^L). Variables that vary continuously are called *geometry variables.* Variables representing symbolic values are called *section variables.*

Concerning constraints, there are three types: stiffness, strength, and stability. Stiffness constraints are formulated as shown in equation (2), where Δj is the displacement in a specified coordinate direction of a node *j*, and Δ_{allow} is an allowable displacement in that coordinate direction.

$$g_j(\mathbf{x}) = \frac{\Delta_j}{\Delta_{allow}} - 1 \tag{2}$$

Strength criteria, defined using the AISC LRFD steel design code provisions [11], are formulated so the constraint value is zero or negative when the internal forces are at acceptable levels, and positive when the internal forces are not acceptable. In addition to stiffness constraints for nodes and strength constraints for elements, the method can also account for stability constraints for the entire structure with an option to perform a large displacement analysis. If the analysis fails to converge, the solution is considered unstable and assigned a large constraint value ($1.0\text{x}10^6$ for example in this paper).

3. THE HARMONY SEARCH METHOD

3.1. Review of Conventional Harmony Search

The harmony search method was introduced by Geem et al. [12], and is based on a musical metaphor where variable values are viewed as musical pitches, and a solution vector is analogous to a musical chord. That method will be called conventional harmony search. The following is a brief summary of its essential steps:

1. **Initialization**. Produce a collection of solutions with random values. The collection is called harmony memory, and the number of solutions is the harmony memory size (HMS).
2. **Improvisation**: "Improvise" (i.e. generate) a new solution derived from the solutions in harmony memory using a stochastic procedure described below.
3. **Replacement**: If the fitness of the new solution is better than that of the worst solution in harmony memory, replace that worst solution with the new solution. This replacement strategy will be called *global replacement* (GR).
4. **Termination check**. If the termination criterion is not satisfied, go to step 2. The termination criterion is typically a specified maximum number of cycles.

Step 2 generates a new solution using three control parameters to guide the stochastic process: the harmony memory consideration rate (HMCR), the pitch adjustment rate (PAR), and the bandwidth (BW). The HMCR and PAR parameters represent probabilities. Improvising a new solution involves the following steps: First, initialize a new solution vector by assigning each variable a random value. For each variable, with a probability defined by HMCR, set the value to that of a randomly chosen solution in harmony memory. If the value was chosen from harmony memory, then with a probability defined by PAR, modify the value by a small change, called a pitch adjustment. For continuous variables, the pitch adjustment is calculated as a uniform random number on the interval [-1,1] times the bandwidth parameter BW, which is typically a small percentage of the possible range. For discrete variables, the pitch adjustment typically means a few discrete steps. The example in this study used a bandwidth of 1 percent of the variable range for continuous variables, and one discrete step for discrete variables. The success of the harmony search is typically sensitive to the values of the control parameters. [1]. The example in this study uses an adaptive method proposed by Hasancebi et al. [2], with initial values HMCR=0.8 and PAR=0.2.

3.2. Modifications for Multimodal Harmony Search

One characteristic of conventional harmony search is the tendency of the solutions to converge toward a common configuration. This characteristic supports unimodal optimization, but not multimodal optimization. In terms of the musical metaphor, multimodal optimization seeks to produce not a single harmonious chord, but rather a diverse collection of chords, as in an interesting piece of music. The proposed modifications to support multimodal optimization concern the improvisation and replacement steps of the algorithm, with the replacement step relying on an approach to constraint handling that avoids penalty functions. The modifications also rely on methods to measure similarity among solutions. The discussion begins with constraint handling and measures of similarity, and then moves to the replacement and improvisation steps.

Constraint Handling

Published implementations of harmony search have commonly handled constraints using penalty functions, which modify the value of the fitness function so that solutions with high constraint violations have poor fitness [1-4,6]. There are, however, several well-known drawbacks to that approach [13]. The method presented in this paper handles constraints by adapting the method proposed by Deb [13] for genetic algorithms. This method uses the following three rules for comparing solutions:

1. Any feasible solution is preferred to any infeasible solution.
2. Among two infeasible solutions, the one having smaller constraint violation is preferred.
3. Among two feasible solutions, the one having better objective function value is preferred.

When applied to harmony search, these rules create two modes of operation, called *culling*, and *refinement*. Culling concerns the application of rules 1 and 2 in guiding the replacement of infeasible solutions in harmony memory. Refinement concerns the application of rules 1 and 3 in guiding the replacement of feasible solutions in harmony memory.

Measure of Similarity: Normalized Euclidean Distance

The methods proposed in this study use *normalized Euclidean distance* as a measure of similarity between solutions. The following discussion reviews this concept and explains its application in guiding the search. The definition begins with the concept of *normalized value* for a variable. For a solution **x**, the normalized value v_i of the continuous variable x_i is defined by equation (3).

$$v_i = \frac{x_i - x_i^L}{x_i^U - x_i^L} \tag{3}$$

The concept of normalized value leads to normalized difference. Considering two solutions $\mathbf{x}^{(j)}$ and $\mathbf{x}^{(k)}$, the normalized difference $d_i(^{jk})$ between the corresponding variables $x_i^{(j)}$ and $x_i^{(k)}$ is defined as the difference between their normalized values:

$$d_i^{(jk)} = v_i^{(j)} - v_i^{(k)} = \frac{x_i^{(j)} - x_i^{(k)}}{x_i^{U} - x_i^{L}} \tag{4}$$

For section variables, which are assigned discrete symbolic values from a predefined list, the organization is similar but somewhat more complex. In addition to specifying the list of sections, the user also specifies the primary section property (e.g. area, or strong axis moment of inertia) for the variable. The numeric values of that section property serve as the basis for computing normalized difference per equation (4).

Using normalized difference, the normalized Euclidean distance $D(^{jk})$ between solutions $\mathbf{x}^{(j)}$ and $\mathbf{x}^{(k)}$ is defined as follows [13]:

$$D^{(jk)} = \sqrt{\frac{1}{N_d}\sum_{i=1}^{N_d}\left(d_i^{(jk)}\right)^2} \tag{5}$$

Where N_d is the number of design variables in a solution. For the sake of brevity, the following discussion uses the term *distance* to mean normalized Euclidean distance, unless otherwise noted.

Using these concepts, design space can be viewed as a unit hypercube with a dimension equal to the number of decision variables; any solution can be represented as a point in that space. One particularly useful concept is the maximum distance between two feasible solutions, which will be called the *feasible diameter* (D_f) of the design space. The new replacement strategy proposed in this study, described below, employs this concept. The algorithm maintains a lower-bound estimate of D_f at design cycle *i*, denoted $D_f^{(i)}$, by computing the distance between each newly generated feasible solution and all previously determined feasible solutions. The $D_f^{(i)}$ parameter is set to the maximum distance found so far in the search.

Replacement Strategy: Local Replacement

To counter the tendency of conventional harmony search to converge, the search algorithm employs a new replacement strategy called *local replacement*. When a new solution is improvised, local replacement first considers the set of feasible solutions whose distance from that solution is less than a specified distance R_n, called the *neighborhood radius*; the solutions in harmony memory within that radius are called *close neighbors* of the new solution; the number of close neighbors N_n is called the *close neighbor count*.

The close neighbor count is compared with a specified *neighborhood size*, S_n. If $N_n<S_n$, the neighborhood is called *uncrowded*; if $N_n=S_n$, the neighborhood is called *crowded*; and if $N_n>S_n$ the neighborhood is called *overcrowded*. When the neighborhood of a new feasible solution is uncrowded, the new solution is compared with and may replace the worst solution in harmony memory. If the neighborhood is crowded or overcrowded, then the set of close neighbors is sorted by fitness and the new solution is compared with the close neighbor with fitness rank equal to S_n. If the fitness of the new solution is better than the fitness of the *Sn*-ranked neighbor, then the new solution replaces that neighbor. When the neighborhood is overcrowded, then the algorithm employs *thinning*, where all solutions whose fitness is worse than that of the *Sn*-ranked close neighbor are *reset*, meaning that the constraint violation is set to a large value (1.0×10^5 for examples in this study) which marks the solution as infeasible. Local replacement inhibits convergence by limiting the number of feasible solutions within a neighborhood. When the algorithm finds a neighborhood that is crowded or overcrowded, it puts only better solutions there, not more solutions.

Local replacement depends on two parameters: S_n and R_n. The S_n parameter needs to be a fraction of the harmony memory size, HMS. The example in this study set S_n to HMS/5, using HMS=75 and S_n=15. Similarly, R_n needs to be a fraction of the feasible diameter D_f; if R_n were greater than or equal to D_f, then all feasible solutions could fall in one neighborhood, which would defeat the purpose. The examples in this study used $R_n = 0.25\ D_f^{(i)}$.

Improvisation strategy: Close Harmony

As described above, when conventional harmony search improvises a variable value for a new solution it may use a value from a randomly selected existing solution, where all solutions in harmony memory have an equal probability of being chosen. This approach will be called *full harmony improvisation* and has proven to work well for unimodal optimization. To better support multimodal optimization, the method proposed in this study uses a new improvisation strategy called *close harmony improvisation*, where the algorithm selects instead from a subset of closely related solutions. The algorithm randomly selects a solution from harmony memory, and then gathers all solutions within a specified distance. This set of solutions is called the *close harmony set*, and the specified distance is called the *close harmony radius* (R_h). The new solution is improvised from the close harmony set in the same way that full harmony improvisation generates a solution from the entire harmony memory. In order to make R_h adaptive, in this study it is set equal to the average distance among all solutions in harmony memory.

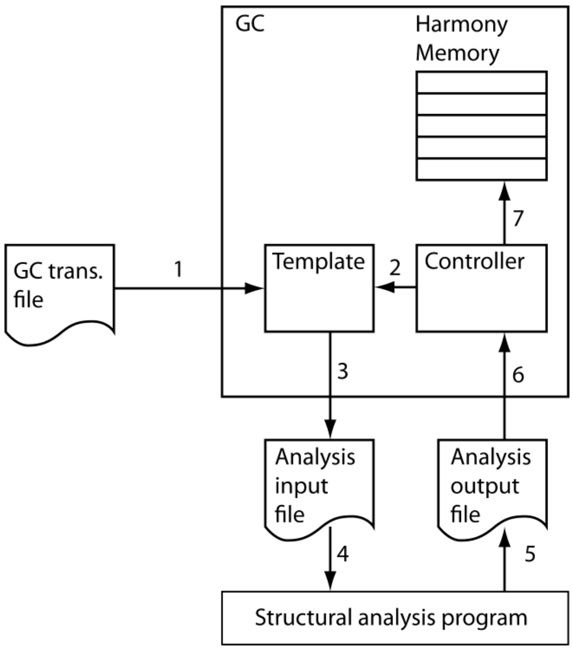
4. IMPLEMENTATION AND DATA FLOW

The system is implemented in the C# language, as an assembly within GC. Figure 1 shows the flow of data among software components. The key components are a controller, which manages the execution of the algorithm, and a template, which is a GC model that includes custom GC features defining aspects of the structural model, such as nodes, elements, supports, load patterns, etc. Geometry variables are represented by specially designated graph variables. Section variables are represented by a custom feature called a *section group*. The specification for a section group includes a list of strings, each of which corresponds to a standard steel cross-section designation. The list may include any number of cross-section designations, up the number available for the particular cross section type. Currently, the program supports all hollow round and rectangular sections defined by the American Institute of Steel Construction [11]. The specification of a structural element includes the name of the section group that defines its cross section. A single section group may specify the cross section for any number of elements in a model.

One of the most important custom features is the structural node. This feature is a subclass of the GC Point class, so that nodes can be positioned using all of the update methods available to GC Points. Since the geometry of a structural model is defined through the node coordinates, all the GC functions for controlling points are available to control structural geometry. Using GC's functionality, It is possible for the position of dozens or hundreds of nodes to be controlled through a single geometry variable.

The controller manages the algorithm, and the data flow corresponds to the numbered arrows in figure 1 as follows:

1. GC reads the transaction file defining the parametric structural template.
2. The controller improvises a new solution vector of geometry and section variables, and plugs the values into the template to create an instance of a model.
3. The controller uses the model instance to generate an input file for the structural analysis program.
4. The controller invokes the structural analysis program to read the input file and perform the structural analysis.
5. The structural analysis program emits the output file with the structural response.
6. The controller reads the analysis output file and evaluates all strength, stiffness and stability constraints.
7. Based on the constraint values and fitness value, the controller either replaces one of the solutions in HM with the new solution, or discards the new solution. The controller then returns to step 2 to begin the next cycle.



◀ Figure 1: Software organization and data flow.

In models which consider multiple load combinations, steps 3 through 6 are performed for each combination, step 7 then determines the maximum constraint values over all the load combinations. Currently, the structural analysis program is OpenSees [14].

5. EXAMPLE: BASKET-HANDLE ARCH BRIDGE

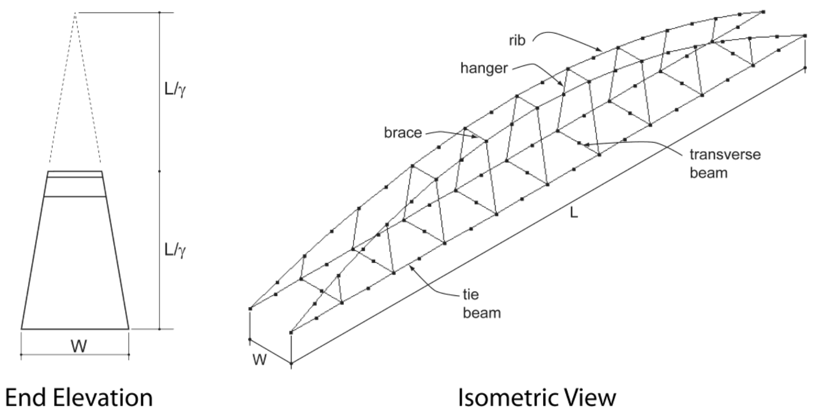
5.1. Overview

The following discussion considers the arch structure shown in figure 2, a simplified model of a pedestrian bridge with the arches in a "basket-handle" configuration. The span L= 30.5 m (1200 in.), the width W=2.44 m (96 in.). There are 10 structural bays along the span. The fitness is defined as the structural weight based on member length, cross section area, and a material density of 77.0 kN/m^3 (0.490 kip/ft^3). Cross sections are selected either from a list of 32 AISC HSS round sections, or a list of 75 AISC HSS rectangular tube sections; the lists were compiled by selecting the lightest section for each shape variety. The model includes one geometry variable, the span-to-depth ratio of the arch (γ), which ranges from 4 to 12, plus the five section variables indicated in figure 2 as follows:

- **Rib**: The main arch rib, a round HSS section.
- **Brace**: Members that connect the two arches near the crown, a round HSS section.
- **Hanger**: The suspenders which transfer load from the deck to the arch, a round HSS section.

- **Tie beam**: The longitudinal beams at the deck level, a rectangular HSS section.
- **Transverse beam**: The beams which span between the ties, a rectangular HSS section.

▶ Figure 2: Configuration of arch-bridge example.



▶ Figure 3: Geometric framework for the arch-bridge example. The only decision variable is the span-to- depth ratio, which determines the height of the mid-span control point E.

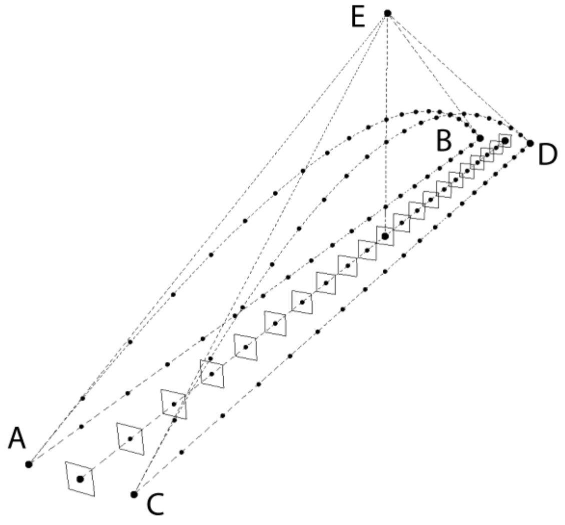


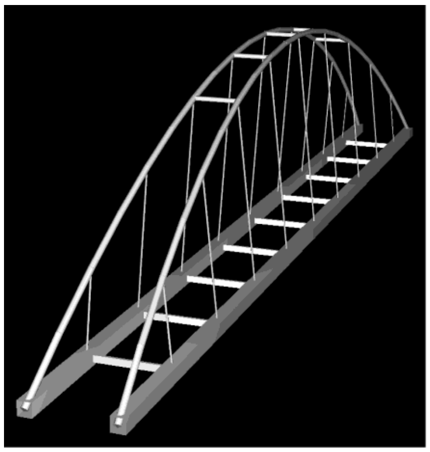
Figure 3 shows the geometric framework defined in GC. The points labeled A, B, C, and D define the four corners of the bridge deck. Two edge lines are defined with endpoints A-B and C-D. A centerline is defined bisecting the space between the edge lines. Point E is defined above the midpoint of the centerline. The height of point E above the deck is set by 2L/γ, for reasons explained below. Arch lines, the centerlines of the arch ribs, are defined as b-spline curves with three control points: one with the points A-E-B, and the other with C-E-D. Defining a b-spline in this manner results in a parabolic curve where the height of the parabola is one-half the height of point E, so that the height of the parabola equals L/γ. The parabola is the appropriate funicular shape for a uniformly loaded arch. A series of planes is defined normal to the centerline, equally spaced along intervals equal to one-half the structural bay. Nodes along the arch ribs are defined by the intersections of those planes with the arch lines. Nodes along the edge lines are defined by intersections of the planes with the edge lines. With this arrangement, the single decision variable γ controls the position of 38 nodes, 19 along each inclined parabolic arch line.

Concerning structural modeling, the structure uses pin supports at points A and C, and roller supports at points B and D. The hanger members are pin-ended, and the connections between the arches and the tie beams are also pinned. Concerning materials, the rectangular tube sections use a yield stress of 317 MPa (46 ksi), and the round sections use a yield stress of 290 MPa (42 ksi). Dead load includes the self weight of the model plus a superimposed dead load on the deck area of 3.83 kPa (0.08 kip/ft^2). The live load is 4.07 kPa (0.85 k/ft^2) distributed on the deck area. Superimposed dead and live loads are applied to the nodes of each transverse beam, at the beam ends and a midspan node according to tributary area. The vertical deflection of the nodes of the tie beam are limited to $L/1000$ for live load only. The analysis includes four load combinations: two to check stiffness, and two to check strength and stability. The combinations for stiffness include one with full live load, and one with live load on half the span; these combinations use linear analysis. The combinations for strength and stability include one with dead plus live, and another with dead plus live load on half the span; these combinations are factored according to the AISC LRFD code [11], and use large displacement analysis. Note that these load combinations are unrealistically simple, in particular since they do not account for lateral loads.

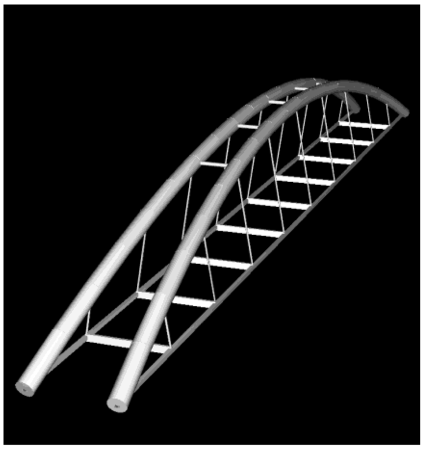
The study of this example considers four search methods based on the possible combinations of two improvisation strategies, full harmony (FH) and close harmony (CH), and two replacement strategies, global replacement (GR), and local replacement (LR). The four methods are denoted as follows: FH-GR, CH-GR, FH-LR, CH-LR. Note that FH-GR corresponds to conventional harmony search. The primary method of interest is CH-LR, which employs both of the new strategies. The CH-GR and FH-LR methods are included to better demonstrate the effect of each new strategy acting alone. For each of the four methods, the study performed 10 optimization runs of 4,000 design cycles each.

5.2. Results

The motivation for this example is that it is multimodal. Optimization runs reveal two distinct solution types that result in good fitness. Examples of these types are shown in figure 4 and figure 5 and will be called the "slender-arch" type and the "stiff-arch" type respectively. Although the stiff-arch solution type is clearly suboptimal in terms of structural weight, it could be preferred when esthetics are considered. The challenge for a multimodal optimization algorithm is to find good solutions of both types so that the designer can be aware of the range of viable options.



► Figure 4: A representative example of the slender-rib solution type.



► Figure 5: A representative example of the stiff-rib solution type.

Table 1 summarizes the results of the four methods. For each method, the table lists the average, best, and worst fitness for the best solution found over the 10 runs. In addition, the table includes notes concerning whether the solutions at the conclusion of each run included only the stiff-rib type, only the slender-rib type, or included both types. The results show that global replacement has a strong tendency to converge.

Both the FH-GR and CH-GR methods converged to the locally-optimal stiff-rib solution type in 5 of the 10 runs, and FH-GR converged to the slender-rib solution type in 4 of the 10 runs. In contrast, the methods employing local replacement, FH-LR and CH-LR, concluded with both solution types in all of their respective runs. For this problem, it is clear that local replacement is superior to global replacement, not only for multi-modal optimization, but also for unimodal optimization, since local replacement more reliably finds the globally optimum solution type.

Method	Fitness (kN)			Notes
	Average	Best	Worst	
FH-GR	76.5	70.9	79.1	5 runs concluded all stiff-rib type. 1 run concluded all slender-rib type. 4 runs concluded with both types.
FH-LR	74.8	72.9	78.8	All runs found both solution types.
CH-GR	75.4	71.8	78.5	5 runs concluded all stiff-rib type. 4 runs concluded all slender-rib type. 1 run concluded with both types.
CH-LR	73.7	69.3	76.2	All runs found both solution types.

◀ Table 1: Fitness results for best solution in ten runs of each method.

Concerning the question of full harmony vs. close harmony improvisation, the effect of close harmony improvisation is mildly positive. The CH-GR method produces an average best that is better than that of FH-GR, and CH-LR produces an average best better than that of FH-LR, by about 1.5 percent in both cases. Since the global replacement strategy is inferior for this problem, further comparison of the full harmony and close harmony strategies focuses on the methods that employ local replacement. In the preceding discussion, table 1 compared the fitness of the best solution found by each method; in considering multimodal optimization, it is also important to examine the fitness of locally-optimal alternative solutions. Table 2 compares the best slender-rib solutions, and the best stiff-rib solutions for the FH-LR and CH-LR methods. The results show that for the slender-rib solution, CH-LR produces better average, best and worst results among the 10 runs. For the stiff-rib solution type, CH-LR produces average, best, and worst results about 1 percent better than FH-LR, a modest but consistent difference.

Method	Fitness (kN)					
	Slender-rib solution			Stiff-rib solution		
	Avg.	Best	Worst	Avg.	Best	Worst
FH-LR	74.8	72.9	78.8	79.5	78.4	80.9
CH-LR	73.7	69.3	76.2	78.7	77.5	80.2

◀ Table2: Comparison of fitness for each solution type.

6. SUMMARY AND CONCLUSIONS

Concerning multimodal optimization, the example demonstrated the strong tendency of the global replacement strategy to converge toward a single solution type. This tendency is effective for unimodal optimization, provided that the solutions converge toward a global optimum, but in 5 of their 10 runs, both FH-GR and CH-GR converged to the locally optimum stiff-rib solution. The methods employing local replacement, FH-LR and CH-LR, found both solution types in all 10 runs, meaning that local replacement was far more reliable in finding both the globally and locally optimal solution types. Concerning close harmony improvisation, in both examples close harmony improvisation produced better fitness than full harmony improvisation, although the improvements were modest. In summary, local replacement had a profound effect in improving the diversity of the search result, while close harmony improvisation had a consistent but mild effect on improving fitness.

Concerning geometric control, GC enabled using a single geometry variable γ to control the 3D coordinates of the 38 nodes defining the inclined parabolic arches. This kind of high level control is a natural fit with optimization methods because it reduces the number of decision variables, and would be difficult to achieve with a special purpose program. The incorporation of parametric geometric modeling enables the application of structural optimization methods to a broader class of more realistic structures.

References

1. Degertekin, S. O. Harmony search algorithm for optimum design of steel frame structures: A comparative study with other optimization methods. *Structural Engineering and Mechanics*, 2008, 29(4), 391-410.
2. Hasancebi, O., Erdal, F., Saka, M.P., Adaptive harmony search method for structural optimization. *Journal of Structural Engineering,* 2010, 136(4), 419-431.
3. Lee, K.S., Geem, Z.W., Lee, S.H., Bae, K.W., The harmony search heuristic algorithm for discrete structural optimization. *Engineering Optimization* 2005, 37(7), 663-84.
4. Lee, K. S., and J. K. Song. Configuration optimization of truss structures using harmony search heuristic algorithm. *Advanced Materials Research*, 2007, 26-28(1) : 793-796.
5. Balling, R., Briggs, R., Gillman, K., Multiple optimum Size/Shape/Topology designs for skeletal structures using a genetic algorithm. *Journal of Structural Engineering*, 2006, 132, (7), 1158-1165.
6. Saka, M. P., Optimum geometry design of geodesic domes using harmony search algorithm. *Advances in Structural Engineering,* 2007, 10(6), 595-606.
7. http://www.bentley.com/en-US/Products/GenerativeComponents/ [14-7-2010]
8. von Buelow, P., Advantages of evolutionary computation used for exploration in the creative design process. *Journal of Integrated Design & Process Science* 2007, 11(3), 5-18.

9. Winslow, P., S. Pellegrino, S. B. Sharma, and Buro Happold, Free form grid structures, *Structural Engineer*, 2008, 86(3), 19-20.

10. Hasançebi, O., Çarbaş, S., Doğ an, E., Erdal, F., Saka, M. P., Performance evaluation of metaheuristic search techniques in the optimum design of real size pin jointed structures. *Computers & Structures*, 2009, 87(5-6), 284-302.

11. American Institute of Steel Construction, *Manual of Steel Construction: Load and Resistance Factor Design*, 2001.

12. Geem, Z. W., Kim, J.H., Loganathan, G.V., A new heuristic optimization algorithm: Harmony search, *Simulation*, 2001, 76, (2), 60-68.

13. Deb, K.. An efficient constraint handling method for genetic algorithms. *Computer Methods in Applied Mechanics and Engineering*, 2000, 186 (2-4), 311-338.

14. http://opensees.berkeley.edu/ [14-7-2010]

Kirk Martini

University of Virginia, United States

Email: martini@virginia.edu