Fall 2014

Truss size and topology optimization using harmony search method

Haitham Farah Hanna Al Rabadi
University of Iowa

TRUSS SIZE AND TOPOLOGY OPTIMIZATION USING HARMONY SEARCH METHOD

by

Haitham Farah Hanna Al Rabadi

A thesis submitted in partial fulfillment
of the requirements for the Master of
Science degree in Civil and Environmental Engineering (Structures, Mechanics and Materials)
in the Graduate College of
The University of Iowa

December 2014

Thesis Supervisor: Professor Colby C.Swan

Graduate College
The University of Iowa
Iowa City, Iowa

CERTIFICATE OF APPROVAL

_____

MASTER'S THESIS

_____

This is to certify that the Master's thesis of

Haitham Farah Hanna Al Rabadi

has been approved by the Examining Committee
for the thesis requirement for the Master of Science
degree in Civil and Environmental Engineering (Structures, Mechanics and Materials) at the December 2014 graduation.

Thesis Committee: _____
Colby C.Swan, Thesis Supervisor

_____
Jasbir S. Arora

_____
M. Asghar Bhatti

ABSTRACT

When a truss structure is to be designed two important questions should be answered. First, how should the truss that requires the minimum amount of materials look like? Second, what are the possible members' cross sectional areas that support the applied load safely? In this work these two questions are answered using a two stage optimization process. In stage one, starting with hyper-connected truss ground structures that fully encompass the design region together with specified support conditions and load cases, the discrete combinations and arrangement of truss members that will yield the lightest and stiffest structure are sought. In the second stage the minimum cross sectional areas of the members that satisfy the imposed constraints are sought. In both stages harmony search algorithms are employed. These represent one of the most recent heuristic, stochastic search optimization methods that, while relatively simple and robust, overcome some of the drawbacks of preceding search techniques such as: the ability to converge to a global optimum, and the ability to optimize non-differentiable and non-continuous functions. The viability of the proposed framework is documented on numerous benchmark problems from the literature.

TABLE OF CONTENTS

LIST OF TABLES

Table

LIST OF FIGURES

Figure

CHAPTER 1

INTRODUCTION

Civil engineering structures such as bridges, and buildings are of significant importance in facilitating our life and everyday activities. Traditional approaches to designing these structures that depend on the designer's prior experience and some common rules of thumb can be helpful, although they may not necessarily result in efficient and economical designs. Although both the traditional and the optimum design processes are iterative in nature they differ in the way in the way the iterative process is terminated. In traditional design a preliminary design is first introduced followed by analyses to determine if the structure's performance satisfies the necessary strength, stiffness, and stability specifications. If it does the design process is terminated, otherwise the design is refined and tested again. On the other hand, the iterative process in the case of the optimum design approach is not terminated after finding an adequate (or feasible) design, but only when a different termination or convergence criteria related to optimality is achieved [1].

Optimal structural designs are not limited to those that are most economical, but could also include those with optimized performance characteristics. The bottom line is that the criteria to be minimized or maximized (objective function) depends on the designer's needs. Nevertheless, in this thesis the goal or objective is always to obtain a structure that requires the least amount of material to satisfy specified constraints.

This first chapter contains three sections. In the first, a brief introduction to structural optimization is given. The second section presents a brief background on optimization methods, and finally the third section presents an overview of this thesis and its goals.

1.1 Structural Optimization

The general form of an optimization problem can be given as follows:

$$\begin{aligned} &\textit{Min or Max} && f(\mathbf{x}) \\ &\text{s.t} && h_i(\mathbf{x}) = 0,\ \text{i=1 to } p \\ & && g_j(\mathbf{x}) \le 0,\ \text{j=1 to } m \end{aligned} \tag{1.1}$$

Above, $h_i(\boldsymbol{x}) = 0$ is the i[th] equality constraint out of p equality constraints, and $g_j(\boldsymbol{x}) \le 0$ is the j[th] inequality constraint out of m inequality constraints. In structural optimization inequality constraints are frequently imposed to control the behavior of the structure such as by limiting the allowable displacements and the allowable stresses while avoiding instabilities such as those associated with buckling. It is required that calculated structural performance measures should not exceed maximum permissible values which are denoted in this text as u_{max}, σ_{max}, and σ_{bmax} (maximum buckling stress) . Inequality constraints can also be imposed to limit the structural members' cross sectional areas such that they fall between upper and lower limits , (A_{min}, A_{max}). On the other hand, equality constraints are generally imposed only to ensure that for design vector **x**, a bounded equilibrium solution **u** of the static load-deflection problem that follows exists:

$$\mathbf{K(x)\,u = f} \tag{1.2}$$

Here $K(\boldsymbol{x})$ is the stiffness matrix of the structure, $\boldsymbol{u}$ is a vector of nodal displacements, $\boldsymbol{f}$ is a vector of applied forces, and **x** denotes the vector of design variable values.

Structural optimization problems are generally divided into three classes [2]:

1. Size optimization problems: In this class of problems the goal is to find the optimum cross sectional areas or thicknesses for the structural members.
2. Shape optimization problems: In this class of problems the location of finite elements nodes are set as design variables and the goal is to find the optimum

location of the nodes. In other forms shape optimization is interpreted as finding the best function that describes the bounding shape of the structure [2].

3. Topology optimization: in this class of problems the goal is to find the optimal connectivity between the structural members that comprise the structure.

Ideally shape optimization is a subclass of topology optimization, and large number of nodes in the discretized structure could eliminate the need of shape optimization. Figures 1.1 shows an initial design, topology optimized design, and size optimized design for a cantilever truss.

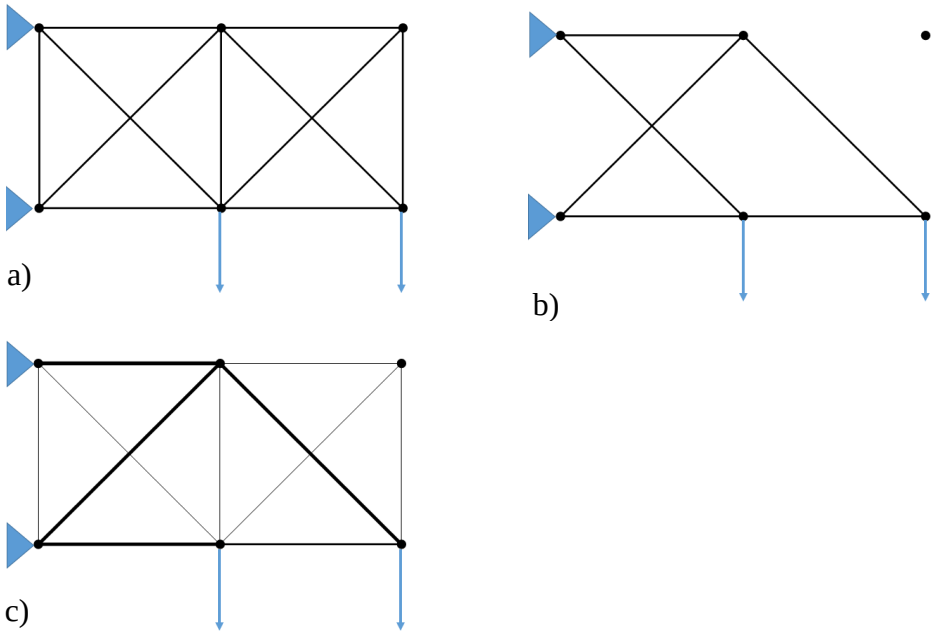


Figure 1.1: Different classes of structural optimization problems. a) Initial truss design; b) topology optimized truss; c) a design where the cross sectional areas of the members have been optimized and line weight are proportional to cross-sectional areas.

1.2 Optimization Methods and Techniques

Optimization techniques can, for the sake of simplicity, be divided into two basic groups: (1) gradient-based or classical methods; and (2) heuristic methods. Gradient-based techniques such as linear programing (LP) and nonlinear programing (NLP) employ calculus to find the local optimum solution(s) for continuous and differentiable functions. On the other hand, heuristic search methods attempt to find a global optimum solution using nature-inspired algorithms that consist mainly of two main features: intensification and diversification.

Both classical and heuristic methods have particular classes of problems to which they are best applied. Classical mathematical programming methods apply best to problems where the objective and constraint functions are continuous in the design space of **x.** Classical methods can also be applied equally well to both small problems involving only a few design variables, and to large problems involving hundreds of thousands of design variables. Heuristic methods, on the other hand, apply equally well to problems where the objective and constraint functions are continuously differentiable in the design space of **x**, or not. They rely on searching the design space of **x** with many, many function evaluations to find globally optimal solutions. While their ability to find the global optimum is a virtue, they tend to work best with relatively small numbers of design variables (say, less than 100 or so). [See Swan (2013), chapter 5 of *Structural Topology Optimization for Engineering Applications*]. For the discrete structural topology optimization problems to be addressed in this thesis, heuristic search methods are a natural choice.

Heuristic search methods involve a combination of rules and randomness, and most of them are inspired by naturally-occurring phenomenon. For example, simulated annealing [3] is a heuristic method inspired by the metallic annealing process in which a metal is cooled slowly from a high energy (temperature) state, and the structure of the

metal slowly changes until it eventually reaches the configuration of lowest energy, at which point no more changes happen. Dowsland (1995) noted that the relationship between the physical annealing process and the simulated algorithm can be represented as follows: the iterative process resembles the slow cooling process, and the values of cost function at different points simulates the energy at different temperatures, and finally the optimum solution resembles the minimum energy frozen state or the configuration [4].

The genetic algorithm is another heuristic method example that mimics the process of natural selection and evolution and is part of a bigger group of heuristic methods called evolutionary algorithms (EA) [3]. The relationship between genetic algorithm and the process of natural selection can be explained as follows: In nature, individuals compete for food and mates, with only the most fit individuals surviving to and spread their genes to later generations. Mixing many different good genes may produce a generation with high fitness. Similarly, in genetic algorithm, an entire group of individuals (or candidate solution) is called a "generation". The fitness of each individual in the generation is evaluated with only individuals of a high fitness surviving and are used to produce subsequent generations. The reproduction process continues until a termination criteria is satisfied, which is a usually a maximum number of generation production [6].

In 2001 Geem developed another heuristic technique known as the harmony search method [7] that mimics the improvisation of musical harmony by combining different musical notes. In this work harmony search and some of its variants are used to obtain optimum discrete truss topology solutions.

1.3 Thesis Overview: Two Stage Truss Optimization

As mentioned in the previous section, structures such as trusses can be optimized by varying the structure's size, shape, and topology. Although combining these three classes of optimization simultaneously can ultimately yield better results, the underlying

mathematical model becomes very complicated [8]. Therefore, in this work trusses are optimized using a two stage process. In the first stage the optimum truss topology for a given applied load is found. Starting with a hyper-connected ground structures, optimization process is performed to eliminate the unnecessary or least efficient structural members by minimizing the compliance of the structure subject to structural volume constraints.

After the optimum topology of the structure is found, the second stage is started wherein the goal is to find the optimum cross-sections of the members that compose the structure. This is achieved by minimizing the weight of the structure under stress, displacement, and buckling constraints.

CHAPTER 2
HARMONY SEARCH METHOD FOR UNCONSTRAINED OPTIMIZATION

2.1 Overview of Harmony Search

Harmony search by Geem [7] mimics the improvisation of a musical harmony and it is one of the most recent heuristic methods. Since it was introduced in 2001 and due to its simplicity, harmony search has been implemented to solve different optimization problems. In this text the focus is on the application of harmony search in structural optimization. In 2001 Geem [7] implemented the harmony search method to the design of a pipeline network and it was shown that the results obtained using the harmony search outperformed many other existed mathematical and heuristic methods. In 2004 Lee and Geem [9] applied the harmony search to size optimize various truss examples with continuous design variables and they concluded that the harmony search is a powerful search method compared to both mathematical methods and genetic algorithm. Discrete size optimization problems for trusses have been solved using harmony search by Lee et al [10]. In 2008 Degertekin [11] applied the harmony search to optimize steel frames and lighter structures were obtained than those optimized using genetic algorithm and (Ant Colony Optimization) ACO algorithm. Harmony search [12] is not limited to structural application it has also been extended to solve optimization problems in different engineering fields such as mechanical, industrial, geological and aerospace engineering.

The reminder of this chapter contains five sections devoted to explaining the harmony search: In the first and second sections the basic idea, steps, and parameters of the harmony search are introduced. The third presents both discrete and continuous unconstrained numerical examples while the fourth section presents an overview of some variants of the harmony search that have been presented in the literature. Finally, the

Explorative Harmony search method (EHS) is presented and discussed in the fifth section.

2.2 Harmony Search: basic ideas, parameters and steps

The harmony search method derives from an analogy with making music, or more specifically combining notes played by different instruments to create harmonious sounds. To begin the analogy, consider a musical group consisting of a fiddle, a saxophone, and a piano that wants to play and hold a three-note chord. For simplicity assume that each instrument can instrument can play only three possible notes: (C, A, or B) for the fiddle, (E, F, or D) for the saxophone, and (C, A, or, G) for the keyboard. The operative question is, what note should each instrument play to strike a chord of optimal harmony? In this example, there are only 3 x 3 x 3 = 27 possibilities for the chord, so each possibility could be tried out by enumeration and the optimum chord found in that manner. Alternatively, if one wants to use a method that will use fewer than 27 trials to find the optimum harmony, the harmony search algorithm can be employed.

Assume that three random trial harmonies have been generated and are stored in the so-called *harmony memory* (HM) of Figure 2.1. Each row in the HM represents a candidate harmony to be played: Harmony 1: C, E, C; Harmony 2: A, F, A; and Harmony 3: B, D, G. Each column represents the notes played by each instrument in the three trial harmonies under consideration. The fourth column in the HM of Fig. 2.1 is the evaluation column where the quality of each harmony is quantified based on its aesthetic value. In this illustration, let us assume that Harmony 1 is the best (has the best aesthetic evaluation), Harmony 3 is the worst, and Harmony 2 is intermediate. When a new harmony is to be improvised for consideration, each instrument can pick from any of its possible values. Let us say that for a new improvised harmony [C,F,G], C for the fiddle, F for the saxophone, and G for the keyboard. After the improvisation process the aesthetic value of this new improvised harmony is compared with that of the worst

harmony in the memory. If the new improvised harmony is superior to the worst harmony in the memory (Harmony 3), then it replaces the worst harmony in the harmony memory, otherwise the new harmony is ignored or forgotten and the improvisation and analysis process is repeated.

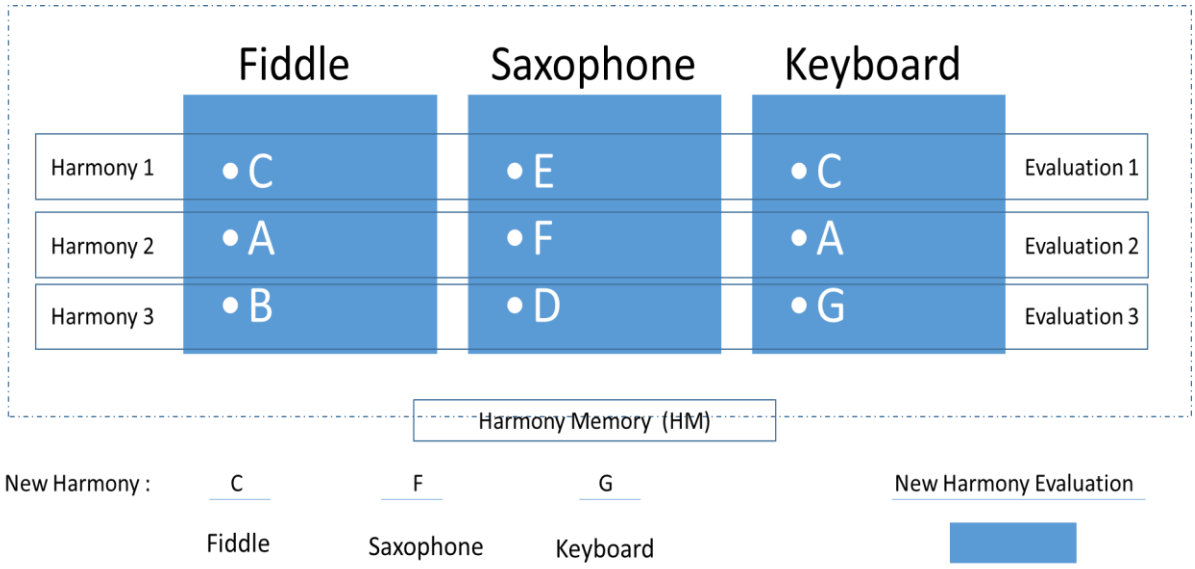


Figure. 2.1. Example harmony memory containing three harmonies (designs) each containing three notes.

From this example it is clear that the search for aesthetically pleasing harmonies is analogous to the search for unconstrained optimal design solutions. The particulars of this analogy are highlighted in Table 2.1 below. The specific details of how new harmonies (designs) are generated from the harmony (design) memory is crucial since it determines how many iterations will be required to converge to an optimum solution, and the quality of that optimum solution.

Harmony search	Unconstrained Design Optimization
instruments	components of system being designed
notes: set of possible tones to be played by instruments	design variables: set of possible design values for system components
chord: specific collection of notes	design: specific collection of design variables
harmony: synthesis of all notes	objective function: synthesis of all design variables
goal: maximize harmony	goal: extremize objective function
process: trial and error, improvise new harmonies and try them out	process: improvise on designs by trial and error

Table 2.1 Analogy between harmony search and unconstrained design optimization.

In real mathematical optimization problems, harmony search utilizes the so-called harmony memory (HM) to store the candidate solutions, and the harmony memory size (HMS) represents the number of candidate solutions stored in the harmony memory. The harmony memory is initialized by assigning random values to each design variable from its own set of possible values $\boldsymbol{X_i}$. With the initial HM established, the value of the objective function for each harmony is evaluated and the harmony with the worst (highest) objective function value is determined. Figure 2.1 shows a general format of the HM.

In each iteration a new harmony is improvised and its objective function is evaluated, if it has a better objective function value, it replaces the worst harmony in the harmony memory and the harmony memory is updated. The previous procedure is repeated until a termination criteria is met which is usually a predefined maximum

number of iterations or until no further improvement of harmony or design performance can be achieved.

2.2.1 The Classical Harmony Search Method

One of the key issues that governs the performance of the harmony search algorithm is how new designs are improvised in order to improve the overall harmony of the system. In the classical harmony search, this is achieved using the following constructs and algorithmic parameters:

- ***Harmony memory*** (hm). This is the working set of chords (designs) that are held in memory and that will be used to generate or improvise new chords (designs) for consideration.
- ***Harmony memory size*** (hms) is a parameter that specifies how many chords (designs) are actually held in memory.
- ***harmony memory considering ratio*** (hmcr) is a parameter on the interval [0,1] that specifies the weight that existing chords (designs) are given when improvising new ones.
- ***pitch adjustment ratio*** (par) is another parameter on the interval [0,1] that weights the extent to which chords (designs) taken from memory are adjusted or not in the improvisation process.
- ***bandwidth parameter*** (bw) is an additional parameter that dictates them extent to which design variables or notes can vary when their pitch is adjusted.

In mathematical optimization problems, harmony search utilizes the so-called harmony memory (HM) to store the candidate solutions, the number of candidate

solutions stored in the harmony memory equals the harmony memory size (HMS). The harmony memory is initialized by assigning random values to each design variable from the set of possible range $\boldsymbol{X_i}$. Let's consider the following initial harmony memory , the value of the objective function for each harmony is evaluated and the harmony with the worst (highest) objective function value is determined. Figure 2.1 shows a general format of the HM.

In each iteration a new harmony is improvised and its objective function is evaluated. If better than the worst design in the harmony memory, the new harmony (design) replaces the worst harmony in the harmony memory and the ordering of the harmony memory is updated. The process is repeated until a termination criteria, usually associated with a predefined maximum number of allowable iterations, or changes in the harmony memory composition, is satisfied.

2.2.2 Harmony Search Parameters

The harmony search depends on a number of parameters that affect its performance such as: Harmony Memory Consideration Ratio (HMCR), Harmony Memory Size (HMS), Pitch Adjusting Ratio (PAR), and a band width parameter (bw). The HMCR indicates the probability of choosing a value from the HM for the design variable and it falls in the range between 0 and 1. For example a value of 0.75 means that the probability of picking a design variable value from the harmony memory is equal to 0.75 and the probability to pick a random value for the possible range is 0.25.

Once a decision is made to pick a value (pitch) for a design variable from the harmony memory (memory consideration), then another test is performed to see whether this picked value should be adjusted or perturbed to another value within the admissible range. The process of adjusting the chosen value called pitch-adjustment and has a

probability equal to $(HMCR \times PAR)$, and the probability of keeping the chosen value as it is without pitch adjustment is $(HMCR \times (1 - PAR))$.

If the pitch adjustment is to happen then the chosen value (pitch) is adjusted either to a larger or smaller value. This decision is made by comparing a randomly generated number r with a value of 0.5, where $r \in [0,1]$.

The harmony memory size (HMS) is the number of harmonies (candidate solutions) that can be stored in the HM. In most examples presented in the literature HMS varies between 10 and 50, depending on the type of the problem to be optimized.

2.2.3 Harmony Search Steps

Harmony search consists of the following five steps:

1. Formulate the optimization problem and specify the harmony search parameter values
   (HMCR, PAR, HMS, bw).
2. Initialize the HM.
3. Improvise a new harmony and evaluate its performance.
4. Update the HM as necessary (ie, if the improvised harmony is superior to one or more harmonies in memory).
5. Check for the termination criteria. If satisfied stop. Otherwise return to step 3.

Each step is defined further and explained as follows:

Step 1: Formulate Optimization Problem/ Specify HS Parameters

The first step is basically expressing the objective function and the constraints mathematically. For the present we assume that the only constraints imposed are on

design variables' ranges (upper and lower limits), or they come from the set of admissible values.

$$\min_{x} f(\boldsymbol{x})$$
$$\text{where: } \mathbf{x} = \{ x_i \in \mathbf{X}_i,\ i = 1,2,3,...,N \} \tag{2.1}$$

where:

x_i : i^{th} design variable value

$\boldsymbol{X}_i$: set of possible values of the design variable x_i .

N: Number of design variables.

Harmony search parameters should be specified with care, a relatively high HMCR value very close to unity reduce the probability of introducing new values to HM. Conversely picking a small HMCR close to zero value reduces the exploitation of the members stored in HM. With these considerations in mind HMCR values within the range 0.5 to 0.9 are usually used and typically PARs are selected from the range [0.3, 0.8]. Finally HMS values ranging from 10 to 50 have been used in this study.

Step 2: Initialize the HM

a. Case of continuous variables

In this step each component of each candidate solution vector is initialized by assigning a value from the uniform distribution of admissible values. For example the i^{th} variable is continuous with $x_i \in \left[{}_L x_i, {}_U x_i \right]$ then

$$x_i^j = {}_L x_i + \left({}_U x_i - {}_L x_i \right) \cdot ran[0,1] \tag{2.2a}$$

where $= 1,2,\ldots,N$; $N =$ number of design variables ; $j = 1,2,3 \ldots., HMS$.

b. Case of discrete variables

If $x_i \in \{V_{1i}, V_{2i}, V_{3i}, \ldots, V_{Ki}\}$ are the K candidate values for x_i then $x_i^j = V_{li}$ where

$$l = 1 + \text{int}\left[ran[0,1) V_K \right] \tag{2.2b}$$

The harmony memory (HM) can be represented in a matrix form as follows:

$$HM = \begin{bmatrix} x_1^1 & \ldots & x_N^1 & f(x^1) \\ . & . & . & \\ . & . & . & \\ . & . & . & \\ x_1^{HMS} & \ldots & x_N^{HMS} & f(x^{HMS}) \end{bmatrix} \tag{2.3}$$

Step 3: New Harmony Improvisation

As mentioned above each improvised harmony $\mathbf{x}' = \{x_1', x_2', \ldots, x_N'\}$ is obtained either from the harmony memory or generated randomly from the set of admissible values. If the component's value comes from the harmony memory it is further tested to see if it should be pitch adjusted. The pitch adjustment process is different for continuous design variables and discrete design variables. Each component in the improvised harmony can be represented as follows as is done here for the i^{th} component

a. For Continuous design variables:

$$x_i' = \begin{Bmatrix} HM[m,i], P = HMCR \times (1 - PAR) \\ HM[m,i] + (ran[0,1] \cdot bw), P = 0.5\, HMCR \times (PAR) \\ HM[m,i] - (ran[0,1] \cdot bw), P = 0.5\, HMCR \times (PAR) \\ x_{random}, P = (1 - HMCR) \end{Bmatrix} \tag{2.5a}$$

b. For discrete design variables:

$$x_i' = \begin{Bmatrix} HM\left[m,i\right],\ P = HMCR \times \left(1 - PAR\right) \\ HM\left[m+1,i\right],\ P = 0.5\,HMCR \times \left(PAR\right) \\ HM\left[m-1,i\right],\ P = 0.5\,HMCR \times \left(PAR\right) \\ x_{random}\ ,\ P = \left(1 - HMCR\right) \end{Bmatrix} \qquad (2.5b)$$

where:

m : represents a random row number, $1 \le m \le HMS$.

Typically $1 + int[ran[0,1) \times HMS]$

$HM[m,i]$: Is the member with row number m and column number,

$i = 1,2,3,\ldots,N$.

ran : Randomly distributed number between (0,1).

x_{random} : Random value from the admissible set $\boldsymbol{X}_i$ for the i^{th}variable.

Equations 2.5 represent the different ways a new harmony can be improvised. As mentioned earlier HM consists of *m* rows and *N* columns , where each column represent candidate values that can be assign to the i^{th} design variable. The first case in Eqs (2.5a) and (2.5b) represent picking values randomly from the HM without pitch adjusting. The second and the third cases represent picking random values from HM with pitch adjustments. And finally the last case represents picking a value randomly from the admissible set for the design variable, $\boldsymbol{X}_i$.

Step 4: Update HM

If the new improvised harmony is better than any of the harmonies in the HM then it goes into the HM replacing the worst harmony. Otherwise the improvised harmony is neglected. A better harmony is one that has better fitness function which is simply the objective function in case of unconstrained problems. In the case of the

constrained problems, fitter harmonies are the one those that have better objective function values and less constraint violations.

Step 5: Check the termination criteria:

Usually the harmony search stops when it reaches a specified maximum number of iterations, however; some researchers have used other termination criterion such as Cheng et al [13]. In their work the iteration process is terminated when the change in the objective function is less than a small value after a specified number of iterations. For the examples presented in this chapter, the former method is used.

2.3 Summary of the Harmony Search Steps and Algorithm

The overall flow of the harmony search algorithm is shown in Figure 2.2 (adapted from Lee [14]). The improvisation of a new design for continuous variables is shown in Figure 2.3, while that for discrete variables is shown in Figure 2.4.

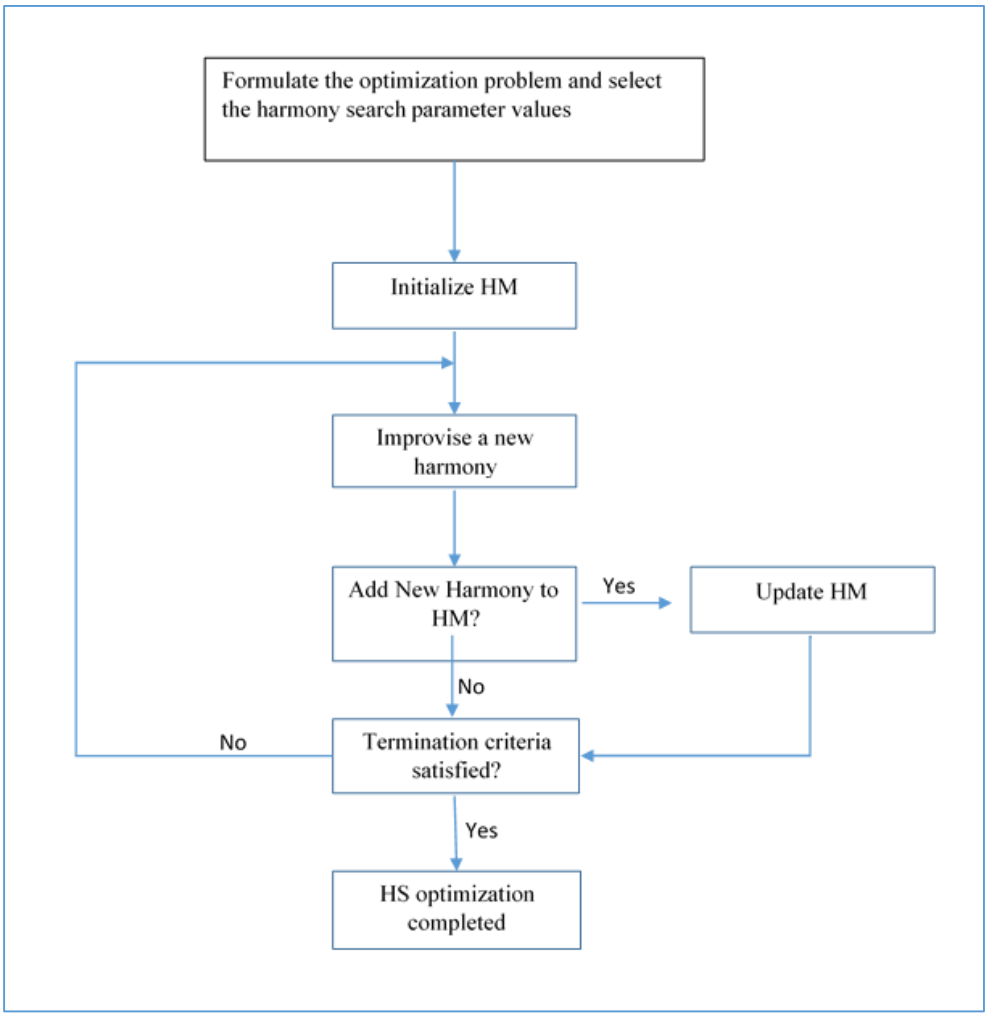


Figure 2.2: Classical HM steps for unconstrained optimization process.

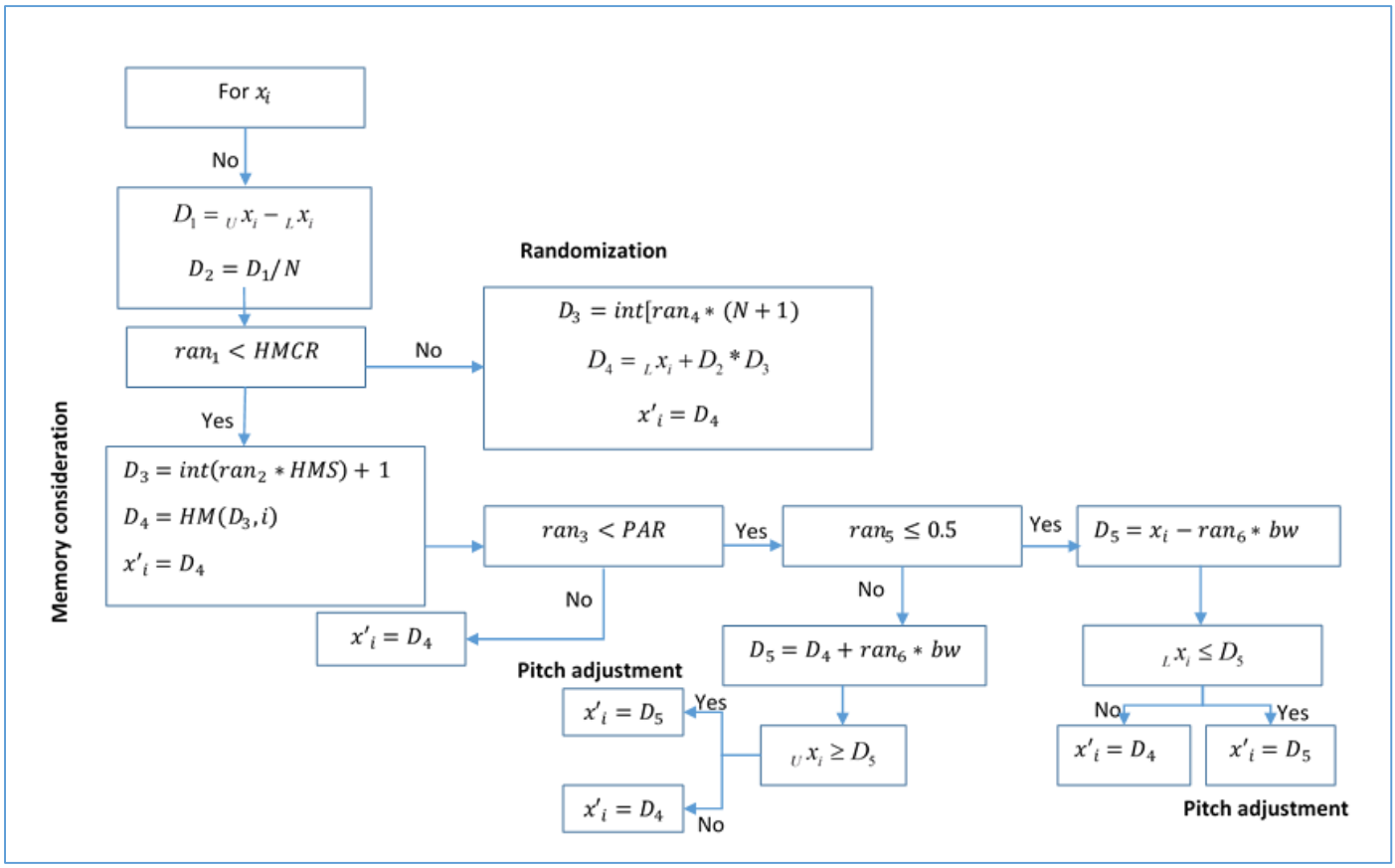


Figure 2.3: New harmony improvisation for a continuous variable x'_i.

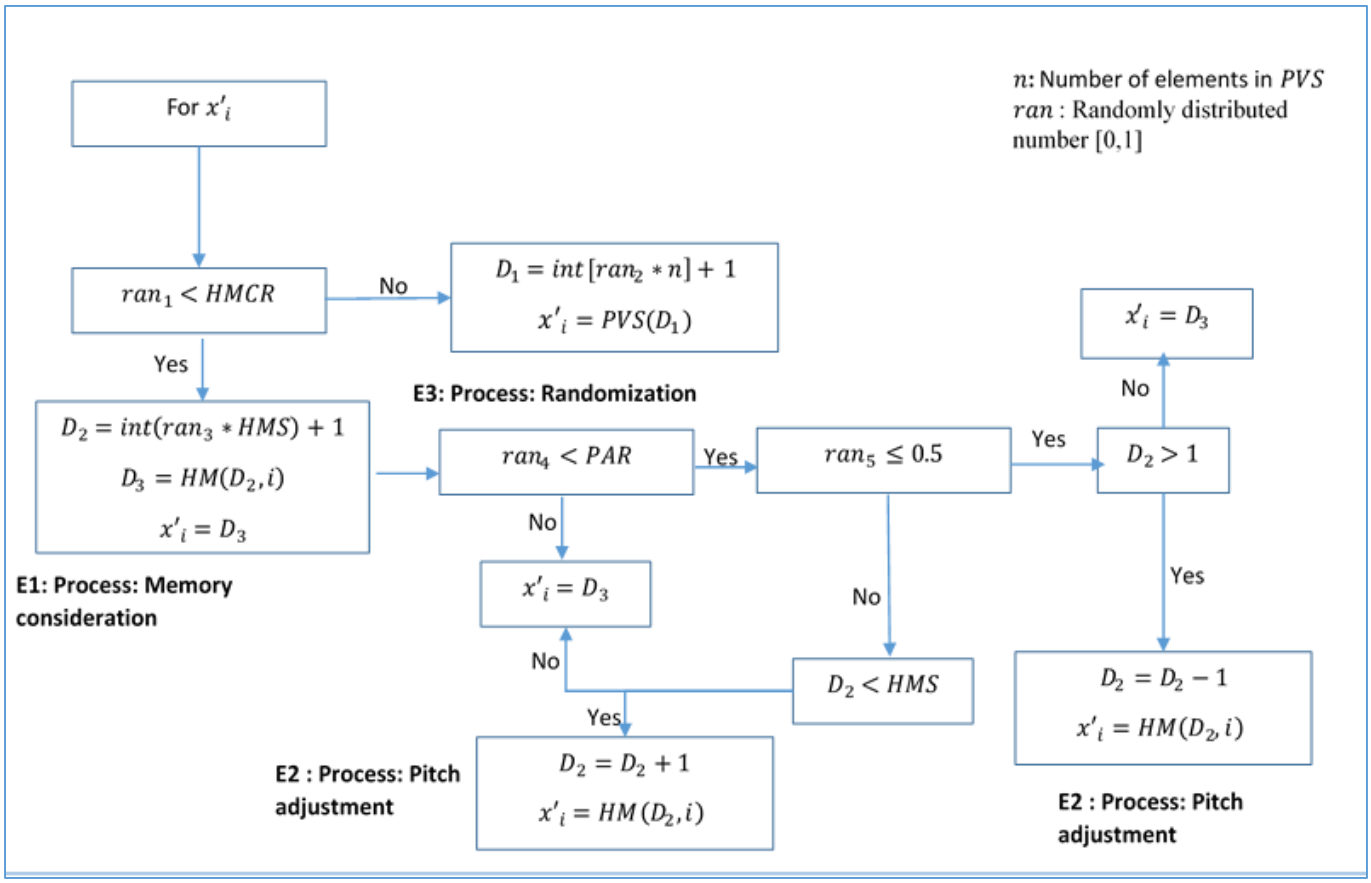


Figure 2.4: New harmony improvisation for a discrete variable x'_i.

2.4 Unconstrained Numerical Examples

2.4.1 Example Problem 2.1: A Discrete Unconstrained Example

In the following a simple unconstrained numerical example is solved to further explain the basic steps and ideas of the harmony search. The problem involves finding the global minimum of the function which by inspection occurs at the point $(x_1, x_2, x_3) = (0,1,2)$

$$\begin{aligned} &\textit{Minimize}: \quad f(x_1, x_2, x_3) = x_1^2 + (x_2 - 1)^2 + (x_3 - 2)^2 \\ &x_1, x_2, x_3 \in \boldsymbol{X} = \{0,1,2,3,4,5,6,7\} \end{aligned} \tag{2.6}$$

This Problem is solved as a discrete problem with 3 design variables, each of which can take on 8 potential values. There are thus $8^3 = 512$ possible solution to this problem. The harmony search utilizes the so called harmony memory (HM) to store the HMS candidate solutions and for this problem HMS = 4 , HMCR=0.8 ,and PAR= 0.3.
The harmony memory is initialized by assigning random values to each design variable from the set of admissible values $\boldsymbol{X_i}$, as indicated by equation 2.3. For this example an initial HM is as shown in Figure 2.5

Initial HM				
	Design variables			
Harmony number	x_1	x_2	x_3	$f(\boldsymbol{x})$
1	0	3	5	13
2	2	2	5	14
3	6	1	0	40
4	7	1	2	49

Figure 2.5: A starting HM for example problem 2.1

In each iteration a new harmony is improvised and its objective function is evaluated. If it has a better function value then it replaces the worst harmony in the harmony memory. To illustrate, assume that a new improvised harmony is $x' = (5,2,1)$ with a corresponding function value equal to 27. Since the performance of this harmony exceeds that of both harmonies 3 and 4 in the HM, the new improvised harmony replaces the worst harmony ie. (7,1,2) in the initial HM, and the HM is updated as shown in Figure 2.6 .

Updated HM				
	Design variables			
Harmony number	x_1	x_2	x_3	$f(\boldsymbol{x})$
1	0	3	5	13
2	2	2	5	14
3	6	1	0	40
4	5	2	1	27

Figure 2.6: Updated HM after first iteration of example problem 2.1

The process of improvising new harmonies, checking their performance and incorporating them into the HM continues until a specified number of iterations have occurred or until the performance of the best design (harmony) can no longer be improved. With the parameters specified, the algorithm took an average of **37** iterations to find the optimum point$\boldsymbol{x}' = (0,1,2)$. Since in this explanatory example the optimum point is known by inspection, the HS algorithm is terminated immediately after the harmony $\boldsymbol{x}' = (0,1,2)$ improvised and included in HM.

The choice of parameters HMS, HMCR, PAR and bw can have an enormous influence on the performance of the harmony search algorithm. To illustrate, example 2.1 was solved with five different values for HMCR (0.1, 0.3, 0.5, 0.8, and 0.9). Since the harmony search is stochastic, the problem was solved 10 times with each of the different HMCR values .The maximum, minimum, and average number of iterations required to find the optimum point are shown in Figure 2.7 and Table 2.2. Table 2.3 shows the number of iterations required to find the optimum for each trial of the different HMCR values.

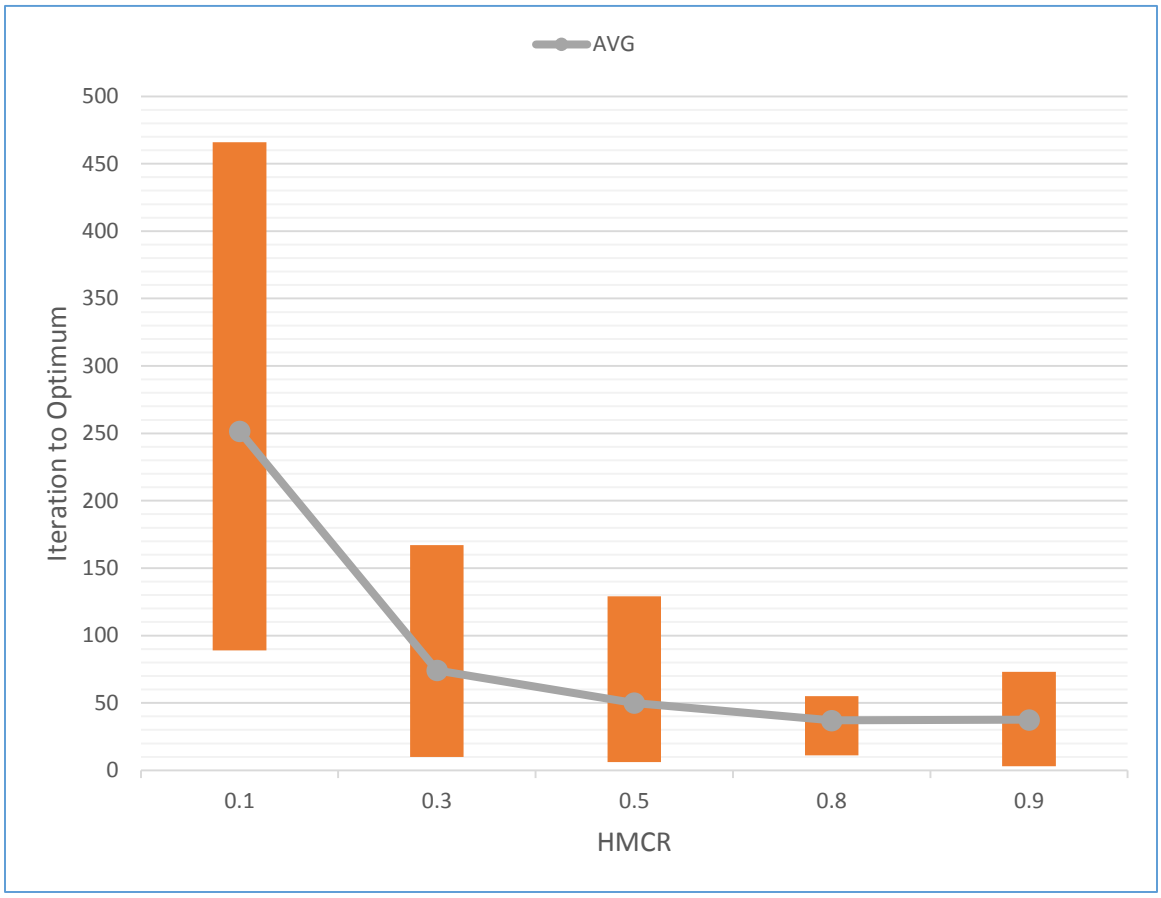


Figure 2.7: Influence of HMCR on number of iterations required to solve example problem 2.1. The problem was solved ten times with each value of HMCR to obtain a range of iterations for convergence.

As shown in Figure 2.7 the HS method gains its best performance for this example when HMCR is around 0.8.

	HMCR				
	0.1	0.3	0.5	0.8	0.9
Minimum iterations	89	10	6	11	3
Maximum iterations	466	167	129	55	73
AVG	251.5	74.2	50	37	37.5

Table 2.2: Maximum, minimum, and average number of iterations to solve example problem 2.1 for different HMCR values. PAR=0.2 for all calculations.

	HMCR=0.1	HMCR=0.3	HMCR=0.5	HMCR=0.8	HMCR=0.9
Trial	Number of Iterations	Number of Iterations	Number of Iterations	Number of Iterations	Number of Iterations
1	363	69	61	31	73
2	84	123	109	28	48
3	450	167	51	51	16
4	199	45	129	34	16
5	251	36	31	55	25
6	139	91	33	23	15
7	466	23	9	20	73
8	165	111	6	78	67
9	89	10	21	11	3
10	309	67	50	39	39
AVG	252	74	50	37	30

Table 2.3: Number of Iterations required to find the optimum point for different HMCR values of problem 2.1

Similarly in order to study the influence of PAR on problem 2.1, six different PAR values (0.99, 0.8, 0.5, 0.3, 0.2, and 0.1) were tested and each was run 10 times. The HMCR value was fixed and equals 0.8. Figure 2.8, Table 2.4, and Table 2.5 show the number of iterations required to find the optimum point for the different PAR values. It can be seen that for this example HS method gains its best performance when the PAR value ranges between 0.3 and 0.5.

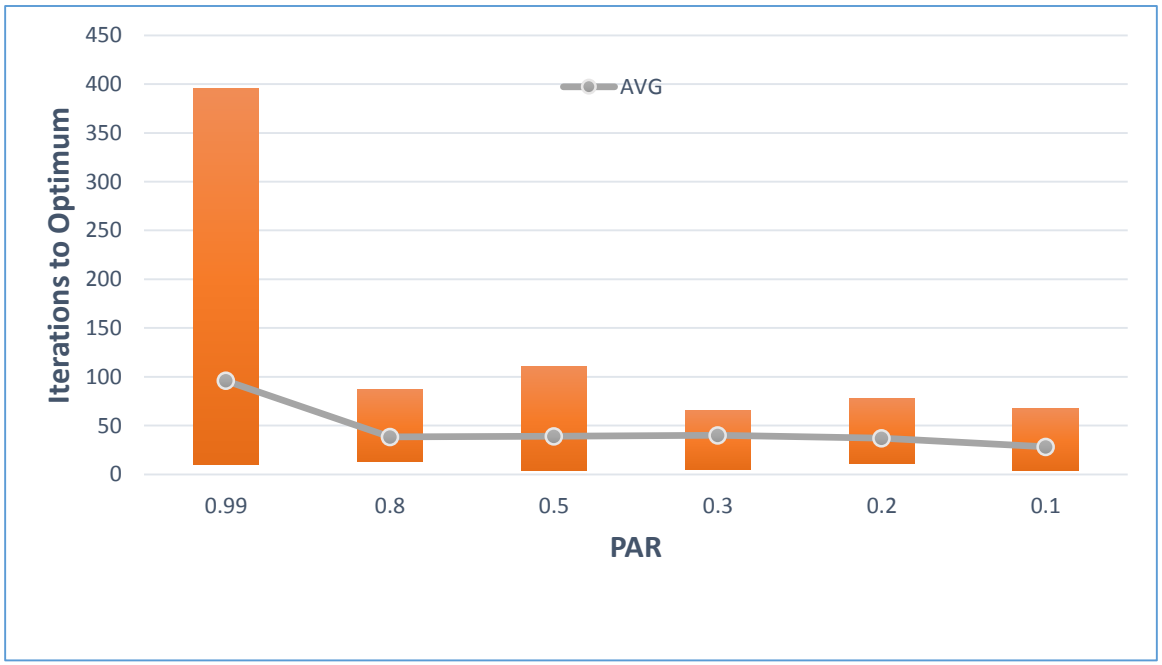


Figure 2.8: Influence of PAR on example problem 2.1 solved with HMCR=0.8.

The previous problem has a unique global optimum solution. It is thus not surprising that after many iterations the harmony memory is homogeneous with all the harmonies identical, as shown in Figure 2.9.

	PAR					
	0.99	0.8	0.5	0.3	0.2	0.1
Min iterations to optimum	10	13	4	5	11	4
Max iterations to optimum	395	87	110	58	78	67
AVG	96	74	39	40	38	28

Table 2.4: Maximum, minimum, and average number of iterations for different PAR values to solve example 2.1

PAR=	0.99	0.8	0.5	0.3	0.2	0.1
Trial	Iter	Iter	Iter	Iter	Iter	Iter
1	56	33	27	20	31	29
2	120	31	105	58	28	67
3	14	51	30	14	51	4
4	129	52	110	15	34	34
5	32	13	45	49	55	19
6	395	73	10	57	23	45
7	45	11	17	65	20	24
8	10	18	18	5	78	22
9	103	14	4	65	11	16
10	54	87	24	52	39	22
AVG	96	38	39	40	37	28

Table 2.5: Number of Iterations required to find the optimum point for different PAR values of problem 2.1

Harmony number	Design variables			
	x_1	x_2	x_3	$f(\boldsymbol{x})$
1	0	1	2	0
2	0	1	2	0
3	0	1	2	0
4	0	1	2	0

Figure 2.9: Homogenous HM where all harmonies consist of global optimum

2.4.2 Example 2.2: A Continuous Unconstrained Example

The forgoing example 2.1 was solved with discrete design variables, but here it will be solved with continuous design variables that all range between -5 and 5, $-5 \le x_i \le 5$ for $i = 1,2,3$ .The problem is stated as follows:

$$\min_{\mathbf{x}} : \quad f(x_1, x_2, x_3) = x_1^2 + (x_2 - 1)^2 + (x_3 - 2)^2$$
$$-5 \le x_i \le 5 \text{ , i=1,2,3} \qquad (2.7)$$

As before this problem has three design variables with a minimum value of zero at the point $(x_1, x_2, x_3) = (0,1,2)$. For problem 2.2, HMS = 10 , HMCR=0.8 ,PAR= 0.4 ,and bw = 0.1.

The harmony memory is initialized by assigning random values to each design variable from the set of admissible range, as indicated by equation 2.2. For this example an initial HM shown in Table 2.7.

x_1	x_1	x_3	$f(x)$
2.25	4.33	−4.12	53.73
2.26	−0.70	−0.34	13.52
4.12	−0.14	−2.00	34.4
−3.97	4.74	−3.87	64.28
1.02	4.98	2.69	17.44
−1.62	1.84	−1.06	12.76
2.44	4.05	−1.57	28.13
1.18	−4.61	−0.65	40.01
−0.96	4.08	1.00	11.44
3.66	4.85	4.63	35.24

Table 2.6: A starting HM for example problem 2.2.

A near optimum cost function value of 0.0004 at the point (x_1, x_2, x_3) =(.0001, 0.98, 1.983) was obtained in less than 1000 iterations. Figure 2.10 shows the change if $f(x)$.

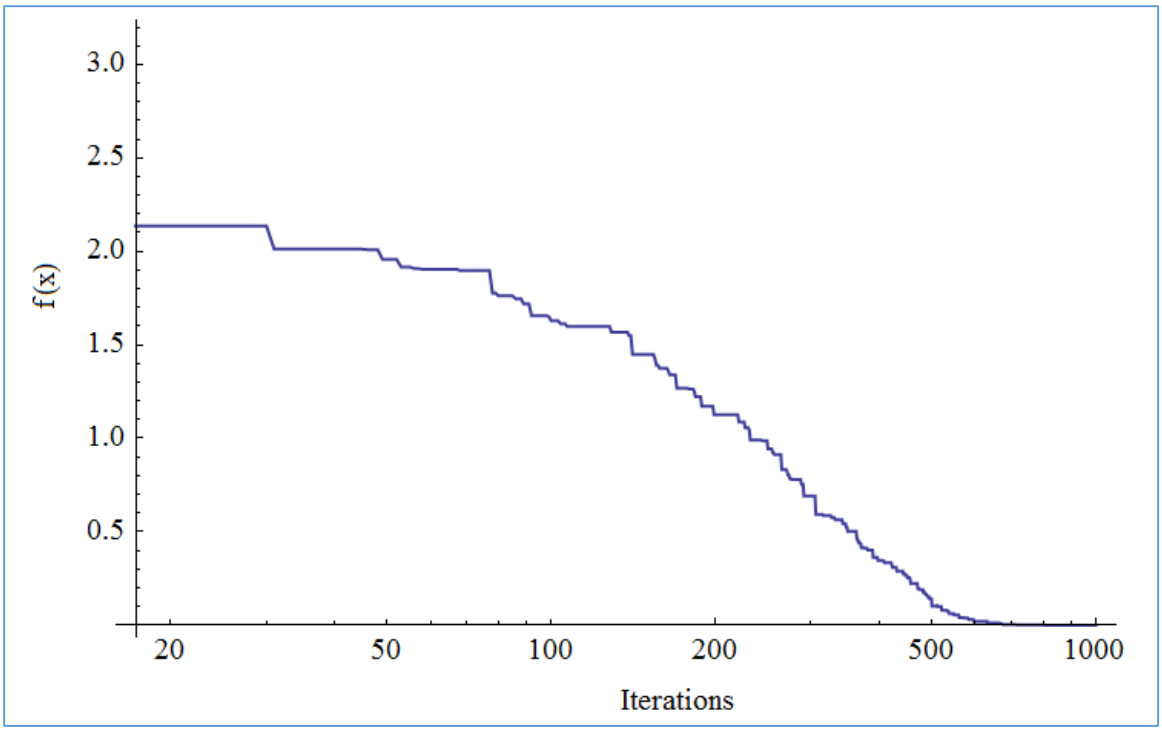


Figure 2.10: *f(x)* vs. iteration count for example2.2

2.5 Variants of Harmony Search

Since it was first introduced in 2001, variations of the harmony search have been proposed and investigated with the goal of improving the performance of the algorithm in both constrained and unconstrained problems.

Degerteki[15] modified the initialization process by generating two harmony memories instead of one. Other researchers such as Mahdavi et al [16], proposed expressions for selecting and dynamically updating the search parameters such as HMCR, bw, and PAR, as the search progresses. Some of these expressions are given as follows:

$$\alpha_g = \alpha_{\min} + \frac{(\alpha_{\max} - \alpha_{\min})}{NI} \times g \tag{2.8}$$

$$bw_g = bw_{\max} \exp(c.g)$$
$$c = \frac{\mathrm{Ln}\left(\frac{bw_{\min}}{bw_{\max}}\right)}{NI} \tag{2.9}$$

where:

α_g : Pitch adjusting rate for each generation.

α_{min} :Minimum pitch adjusting rate.

α_{max} : Maximum pitch adjusting rate.

NI : Number of solution vector generations.

α_{max} : Maximum pitch adjusting rate.

bw_g : bandwidth for each generation.

bw_{min} :Minimum bandwidth.

bw_{max} : Maximum bandwidth.

Hasancebi and Saka [17] developed the adaptive harmony search in 2009 in which the HMCR and PAR are adjusted dynamically as follows:

$$\mu^k = \left(1 + \frac{1-\bar{\mu}}{\bar{\mu}}. e^{-\gamma N(0,1)}\right)^{-1} \tag{2.11}$$

$$\alpha^k = \left(1 + \frac{1-\bar{\alpha}}{\bar{\alpha}}. e^{-\gamma N(0,1)}\right)^{-1} \tag{2.12}$$

where:

μ^k: Harmony memory consideration ratio for the k-th iteration.

α^k: Pitch adjusting ratio for the k-th iteration.

$N(0,1)$: Is a normally distributed number between 0 and 1.

γ: Is a learning rate of control parameters which is recommended to be in a range of [0.25 . 0.5] .

$\bar{\mu} = \sum_{i=1}^{n} \mu^i / n$

$\bar{\alpha} = \sum_{i=1}^{n} \alpha^i / n$

n : is the harmony memory size.

The harmony search method has been also modified by changing the termination criteria as in the work of Cheng et al [13] work where the iteration process is terminated when the change in the objective function is less than a small value after a specified number of iterations.

The original HS has also been hybridized with other optimization methods such as: artificial bee colony [18] , firefly method [19] , sequential quadratic programming [20] and many other methods.

2.6 Explorative Harmony Search (EHS)

2.6.1 EHS Basic Idea

EHS was developed by Das et al [23] and published in one of the few papers that mathematically analyzed the harmony search method .The efficiency and robustness of any heuristic method depends on both its explorative power and its exploitation power, and its ability to balance between them. Exploration (diversification) is the ability of the algorithm to explore the design space, while exploitation (intensification) is the ability of the algorithm to use and exploit the information gathered during the search process to converge to an optimum. Too much intensification might cause the algorithm get trapped in local minima, while too much diversification causes the search to scatter around some potential optima in the search space.

Das studied the exploratory powers of the HS algorithm by analyzing the variance of the population over successive generations and introduced the following analytical expression for the expected population variance:

$$E\left(Var(Y)\right)=\frac{m-1}{m}\left[\mu\ \cdot\ Var(x)+\mu\ \cdot(1-\mu)\ \cdot\ \bar{x}^2 + \frac{1}{3}\ \cdot\mu\ \cdot\ \alpha\ \cdot\ b_w{}^2+\frac{a^2}{3}(1-\mu)\right] \quad (2.13)$$

where:

μ: Is the harmony memory consideration ratio.

α: Is the pitch adjustment ratio.

$\boldsymbol{Y} = (Y_1, Y_2, \dots, Y_m)$ is an intermediate population obtained by new harmony improvisation.

$\bar{\boldsymbol{x}} =$ population mean.

$b_w =$ an arbitrary distance band width.

$m =$ harmony memory population size (HMS).

$(-a, a) = (x_{min}, x_{max})$ design variable's upper and lower limits.

For the sake of simplicity of the analysis the problem is assumed to consist of one decision variable and the improvised vector in each step is a vector of candidate solutions. "Now, if we can show that the population variance over generations is increasing by applying only the variation operators, it can be inferred that the algorithm has good explorative power".

In order to guarantee that the expected population variance is increased "the distance bandwidth parameter (bw) is chosen to be proportional to the standard deviation of the current population" $bw \propto \sigma(x) = \sqrt{Var(x)}$. So if in equation (2.13) $bw = k\sqrt{Var(x)}$ and HMCR is chosen to be very close to unity – to eliminate the

terms that contain [1-HMCR], the expected population variance for the gth population becomes:

$$E\left(Var(x_g)\right)=\left\{\frac{(m-1)}{m}\ .\ \mu\ \left[1+\frac{1}{3}\ .\ k^2\ .\ \alpha\right]\right\}^g\ .\ Var(x_0) \tag{2.14}$$

It is important that the value of the parameters HMCR, PAR, and *k* should be chosen in such a way that the term in the curly brackets is larger than unity which makes $E\left(Var(x_g)\right)$ grow exponentially.

In the original paper by Das el at [23] where the EHS was published, no discrete examples were presented in this work, however, the basic idea of EHS, making the bandwidth proportional to the population standard deviation, is extended by modifying the basic HS algorithm. In Eq. (2.5.a) it can be seen that the pitch can be adjusted as follows $x'_i = x_i + ran[0,1]\ .\ bw$, in the case of continuous problems x'_i can be any value as long as $x'_i \in X_i$; however, in the case of discrete optimization problems x'_i should be one of the X_i values, thus after the pitch is adjusted it is rounded to the nearest value that is existed in X_i. For example if a discrete design variable x_i with a corresponding HM vector $HM_J^T = \{1.2\ ,\ 1.3\ ,\ 4.5\ ,\ 2\ ,\ 6.7\}$ is set to be 4.5 based on memory consideration. If x_i is to be adjusted according to the classical HM rules then x'_i can be either 1.3 or 2 .However according to this work when EHS is used x'_i can be any value in the possible set. Figure 2.11 shows the pitch adjustment process for discrete problem using EHS.

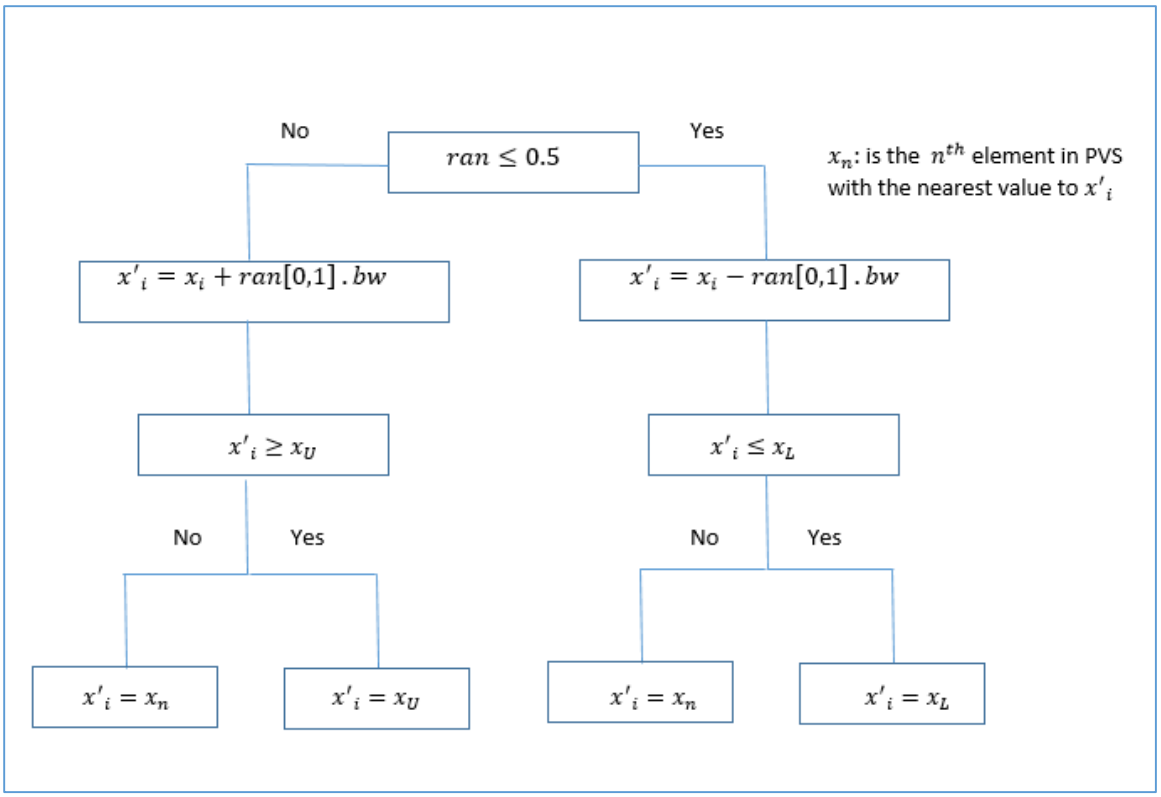


Figure 2.11: Pitch adjustment for discrete problems.

2.6.2 EHS Unconstrained Examples

2.6.1.1 Goldstein and Price (with four local minima)

$$f(x)=\left[1+\left(x_1+x_2+1\right)^2\left(19-14x_1+3x_1{}^2-14x_2+6x_1x_2+3x_2{}^2\right)\right]$$
$$\times\left[30+\left(2x_1-3x_2\right)^2\times\left(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2\right)\right] \quad (2.15)$$

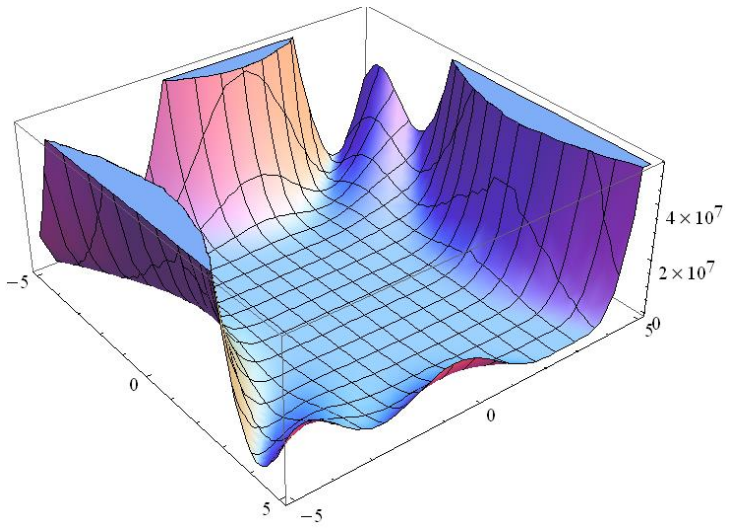


Figure 2.12: The Goldstien and Price function.

For this 2 design variable problem, EHS parameters were set as follows: HMCR = 0.93 , PAR =0.75 , HMS = 10. This Problem has an optimum value of 3 at (0,1). Using the EHS, the optimum value could be obtained in less than 300 iterations compared to 10 000 iterations for the basic HS method. Figure 2.1 shows the change in ***f(x)*** with the number of iteration.

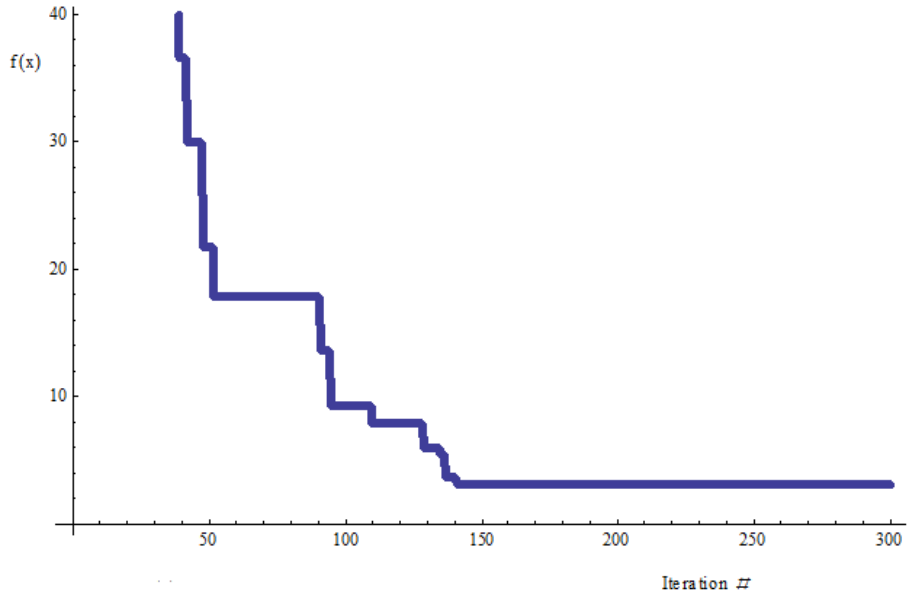


Figure 2.13: History of ***f(x)*** the Goldstien and Price function during solution of example 2.2. with EHS

CHAPTER 3

HARMONY SEARCH METHOD FOR CONSTRAINED OPTIMIZATION

Since practical optimization problems especially structural problems are always subjected to constraints, we discuss constraint handling in this chapter followed by numerical and structural applications.

3.1 Constraint Handling

Transformation methods are widely used to solve constrained optimization problems. In these methods a constrained problem is transformed into a sequence of unconstrained problems. Such methods employ the so-called penalty function which is a combination of the original objective function, the constraints functions, and some penalty parameters. After transforming the constrained problem, it is minimized as an unconstrained problem and if there are any constraint violations, the cost (objective) function is penalized by adding a positive value. The *quadratic loss function* is considered one of the most popular penalty functions:

$$P(h(\mathbf{x}), g(\mathbf{x}), r) = r\left\{\sum_{i=1}^{p}\left[h_i(\mathbf{x})\right]^2 + \sum_{i=1}^{m}\left[g_i^+(\mathbf{x})\right]^2\right\}$$
$$g_i^+(\mathbf{x}) = \max[0, g_i(\mathbf{x})] \tag{3.1}$$

Another widely used penalty function is the augmented Lagrangian:

$$P\left(\boldsymbol{h}(\boldsymbol{x}), \boldsymbol{g}(\boldsymbol{x}), \boldsymbol{r}, \boldsymbol{\theta}\right) = \frac{1}{2}\sum_{i=1}^{p} r_i^{'}\left(h_i + \theta_i^{'}\right)^2 + \frac{1}{2}\sum_{i=1}^{m} r_i\left[\left(g_i + \theta_i\right)\right]^2$$
$$where:\ \theta_i, r_i,\ \theta_i^{'}\ and\ r_i^{'}\ \text{are larger than 0} \tag{3.2}$$

The drawback of using (3.1) and (3.2) is that penalty parameters should be chosen judiciously since they are problem dependent parameters.

In order to avoid the risk of choosing unsuitable penalty parameters for the examples presented in this work the Fitness Priority-Based Ranking Method (FPBRM) (Dong, Tang, Xu, and Wang, 2005) is used[21,22]. In this approach a fitness value between 0 and 1 is given to each candidate solution based on its constraints satisfaction. A value of 1 means that all constrained are satisfied while a value of zero means that none of the constrained are satisfied. The constraint fitness function can be represented as follows:

$$F_{con} = \sum_{i=1}^{P} w_i F_i(x) \tag{3.4}$$

For inequality constraints:

$$F_i(x) = \begin{cases} 1, & g_i(x) \le 0 \\ 1 - \dfrac{g_i(x)}{g_{\max}(x)}, & g_i(x) > 0 \end{cases} \tag{3.5}$$

For equality constraints:

$$F_i(x) = \begin{cases} 1, & \mathrm{h}_i(x) = 0 \\ 1 - \dfrac{|h_i(x)|}{h_{\max}(x)}, & \mathrm{h}_i(x) \neq 0 \end{cases} \tag{3.6}$$

Above, $h_{max}(x)$ $g_{max}(x)$ are the maximum constraint violation for equalities and inequalities respectively and w_i is a weighting factor that ranges between [0,1] where $\sum_{i=1}^{P} w_i = 1$. For the examples represented in this work w_i is calculated as follows:

$$w_i = \left(NI + NE\right)^{-1} = 1/P \tag{3.7}$$

Where NI is the number of inequality constraints and NE is the number of equality constraints.

3.2 Constrained Example

3.2.1 Continuous Constrained Example

In this section we present two structural optimization problems. In both examples the design variables are cross sectional areas of the members which are continuous over the given range. The objective function to be minimized is the weight of the structure.

3.2.1.1 Ten-Bar Planar Truss (Loading Case 1)

Figure 3.1 (adapted from Lee [14]). shows a ten-bar planar truss to be size optimized . Structural material properties are as follows he Material density $(\rho) = 0.1\ \text{lb}/in^3$; Modulus of elasticity = 10000 ksi , σ_{all} =25 ksi (both in tension and compression). An absolute maximum displacement constraint is imposed on the structure of 2.0 in. , The value of the load P_1 is 100. kips, the EHS parameters were set as follows: HMCR = 0.8, PAR = 0.3 , maximum number of iterations= 50,000 . The minimum cross-sectional area for each member was set to be 0.1 in^2.

The problem can be formulated as follows:

$$\begin{aligned} &\min_x: \quad f(\boldsymbol{x}) = \sum_{i=1}^{5} l_i x_i \\ &\textit{subjected to}: \ \sigma_i \le \sigma_{\text{Max}} \quad \text{For all } i \in \{1,...,10\} \\ &|U_i| \le 2'' \qquad\qquad \text{For all } j \in \{1,...,6\} \end{aligned} \tag{3.8}$$

The EHS method is used to solve this problem. Whenever a harmony is improvised, finite element analyses are performed to calculate its objective function value $f(\boldsymbol{x}')$ and fitness value$F(\boldsymbol{x}')$ and compare them with worst harmony in the HM.

An improvised harmony can be either feasible ($f(\boldsymbol{x}')$ =1) or infeasible ($f(\boldsymbol{x}') < 1$). Since the starting HM is generated randomly, usually it contains feasible and infeasible solutions where each is associated to $f(\boldsymbol{x}_j)$ and $F(\boldsymbol{x}_j)$ values $1 \leq j \leq HMS$. The definition of worst harmony in constrained problem is different than that for unconstrained problems. In constrained problems, as long as not all the harmonies in HM are feasible, the worst harmony is the one that has the minimum fitness

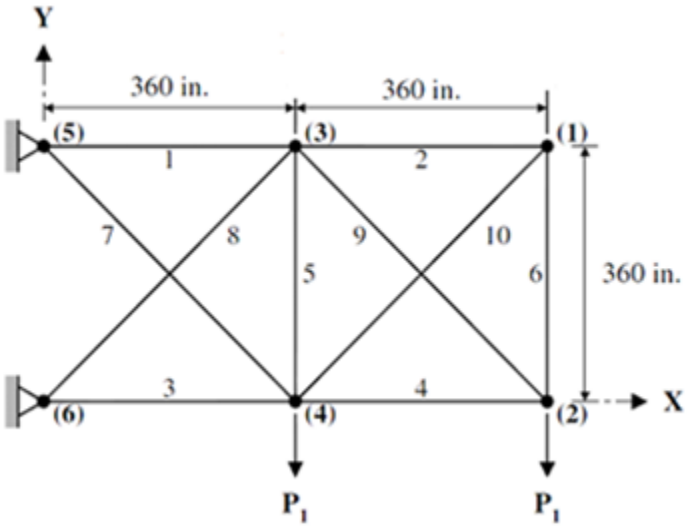


Figure 3.1: Ten-bar planar truss (case1) [14]

value $F(\boldsymbol{x}_{min})$ and if the improvised harmony $\boldsymbol{x}'$ has a higher fitness value it replaces the worst harmony in the HM. Once all harmonies in HM are feasible the problem become similar to unconstrained problems and worst harmony becomes the one that has the highest objective function value$f(\boldsymbol{x}_{max})$. Consequently, the harmony associated with $f(\boldsymbol{x}_{max})$ is replaced by the improvised harmony $\boldsymbol{x}'$ if it is feasible and has a lower

objective function value $f(\boldsymbol{x}') < f(\boldsymbol{x}_{max})$ **.** However, the improvised harmony is neglected if it is infeasible.

Table 3.1 shows the starting harmony (randomly generated) for this problem and their associated $f(\boldsymbol{x}_j)$ and $F(\boldsymbol{x}_j)$ values.

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	$f(\boldsymbol{x})$	$F(\boldsymbol{x})$
1	24.41	32.14	28.06	21.90	25.47	5.15	17.46	6.49	11.08	17.69	7.62	0.95
2	22.47	2.73	4.33	7.49	10.40	3.03	33.83	25.15	7.09	30.16	6.71	0.87
3	27.02	31.23	23.32	34.14	0.28	7.96	23.20	6.30	20.29	12.80	7.64	0.95
4	0.71	14.56	24.21	10.39	21.41	15.36	33.35	22.80	13.65	32.01	8.3	0.87
5	5.85	7.64	20.35	16.56	23.01	29.39	30.58	31.76	20.58	1.99	8.02	0.91
6	8.71	30.37	27.79	6.92	18.51	14.58	4.81	12.49	17.20	19.99	6.62	0.91
7	13.82	10.51	2.19	33.22	4.06	26.51	15.75	27.40	10.08	17.86	6.87	0.86
8	22.53	28.67	9.14	33.53	10.25	11.87	4.04	28.77	19.53	1.38	6.91	0.91
9	16.10	30.81	15.02	0.26	34.43	7.91	20.66	14.30	7.70	17.52	6.82	0.88
10	11.74	32.19	18.71	31.69	28.22	8.17	8.58	15.64	5.41	26.90	7.58	0.92
11	19.36	11.73	6.62	10.54	15.27	3.10	4.30	8.29	16.05	23.71	5.06	0.91
12	20.61	4.72	32.33	14.83	31.94	10.28	19.98	33.84	11.74	10.52	8.00	1.00
13	5.47	32.05	24.31	26.25	32.46	34.19	30.19	34.80	3.22	23.02	10.2	0.91
14	12.81	1.03	22.48	33.72	31.08	0.99	25.50	10.84	33.65	7.54	7.62	0.95
15	2.54	10.50	14.31	13.42	8.47	29.85	29.08	27.10	25.12	7.11	7.34	0.86
16	18.88	7.58	8.88	25.59	20.67	27.32	12.99	10.81	8.03	16.47	6.38	0.91
17	22.62	32.44	19.81	7.06	3.24	32.45	33.40	24.56	5.22	9.86	7.95	0.92
18	34.34	16.54	29.33	22.02	7.32	7.45	17.09	12.15	29.07	25.96	8.5	1.00
19	30.90	26.53	20.75	13.64	23.41	2.20	7.51	0.41	29.86	7.15	6.51	0.80
20	18.09	7.99	25.00	12.65	26.73	7.35	15.05	19.16	23.78	29.35	7.96	0.95

Table 3.1: Starting HM for the Ten-bar planar truss

Since the starting HM contains infeasible solution the worst harmony is the one with the lowest fitness value which is the 19^{th} harmony with $F(\boldsymbol{x}_{19}) = 0.8$. After the first iteration the following harmony was improvised

$\boldsymbol{x}'$ = (14.06, 10.51, 18.95, 12.22, 10.40, 27.32, 33.94, 23.85 , 8.54, 11.60)

with a corresponding fitness value $F(\boldsymbol{x}) = 0.91$. Consequently, $\boldsymbol{x}'$ replaces $\boldsymbol{x_{19}}$ in HM due its larger fitness value. Table 3.2 shows the updated HM

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	$f(\boldsymbol{x})$	$F(\boldsymbol{x})$
1	24.41	32.14	28.06	21.90	25.47	5.15	17.46	6.49	11.08	17.69	7.62	0.95
2	22.47	2.73	4.33	7.49	10.40	3.03	33.83	25.15	7.09	30.16	6.71	0.87
3	27.02	31.23	23.32	34.14	0.28	7.96	23.20	6.30	20.29	12.80	7.64	0.95
4	0.71	14.56	24.21	10.39	21.41	15.36	33.35	22.80	13.65	32.01	8.3	0.87
5	5.85	7.64	20.35	16.56	23.01	29.39	30.58	31.76	20.58	1.99	8.02	0.91
6	8.71	30.37	27.79	6.92	18.51	14.58	4.81	12.49	17.20	19.99	6.62	0.91
7	13.82	10.51	2.19	33.22	4.06	26.51	15.75	27.40	10.08	17.86	6.87	0.86
8	22.53	28.67	9.14	33.53	10.25	11.87	4.04	28.77	19.53	1.38	6.91	0.91
9	16.10	30.81	15.02	0.26	34.43	7.91	20.66	14.30	7.70	17.52	6.82	0.88
10	11.74	32.19	18.71	31.69	28.22	8.17	8.58	15.64	5.41	26.90	7.58	0.92
11	19.36	11.73	6.62	10.54	15.27	3.10	4.30	8.29	16.05	23.71	5.06	0.91
12	20.61	4.72	32.33	14.83	31.94	10.28	19.98	33.84	11.74	10.52	8.00	1.00
13	5.47	32.05	24.31	26.25	32.46	34.19	30.19	34.80	3.22	23.02	10.2	0.91
14	12.81	1.03	22.48	33.72	31.08	0.99	25.50	10.84	33.65	7.54	7.62	0.95
15	2.54	10.50	14.31	13.42	8.47	29.85	29.08	27.10	25.12	7.11	7.34	0.86
16	18.88	7.58	8.88	25.59	20.67	27.32	12.99	10.81	8.03	16.47	6.38	0.91
17	22.62	32.44	19.81	7.06	3.24	32.45	33.40	24.56	5.22	9.86	7.95	0.92
18	34.34	16.54	29.33	22.02	7.32	7.45	17.09	12.15	29.07	25.96	8.5	1.00
19	14.06	10.51	18.95	12.22	10.40	27.32	33.94	23.85	8.54	11.60	7.59	0.91
20	18.09	7.99	25.00	12.65	26.73	7.35	15.05	19.16	23.78	29.35	7.96	0.95

Table 3.2: Updated HM for the Ten-bar planar truss after the first iteration.

Table 3.3 shows the updated HM after performing 155 iterations where the fitness value of all harmonies is equal to 1 ($F(\boldsymbol{x_j}) = 1\ ;\ 1 \leq j \leq HMS$). Which means all harmonies in HM are feasible.

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	$f(\boldsymbol{x})$	$F(\boldsymbol{x})$
1	18.49	34.98	23.47	33.46	15.08	11.60	21.04	9.21	33.65	26.85	9.56	1.00
2	19.31	34.98	31.41	21.86	8.06	24.53	13.97	35.00	12.94	10.57	8.74	1.00
3	19.57	5.22	29.33	32.75	0.10	9.65	24.98	31.02	22.32	0.56	7.49	1.00
4	20.61	27.10	31.41	12.08	3.24	20.08	17.47	13.18	29.86	8.69	7.65	1.00
5	24.66	17.63	33.00	26.17	10.28	9.74	14.06	28.02	11.08	8.69	7.52	1.00
6	34.34	12.52	32.33	4.66	13.40	3.50	31.89	17.55	33.61	11.27	8.43	1.00
7	34.34	4.72	31.41	15.62	7.32	3.47	34.82	10.29	32.78	8.69	7.9	1.00
8	21.03	23.76	31.41	14.61	0.10	7.45	17.00	31.02	33.65	26.85	9.07	1.00
9	25.29	10.19	32.33	21.86	31.94	10.57	14.06	23.12	33.61	16.13	9.18	1.00
10	32.84	4.72	20.30	15.42	13.40	4.65	35.00	15.21	33.65	9.41	8.04	1.00
11	27.02	1.99	31.51	33.46	13.40	5.15	24.98	19.16	25.08	19.82	8.58	1.00
12	20.61	4.72	32.33	14.83	31.94	10.28	19.98	33.84	11.74	10.52	8.00	1.00
13	27.02	15.62	21.91	4.11	13.40	21.04	24.98	23.12	29.86	9.76	8.18	1.00
14	34.34	31.23	31.41	14.06	1.60	5.15	13.27	9.21	33.65	17.35	7.98	1.00
15	27.02	14.08	31.51	24.69	5.72	10.16	24.53	31.02	12.94	11.27	8.14	1.00
16	34.34	27.40	26.50	21.90	5.72	5.15	35.00	28.02	13.59	17.86	9.17	1.00
17	34.34	34.55	29.33	33.46	5.72	0.99	13.70	19.50	15.03	6.87	7.79	1.00
18	34.34	16.54	29.33	22.02	7.32	7.45	17.09	12.15	29.07	25.96	8.5	1.00
19	25.20	33.01	24.77	6.21	15.08	7.45	34.82	19.50	33.65	13.96	9.21	1.00
20	27.02	15.62	27.09	21.86	5.72	0.49	7.03	28.87	22.32	8.22	6.9	1.00

Table 3.3: Updated HM for the Ten-bar planar truss after 155 iterations, where all harmonies are feasible.

Since all harmonies are feasible the worst harmony is the one with the highest objective function which is the first harmony in HM ($\boldsymbol{x}_1$) with a corresponding objective function value $f(\boldsymbol{x}_{max}) = 9.56$.

For the 160^{th} iteration the following feasible harmony was improvised

$\boldsymbol{x}'$=(25.2, 4.72, 18.71, 32.75, 9.97, 10.28, 19.8, 26.6, 14.61) with a corresponding objective function value ($f(x) = 7.72$). Since $\boldsymbol{x}'$ is feasible and has a better objective value it replaces the first harmony in HM.

The optimization process continues until the specified maximum number of iterations is performed. The results obtained using the EHS algorithm shown in table 3.4 are very close to the results obtained by K.S.Lee[14] who obtained an objective function value of 5.057 kip using the same algorithm. Figure 3.1 shows the change in the best objective function value with the number of iterations.

Design variable	Design variable x_i value in^2
x_1	30.87
x_2	0.1
x_3	23.47
x_4	14.51
x_5	0.1
x_6	0.53
x_7	7.5
x_8	21.17
x_9	21.01
x_{10}	0.1
$f(x)$	5.07 kips

Table 3.4: Best solution for the ten-bar planar truss example

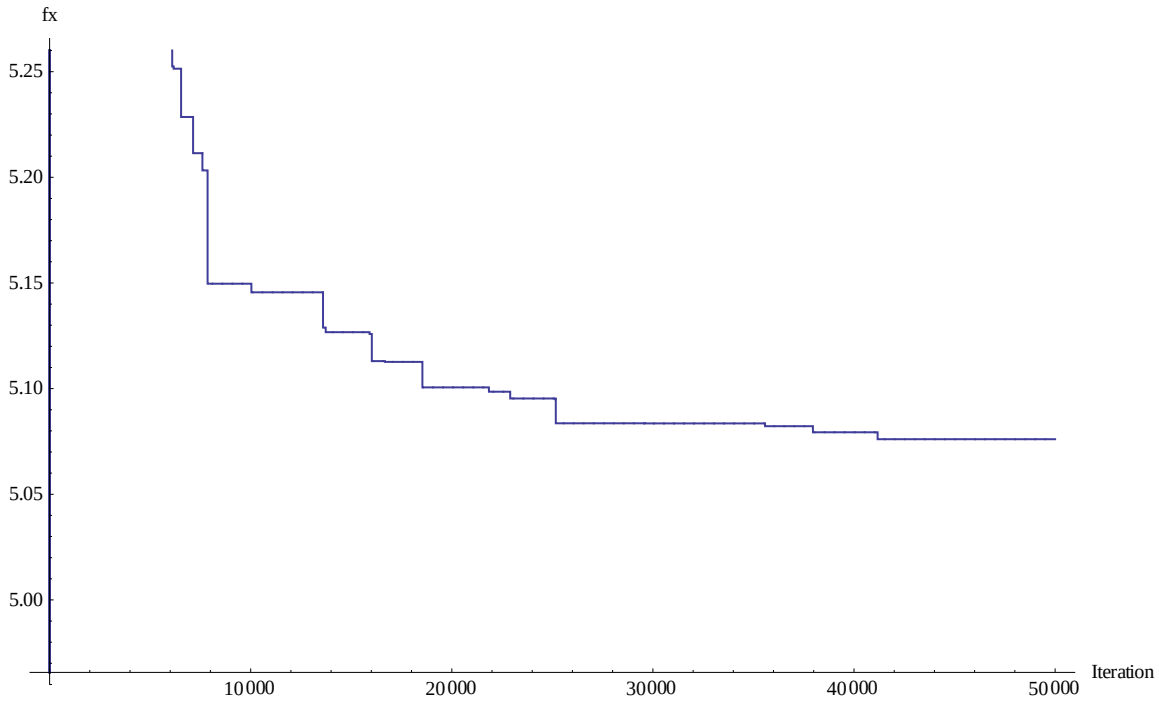


Figure 3.2: History for solution of the 10-bar Planar truss problem

3.2.1.2 Eighteen-bar Planner Truss

The eighteen-bar planar truss shown in Figure 3.3 is to be size optimized, P is 20 kip, $E = 1 \times 10^4 ksi$, $\rho = 0.1 lb/in^3$, $|\sigma_{max}| = 20\ Ksi$, Euler buckling compressive stress limitations were imposed for truss and expressed as follows : $\sigma_{bi} = -kEA_i / L_i^2$

where k is set to 4 in this example. EHS parameters were set as follows: HMCR = 0.8, PAR = 0.3 , maximum number of iterations= 10,000 The minimum cross sectional area $A_{min} = 0.1\ in^2$. In this problem the structural members where linked in four groups as follows:

i. $A_1 = A_4 = A_8 = A_{12} = A_{16}$
ii. $A_2 = A_6 = A_{10} = A_{14} = A_{18}$
iii. $A_3 = A_7 = A_{11} = A_{15}$
iv. $A_5 = A_9 = A_{13} = A_{17}$

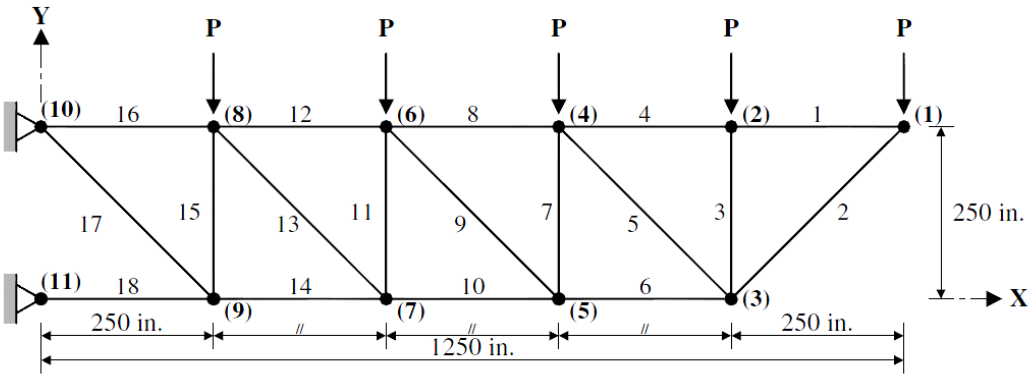


Figure 3.3: Eighteen-bar truss.

Group i	Group ii	Group iii	Group iv	F(x)
$10.00in^2$	21.65 in^2	12.54 in^2	7.098 in^2	6.43

Table 3.5: Eighteen bar planar truss results.

CHAPTER 4

HARMONY SEARCH FOR TRUSS TOPOLOGY OPTIMIZATION

4.1 Introduction

In the previous chapters detailed explanations of the standard harmony search were introduced followed by the explorative harmony search. These methods were used to solve numerical and structural truss size optimization problems. In this chapter, the harmony search is extended to solve binary discrete truss topology optimization problems.

Since it was developed in (2001) HS has been successfully used in many works to solve discrete and continuous optimization problems. Geem[24] applied the HS algorithm to the optimal water pump switching problem (which is a binary problem where the status of the pump is represented either by 0 or 1) and it yielded better results than those obtained using GA. However, in 2009 Greblicki [25] noted inefficiency of the HS algorithm for large binary problems ($n \geq 200$ design variables). Although many variants have been proposed to improve the harmony search ability for optimizing general continuous and discrete problems, few works have been devoted to improving its performance in solving binary-coded problems. Nevertheless in 2010 Wang et al. [26] pointed out that the pitch adjustment strategy used in standard harmony search explains the poor performance of HS for binary problems. Thus, a new pitch adjustment strategy was proposed and it is used throughout this work. After Introducing the discrete binary harmony search algorithm (DBHS) method Wang et al.[27] developed the multi-objective binary harmony search algorithm (MBHS) to handle binary multi-objective optimization problems. In 2013 the adaptive binary harmony search (ABHS) was also proposed by Wang et al.[28] to solve binary coded problems more efficiently. In the former method a new harmony memory consideration strategy was proposed with new expression for the HMCR parameter that made it increase linearly with iteration counts.

This chapter is organized as follows. Section 1 gives a brief overview of discrete topology optimization. In section2, modifications and enhancements to the harmony search that make it suitable for topology optimization are introduced and explained in detail. Finally in section 3 some truss topology benchmark examples that are frequently solved in the literature are presented and solved.

4.2 Topology Optimization

4.2.1 Discrete and Continuum Topology Optimization

Structure topology optimization is the mathematical approach which determines the number, location and shape of the holes within a specified region and the connectedness of the domain in order to minimize or maximize a given criteria (objective function). "The selection of optimal topology is arguably among the most structural optimization problems" [29], since only few quantities are known in advance such as the possible boundary conditions and some constraints on the volume of the structure.

In general there are two main techniques to optimize the topology of a structure and these are:

1. Discrete optimization method.
2. Continuum optimization method.

In the first technique the structure is modeled with a finite number of structural members (truss, frames, beams) and nodes. In the second the structure is modeled as continuum. A comparison between the two techniques is discussed in detail by Rozvany [8] and summarized in table 4.1:

Discrete optimization technique	Continuum optimization technique
Problem is solved using numerical methods iteratively.	Problem is solved using analytical methods, where all equations are solved simultaneously.
The structure is modeled using finite number of structural elements	The structure is modeled using a large number of continuum elements
The minimum prescribed cross sectional are is small or zero	Design variables are typically solid volume fractions.

Table 4.1: Comparison between discrete and continuum structural topology optimization techniques.[following Rozvany].

Since the aim of this work is to use the heuristic harmony search method, the reminder of this chapter will focus on discrete topology optimization of trusses.

4.2.2 Truss Topology Optimization

4.2.2.1 Problem Formulation

Most of the truss topology optimization works in the literature are based on the ground structure concept introduced by Dorn et al in 1964 [30]. A ground structure is a highly connected structure where each node is connected to all other nodes, as shown in figure 4.1. During the optimization process, members that have zero cross sectional area are eliminated from the ground structure in order to reach the best substructure that satisfies the load and support conditions.

Truss topology optimization problem can be formulated in many different ways. A book by Rozvany [9] illustrates in details the techniques of transforming different formulations into equivalent problems. In this work we focus only on the problem of minimizing the compliance (external work) which considered one of the basic

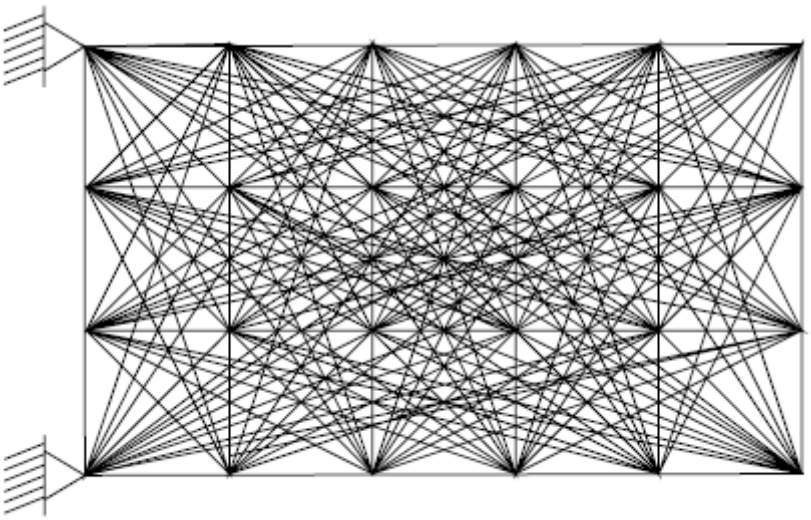


Figure 4.1: Ground structure

Formulations that can be optimized using the harmony search method. The minimum compliance problem is stated as follows:

$$\begin{aligned} \min_{\mathbf{x}} \quad & \frac{1}{2}\mathbf{f}^{\mathrm{T}}\mathbf{u} \\ \text{such that} \quad & \mathbf{K(x)}\,\mathbf{u} = \mathbf{f}, \\ & \mathbf{x} \in X_{\mathrm{V}} \end{aligned} \tag{4.1}$$

Where $\boldsymbol{f}^T = \{f_1, f_2, \dots, f_n\}$ is the external force vector applied to the structural nodes, $\boldsymbol{u} = \{u_1, u_2, \dots, u_n\}$ is the corresponding nodal displacement vector (state variable vector) under load $\boldsymbol{f}$; and term $\boldsymbol{f}^T.\boldsymbol{u}$ represents the structural compliance. Here n denotes the number of degrees of freedom of the truss model after eliminating the fixed nodal degrees of freedom. The constraint in Eq (4.1) requires the structure to be in equilibrium where $\boldsymbol{K}$

is the structural stiffness matrix which results from the assembly of all member level stiffness matrices

$$\mathbf{K}(\mathbf{x}) = \sum_{i=1}^{m} x_i \, \mathbf{K}_i \tag{4.2}$$

x_i Represent the cross sectional area, $\boldsymbol{x} = \{x_1, x_2, \ldots, x_m\}$ is the vector of design variables in which $x_i \geq 0$ is the ith element volume (cross sectional area × length), and m is the number of elements.

$\boldsymbol{K_i}$ is the ith element stiffness square matrix and has a size of(4×4) or(6×6) for planar and space trusses respectively, and is given as follows:

$$\mathbf{K}_i = \frac{E_i}{l_i} \boldsymbol{\gamma}_i \, \boldsymbol{\gamma}_i^T \tag{4.3}$$

In Eq (4.3) $\boldsymbol{\gamma}_i$ is the 6×1 (space truss) or 4×1 (planar truss) directional cosines between the members and coordinate axes.

4.2.2.2 Solving the minimum compliance optimization problem

Through the years minimum compliance truss topology optimization problems and their equivalent formulations have been solved using a multitude of procedures such as linear programing, nonlinear programing, dynamic programing and heuristic methods. Reviews by Topping [31,32] and Rozvany [8] are usually cited as key references ponding insight into the different optimization methods. In recent years heuristic methods have been used extensively to tackle truss topology optimization problems. Hajela and Lee [29] used genetic algorithm to obtain near optimal solutions by employing a two-step algorithm that generated a number of low weight stable solutions in the first step, and minimized the weight in the second step. Simulated annealing (SA) methods have also been used in some works such as a that Hasancebi and Erbatur [33] where SA was used to simultaneously optimize truss structures with respect to size, shape and topology

design variables. Although heuristic search methods are employed in numerous studies, it is noticed that such method have been applied almost exclusively to fairly small scale problems [34]. Thus the aim in this chapter is to study the efficiency of the HS method in tackling truss topology optimization problems.

The original ground structure approach can be considered as a size optimization problem where the member cross sectional area can vanish thereby removing the associated members from the ground structure. The process of removing the members with vanishing cross-sections may in most cases result in unstable structure (singular stiffness matrix) due to the existence of some nodes that are not connected to any elements or the existence of two collinear elements that are connected by an internal hinge [35]. Setting the lower bound on the cross sectional area to a very small value ϵ is proposed in the literature to avoid obtaining unstable trusses during the optimizing process; Nevertheless, this doesn't solve the problem of obtaining unstable structure eventually after dropping all elements with infinitesimal cross sectional area. Figures 4.1 and 4.2 present an example that further explains the previous idea [36]. The truss in Figure 4.2 was size optimized by setting the lower cross sectional area bound to 0.001 and all elements that have an area of 0.001 were dropped at the end of the optimization process which led to an unstable optimal solution as shown in the figure 4.2.

Since in this work we aim to obtain stable optimal truss designs, the cross sectional areas are allowed to vanish hence dropping members from the ground structure. Some of the resulting instability are tackled as follows:

1. If there is a node that is not connected to any element then this node is dropped.
2. If a node (hinge) connects only 2 collinear elements then the hinge is dropped and the two elements are merged together to form one continuous element.

For each candidate solution after dropping the vanishing members, merging collinear elements that are connected with one node, and dropping unconnected nodes the algorithms define a new finite element model with the real number of nodes and elements to be further analyzed for stability as explained in later sections.

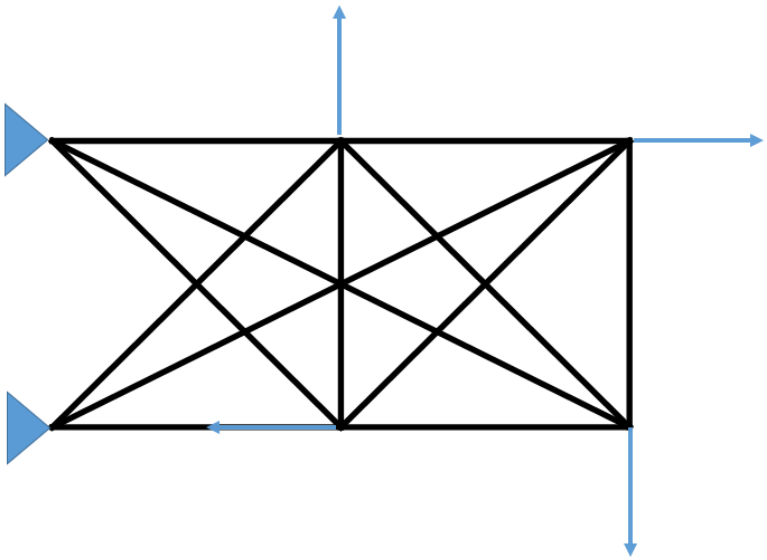


Figure 4.2: A ground structure of truss to be topology optimized

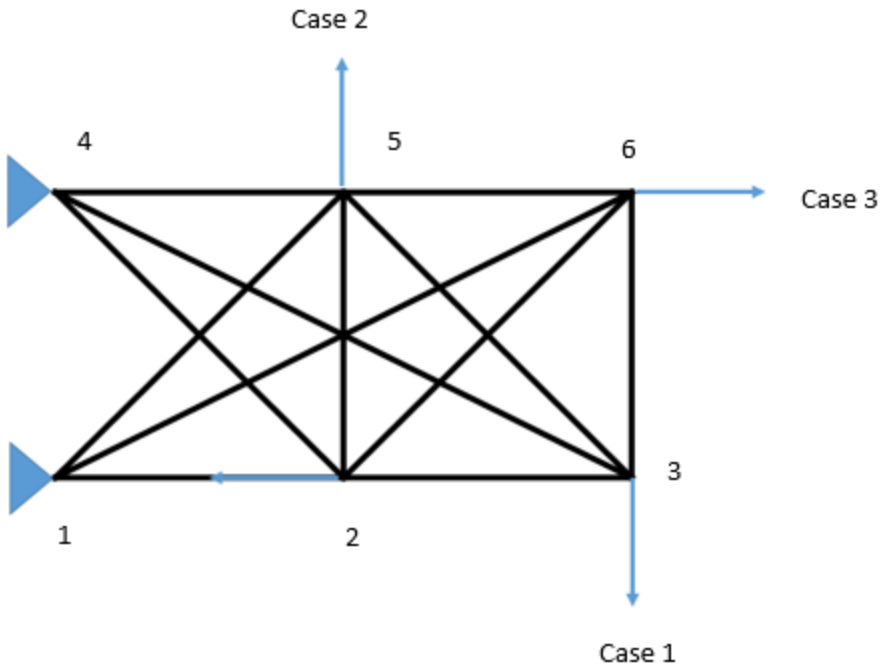


Figure 4.3: Topologically optimized truss after dropping members with vanishing cross-sections. (Stable under Case 1 + Case 2, Stable under Case 1 + Case 2 + Case 3 , otherwise unstable)

In this work Problem 4.1 is redefined in a very similar way as shown in 4.4

$$\begin{aligned} &\textit{Minimize} && \frac{1}{2}\mathbf{f}^{\mathrm{T}}\mathbf{u} \\ &\text{s.t} && \mathbf{K(x)\,u} = \mathbf{f}, \\ & && \mathrm{V}(x) - c\ \ \mathrm{V}_{Max} \le 0 \end{aligned} \tag{4.4}$$

$$\text{Where : } \mathrm{V}_{Max} = \sum_{i=1}^{m} x_i l_i \tag{4.5}$$

The only difference between 4.1 and 4.4 is volume constraint, where $V(x)$ the volume of the structure after dropping zero cross sectional area members, V_{Max} is the

volume of the ground structure and given in equation (4.5), c dictates the portion of the ground structure's material that should be included in the optimal structure with $0 < c \leq 1$.

4.3 Harmony Search Modifications to Suit Topology Optimization

4.3.1 Pitch Adjustment

The role of the pitch adjustment operator is to search for a better local optimum. In standard harmony search the adjustment operator for binary optimization problems is given as follows:

$$\overline{x_j} = \begin{cases} 1 & \text{if} \quad h_{ij} = 0 \\ 0 & \text{if} \quad h_{ij} = 1 \end{cases} \tag{4.6}$$

Where $\overline{x_j}$ is the j^{th}element in the new improvised harmony that is pitch adjusted; h_{ij} is the j^{th}element of the i^{th}harmony in the harmony memory.

As mentioned earlier Wang [26] pointed out that the pitch adjustment rule utilized in the standard HS is not a good choice if a better convergence performance is to be attained for binary problems. So he proposed a new pitch adjustment rule and it is given as follows:

$$x'_j = \begin{cases} h_{bj} & r < PAR \\ x_j & else \end{cases} \tag{4.7}$$

Where x'_j is the j^{th}element of the new improvised harmony $\boldsymbol{x}'$; r is a random number between [0, 1]; h_{bj} is the j^{th}element of the best harmony (row b in the harmony memory). More details are provided about picking the best harmony in latter sections.

4.3.2 Harmony Fitness and HM Updating Strategy

As it is explained in the Chapter 2 the efficiency and robustness of any heuristic search method depends on its explorative and exploitative powers. Explorative power can be increased by making the algorithm search new regions in the design space. In HS, this is achieved when a new harmony component is generated randomly from the allowed range.

$$x_i' \in \mathbf{X}_i \qquad \text{if} \qquad r > HMCR \qquad X_L < X_i < X_U \tag{4.8}$$

$$x' = {}_L x_i + ran\,[0,1] \cdot \left({}_U x_i - {}_L x_i \right) \tag{4.9}$$

where $X_L \; and \; X_U$ are the lower and upper limits for the i^{th} design variable. On the other hand, exploitative ability is achieved by using the search history information stored in the HM and leads the HS to reach the optimum solution. Obviously, the better the information that is stored in the HM the more efficient the HS will be.

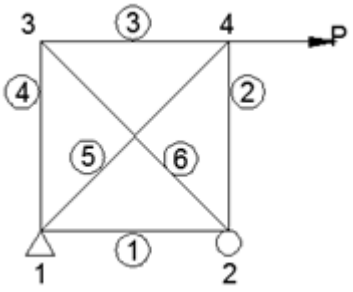
In the case of unconstrained problems an improvised solution that yields a better objective function value than the worst one stored in the HM replaces it. In constrained optimization problems such as the examples solved in the previous chapter, a fitness value that ranges between [0,1] based on the degree of constraints violation is evaluated and assigned to each improvised solution . In discrete truss topology optimization , the two primary constraints are that the designs be stable and that the amount of structural material used doesn't exceed the allowable amount.

In truss topology optimization problems each improvised structural design can be one of the following four options:

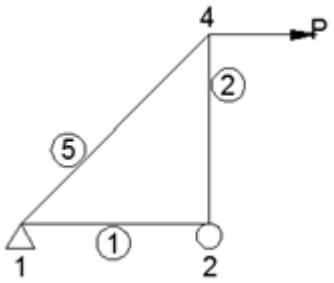
- Unstable solution, satisfies the volume constraints.
- Unstable solution , violates the volume constraints.

- Stable solution, but violates the volume constraint.
- Stable solutions and satisfies the volume constraint.

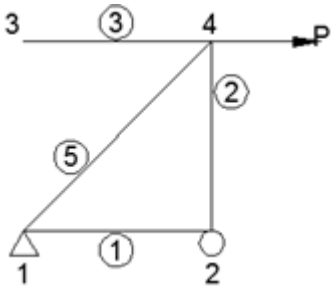
Although information from unstable solutions could in principle be useful to guide the algorithm to the optimum point, there is no easy way to determine the fitness of unstable designs and how to prefer one over another. Figure 4.4 further explains the previous point. Figure 3.4 a shows a simple ground structure supporting a load at node 4, and the optimum topology for the given load case is shown in 3.4 b. Figure 3.4 c shows an unstable design that improvised during the optimization process that is also very close to the optimum. However, if this solutions were to be stored in the HM to use its information in the searching process another unstable topology might be generated such as the ones in d and e. The question here is which solution is better? If this question is not answered the harmony search won't know which harmony is the worst in the HM to replace it with a better one. Since including unstable topologies in the HM gives rise to difficulty in comparing the fitness of two solutions, unstable solutions must be excluded from the HM this work. Consequently, all unstable solution are assigned a negative fitness value which are picked to be (-1) in this work.



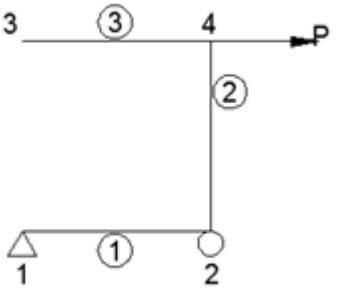
a) Ground Structure



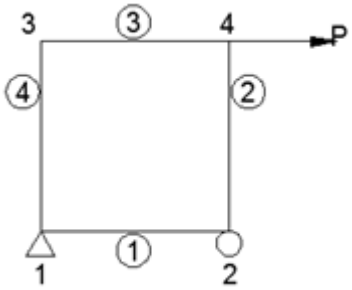
b) stable



c) unstable



d) unstable



e) unstable

Figure 4.4: Stable and unstable designs

Stable solution fitness values range between [0, 1] based on the volume constraint violation. A stable solution fitness value is given as follows:

$$F_j(x) = \begin{cases} 1 & \text{if } g_v(x) \le 0 \\ 1 - \dfrac{\mathrm{V}(x)/\mathrm{V}_{Max}}{(1-c)\cdot \mathrm{V}_{Max}} & \text{if } g_v(x) > 0 \end{cases} \tag{4.10}$$

$$g_v(x) = \mathrm{V}(x) - c \;\; \mathrm{V}_{Max} \tag{4.11}$$

$V(x)$ is the volume of the structure represented by harmony and $g_v(x)$ is the volume constraint.

Stable solutions that violate volume constraints are also penalized by assigning a very high compliance value to them. This value is calculated approximately by assuming that all the members in the ground structure have a very small area which is assumed to be (10^{-4}) in this work. Thus for harmony $\boldsymbol{x}$ the compliance value can be calculated as follows:

$$f(x) = \begin{cases} \dfrac{1}{2}\mathbf{f}^{\mathrm{T}}\mathbf{u}, & g_v(x) \le 0 \\ \dfrac{1}{2}\mathbf{f}^{\mathrm{T}}\mathbf{u}_{\max}, & g_v(x) > 0 \end{cases} \tag{4.12}$$

$\boldsymbol{u}_{max}$ is an approximate maximum displacement vector (maximum state variables) obtained when an area of (10^{-17}) is assigned to all ground structure's elements.

The fitness and compliance values for all harmonies in the harmony memory should be calculated, and the worst harmony should be determined. The improvised harmony's fitness and compliance values should also be calculated and compared with the worst harmony in the HM to check if it should be replaced. In order to compare the different harmonies a constraint handling method proposed by Deb. [37] in 2000 is used

and extended in this work. The method uses the following criteria to prefer one solution on another:

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

In this work the aforementioned criteria is applied by considering stable solutions as feasible solution and among two stable solutions one having smaller volume constraint violations (has a higher fitness value) is a more preferable solution. Moreover among two solutions have the same fitness value the one with lower compliance value is considered a better solution. When all the harmonies in the HM have a fitness value of 1 (volume constraint is satisfied for all solutions) the compliance value becomes the only criteria to evaluate which solution is better.

4.3.3 Finite Element Analysis

As it has been explained in the previous sections stable solutions with good fitness and compliance values are considered and stored in the HM while unstable solutions are disregarded. Since finite element analyses have a high cost and consume a lot of time many strategies have been taken in this work to reduce the number of analysis and can be summarized as follows:

1. Perform analysis on stable solutions only.

The first simple test can be performed to check the instability of a harmony is comparing the number of available equations and essential boundary conditions (reactions) with the number of nodes as follows:

$$r + m < 2j \tag{4.13}$$

where r is the number of reactions , m is the number of members for a given solution, and j is the number of joints. If Eq 4.13 is satisfied then the structure is unstable and a fitness value of (-1) is assigned to the solution.

If Eq 4.13 is not satisfied, in other words $r + m \geq 2j$, the solution is qualified for second stability test which is calculating the determinant of the stiffness matrix. If the stiffness matrix determinant equals to zero, then the solution is unstable and disregarded, otherwise, the solution is stable and may be qualified for farther finite element analysis.

2. Stable solutions with acceptable range of volume constraint are qualified for finite element analysis.

Although stable solutions with large constraint violations are considered in the HM since they have useful information that lead the algorithm to the optimum point, the high cost needed to perform analysis makes it inefficient to analyze all stable solutions. In order to make the algorithm faster a parameter α is introduced and is given as follows:

$$\alpha \leq \alpha_{\max} \tag{4.14}$$

$$\alpha = \frac{Vol(x)}{Vol_{Max}} \tag{4.15}$$

Where α_{max} is a user predefined constant. If α_{max} is set to be 0.4 for example, only stable solutions that have 40% and less of the original ground structure's material are qualified for further analysis. Picking a value of α_{max} depends on the value of the constant c in the volume constraint expression shown in 3.11. If the value of c is set to be 0.1 for example a value ranges between 0.4 to 0.6 may be suitable for α_{max}. For solution where $\alpha > \alpha_{max}$ a fitness value is calculated as in equations 3.10, however this solution is penalized by setting its compliance value to be the approximate maximum compliance value as in equation 4.12.

Figure 4.5 summarize the strategies applied in this work to reduce the number of costly analysis.

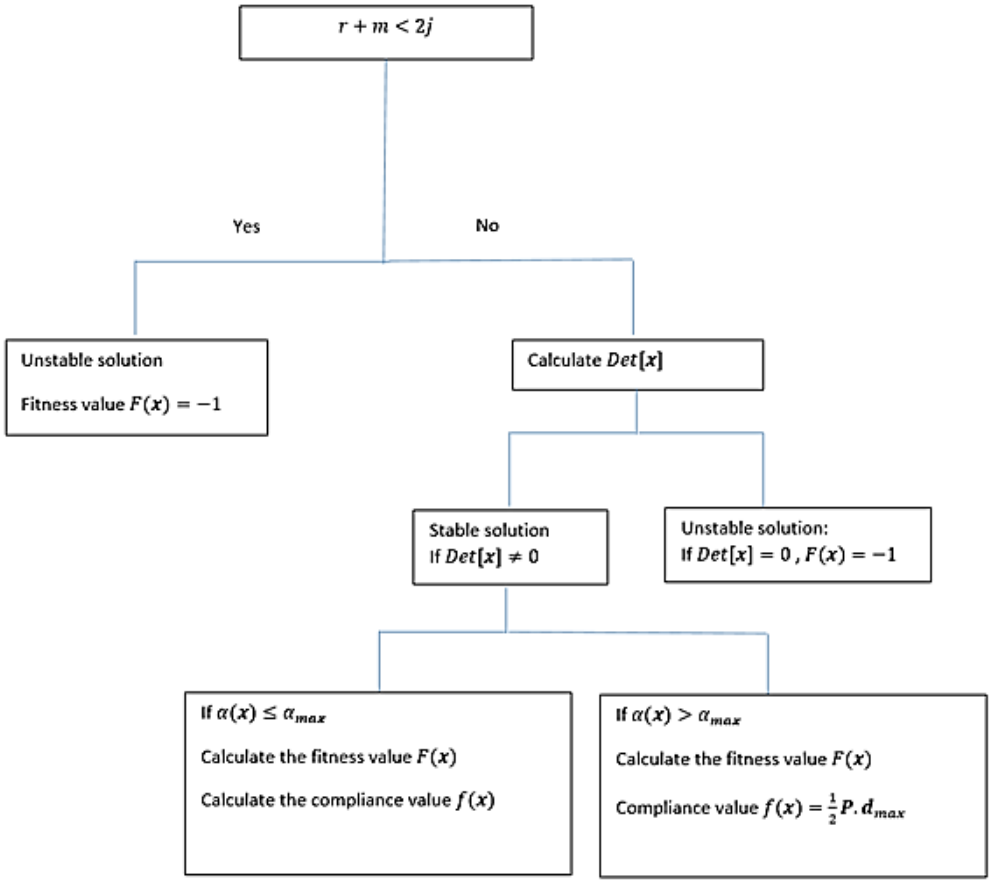


Figure 4.5: Strategies to reduce the number of analysis

4.4 Numerical Examples

A few truss topology examples are presented in this section, for all examples the harmony search parameters are set as follows:

Harmony memory consideration ratio HMCR= 0.7 ; harmony size HS =20 ; pitch adjustment ratio PAR ranges between [0.1 ,0.3] depending on the problem , the volume constraint constant c and the α_{max} values are specified for each problem.

4.4.1 12-Element Ground Structure Truss

The boundary conditions and the loading are given in the figure 4.6 for the 6 nodes 12 elements truss. In this example the c value is set to 0.6 and $\alpha_{max} = 1$. In this example none of the nodes can be dropped since there are boundary conditions at all nodes, also this example considered easy since no two elements needs to be merged into one element. The goal of this example is to show that the algorithm is capable of finding many optimal solutions that enable the designer to compare them and pick the most suitable one since in some cases the constructability of optimal solution might be hard or expensive. Figure 4.7 shows the different optimal topologies obtained by using the proposed algorithm [7].

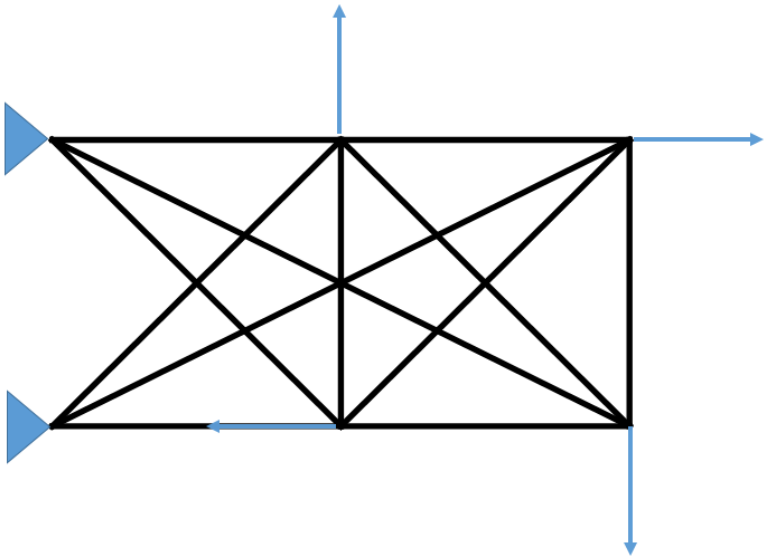


Figure 4.6: 12-Element ground structure for example 4.4.1 [36]

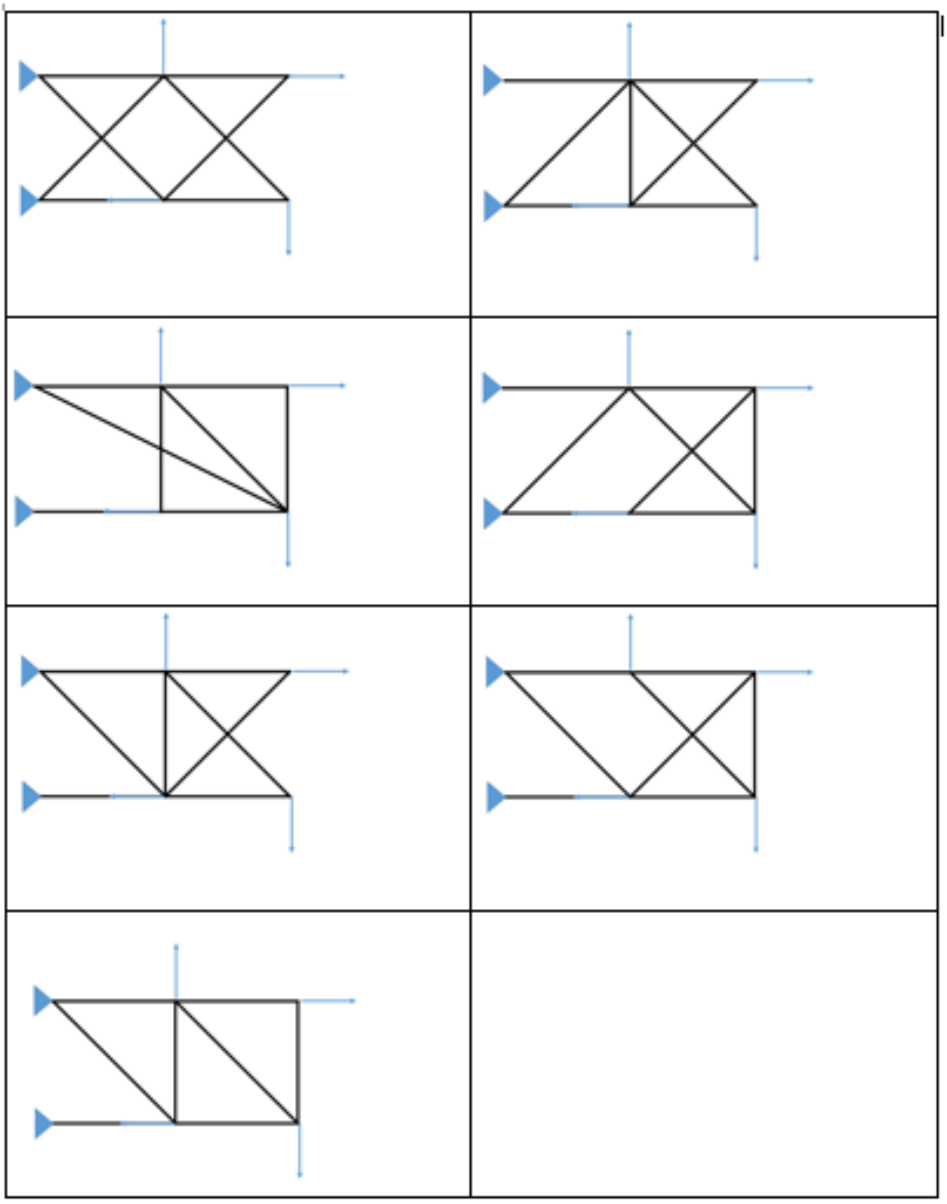


Figure 4.7: Different optimal topologies obtained for example 4.4.1

4.4.2 11-Elemnt Ground Structure Truss

The 6 nodes 11 elements truss shown in Figure 4.8 is presented frequently in the literature. The c and α_{max} parameters are set to be 0.6 and 1 respectively. The optimum topology shown in Figure 4.9.

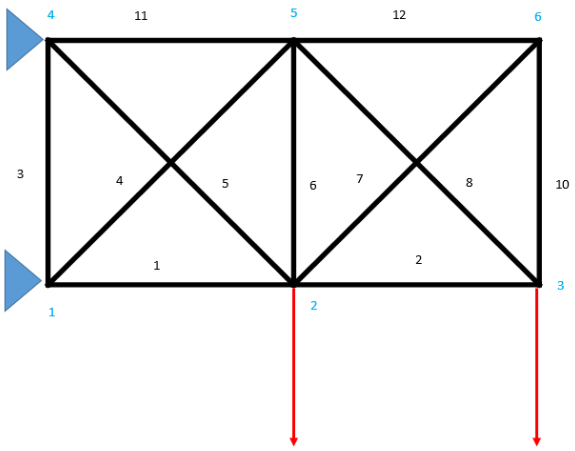


Figure 4.8: Ground structure for example 4.4.2

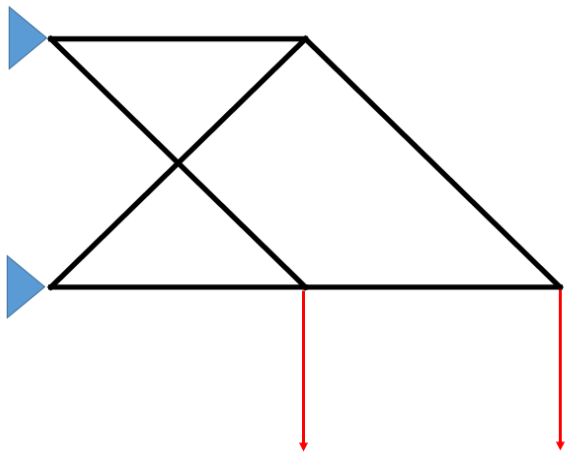


Figure 4.9: Optimal topology for example 4.4.2

4.4.3 42-Element Ground Structure truss

The 42-element truss shown in Figure 4.10 presents the algorithm's capability of merging collinear elements into one element. In this example the c and α_{max} parameters are set to be 0.2 and 0.4 respectively.

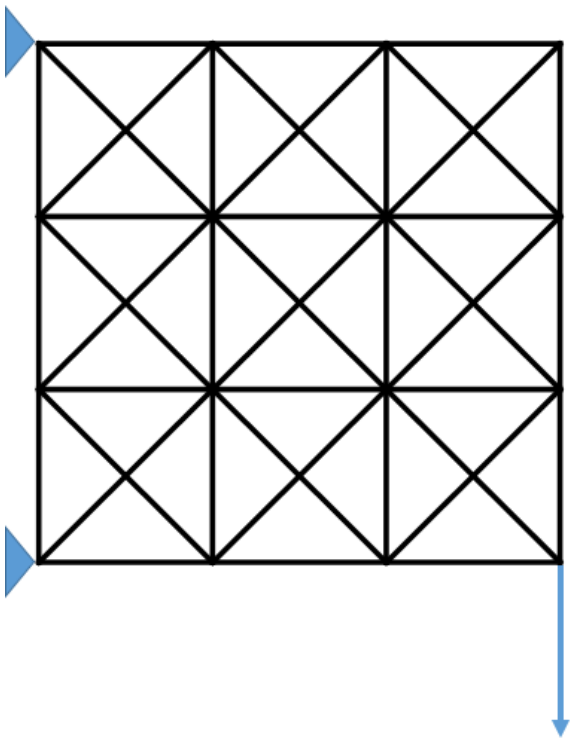


Figure 4.10: 42-Element ground structure for example 4.4.3

The optimal topology is shown in Figure 4.11, where three elements are merged in one element and all other unnecessary members and nodes are dropped.

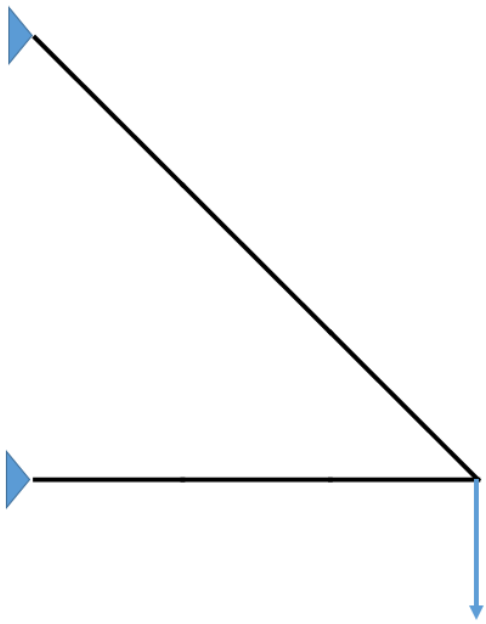


Figure 4.11: Optimal topology for truss in example 4.4.3

4.5 Conclusion

In this chapter many modifications have been made to the classical harmony search method to suit binary truss topology optimization problems. In this work the proposed methodology has been implemented on few benchmark examples and the results were obtained with a small number of iterations compared to the total number of solution.

Although the proposed method has been implemented and yielded optimal or near optimal results for most of the benchmark problems in the literature, its performance declines when the number of design variables increases. That is because the number of iterations required to find one stable (usually infeasible) solution exponentially increases when the number of design variables increases. Figure 4.12 shows the relationship between the number of design variables of the ground structure and the number of iterations required to find one stable configuration with less volume than Vol_{Max}.

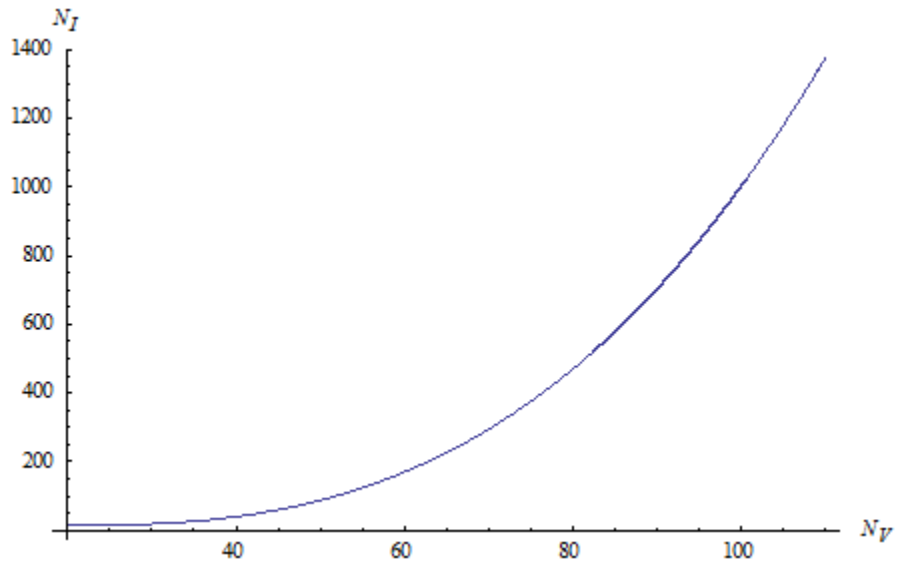


Figure 4.12: Relationship between the number of design variables and the number of iterations to find one stable solution.

REFERENCES

1. Arora, J. (2004). *Introduction to optimum design* Academic Press.
2. Klarbring, A. (2008). *An introduction to structural optimization* (Vol. 153). Springer.
3. Dowsland, K.A. 1995. Simulated Annealing. In Modern Heuristic Techniques for Combinatorial Problems (ed. Reeves, C.R.), McGraw-Hill, 1995.
4. Kirkpatrick, S., Gelatt, C., and Vecchi, M. "Optimization by Simulated Annealing." Science, Vol. 220, No. 4598, pp 671- 680, 1983.
5. Scientific American, 1992, Vol.267(1), pp.66-72 [Peer Reviewed Journal].
6. BEASLEY, D., BULL, D., & MARTIN, R. (1993). An overview of genetic algorithms .1. fundamentals. *University Computing*, *15*(2), 58-69.
7. Geem, Z., Kim, J., & Loganathan, G. (2001). A new heuristic optimization algorithm: Harmony search. *Simulation*, *76*(2), 60-68.
8. Rozvany, G. I. (Ed.). (1997). *Topology optimization in structural mechanics* (No. 374). Springer.
9. Lee, K. S., & Geem, Z. W. (2004). A new structural optimization method based on the harmony search algorithm. *Computers & Structures*, *82*(9), 781-798.
10. Lee, K. S., Geem, Z. W., Lee, S. H., & Bae, K. W. (2005). The harmony search heuristic algorithm for discrete structural optimization. *Engineering Optimization*, *37*(7), 663-684.
11. Degertekin, S. O. (2008). Optimum design of steel frames using harmony search algorithm. *Structural and Multidisciplinary Optimization*, *36*(4), 393-401.
12. Geem, Z. W. (2008). Harmony search applications in industry. In *Soft Computing Applications in Industry* (pp. 117-134). Springer Berlin Heidelberg.
13. Cheng, Y. M., Li, L., & Chi, S. C. (2007). Performance studies on six heuristic global optimization methods in the location of critical slip surface. *Computers and Geotechnics*, *34*(6), 462-484. doi:10.1016/j.compgeo.2007.01.004.
14. Lee, K. S. (2009). Standard harmony search algorithm for structural design optimization. *Harmony Search Algorithms for Structural Design Optimization*, *239*, 1-49.
15. Degertekin, S. O. (2008). Optimum design of steel frames using harmony search algorithm. *Structural and Multidisciplinary Optimization*, *36*(4), 393-401. doi:10.1007/s00158-007-0177-4.
16. Mahdavi, M., Fesanghary, M., & Damangir, E. (2007). An improved harmony search algorithm for solving optimization problems. *Applied Mathematics and Computation*, *188*(2), 1567-1579. doi:10.1016/j.amc.2006.11.033
17. Hasançebi, O., Erdal, F., & Saka, M. P. (2009). Adaptive harmony search method for structural optimization. *Journal of structural engineering*, *136*(4), 419-431
18. Wu, B., Qian, C., Ni, W., & Fan, S. (2012). Hybrid harmony search and artificial bee colony algorithm for global optimization problems. *Computers & Mathematics with Applications*, *64*(8), 2621-2634.
19. Guo, L., Wang, G. G., Wang, H., & Wang, D. (2013). An effective hybrid firefly algorithm with harmony search for global numerical optimization. *The Scientific World Journal*, *2013*.
20. Fesanghary, M., Mahdavi, M., Minary-Jolandan, M., & Alizadeh, Y. (2008). Hybridizing harmony search algorithm with sequential quadratic programming for engineering optimization problems. *Computer methods in applied mechanics and engineering*, *197*(33), 3080-3091.
21. Dong, Y., Tang, J., Xu, B., & Wang, D. (2005). An application of swarm optimization to nonlinear programming. *Computers & Mathematics with Applications*, *49*(11-12), 1655-1668. doi:10.1016/j.camwa.2005.02.006
22. Mun, S., & Cho, Y. (2012). Modified harmony search optimization for constrained

design problems. *Expert Systems with Applications, 39*(1), 419-423. doi:10.1016/j.eswa.2011.07.031.
23. Das, S., Mukhopadhyay, A., Roy, A., Abraham, A., & Panigrahi, B. K. (2011). Exploratory power of the harmony search algorithm: Analysis and improvements for global numerical optimization. *Ieee Transactions on Systems Man and Cybernetics Part B-Cybernetics, 41*(1), 89-106. doi:10.1109/TSMCB.2010.2046035.
24. Geem, Z. W. (2005). Harmony search in water pump switching problem. *Advances in natural computation* (pp. 751-760) Springer
25. Greblicki, J., & Kotowski, J. (2009). Analysis of the properties of the harmony search algorithm carried out on the one dimensional binary knapsack problem. In *Computer Aided Systems Theory-EUROCAST 2009* (pp. 697-704). Springer Berlin Heidelberg.
26. Wang, L., Xu, Y., Mao, Y., & Fei, M. (2010). A discrete harmony search algorithm. In *Life System Modeling and Intelligent Computing* (pp. 37-43). Springer Berlin Heidelberg.
27. Wang, L., Mao, Y., Niu, Q., & Fei, M. (2011). A multi-objective binary harmony search algorithm. In *Advances in Swarm Intelligence* (pp. 74-81). Springer Berlin Heidelberg.
28. Wang, L., Yang, R., Xu, Y., Niu, Q., Pardalos, P. M., & Fei, M. (2013). An improved adaptive binary Harmony Search algorithm. *Information Sciences*, *232*, 58-87.
29. Hajela, P., & Lee, E. (1995). Genetic algorithms in truss topological optimization. *International journal of solids and structures*, *32*(22), 3341-3357.
30. Dorn, W.S., Gomory, R.E., Greenberg, H.J., "Automatic Design of Optimal Structures", Journal de Mecanique, Vo1.3, No. Mars, France, 1964.

31. Topping, B. H. V. (1983). Shape optimization of skeletal structures: a review. *Journal of Structural Engineering*, *109*(8), 1933-1951.

32. Topping, B. H. V. (1993). Topology design of discrete structures. In *Topology design of structures* (pp. 517-534). Springer Netherlands.
33. Hasançebi, O., & Erbatur, F. (2002). Layout optimisation of trusses using simulated annealing. *Advances in Engineering Software*, *33*(7), 681-696.
34. Bendsoe, M. P., & Sigmund, O. (2003). *Topology optimization: theory, methods and applications*. Springer.
35. Ohsaki, M., & Swan, C. C. (2002). Topology and geometry optimization of trusses and frames. *Recent Advances in Optimal Structural Design*, 97-123.
36. Mohr, D. P., Stein, I., Matzies, T., & Knapek, C. A. (2011). Robust Topology Optimization of Truss with regard to Volume. *arXiv preprint arXiv:1109.3782*.
37. Deb, K. (2000). An efficient constraint handling method for genetic algorithms. *Computer methods in applied mechanics and engineering*, *186*(2), 311-338.