KARADENİZ TECHNICAL UNIVERSITY

THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

DEPARTMENT OF CIVIL ENGINEERING

ARTIFICIAL BEE COLONY (ABC), HARMONY SEARCH (HS) AND PARTICLE SWARM OPTIMIZATION (PSO) BASED DISCRETE AND CONTINUOUS OPTIMUM DESIGN OF STEEL TRUSSES

MASTER THESIS

Patrick Jean De Dieu OUEDRAOGO

JANUARY 2018

TRABZON

KARADENİZ TECHNICAL UNIVERSITY

THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

DEPARTMENT OF CIVIL ENGINEERING

ARTIFICIAL BEE COLONY (ABC), HARMONY SEARCH (HS) AND PARTICLE SWARM OPTIMIZATION (PSO) BASED DISCRETE AND CONTINUOUS OPTIMUM DESIGN OF STEEL TRUSSES

Patrick Jean de Dieu OUEDRAOGO

This thesis is accepted to give the degree of

"MASTER OF SCIENCE"

By

The Graduate School of Natural and Applied Sciences at
Karadeniz Technical University

The Date of Submission : **02 / 01 /2018**

The Date of Examination : **22 / 01 /2018**

Supervisor : **Prof. Dr. Ayşe DALOĞLU**

Trabzon 2018

KARADENİZ TECHNICAL UNIVERSITY

THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

DEPARTMENT OF CIVIL ENGINEERING

Patrick Jean de Dieu OUEDRAOGO

ARTIFICIAL BEE COLONY (ABC), HARMONY SEARCH (HS) AND PARTICLE SWARM OPTIMIZATION (PSO) BASED DISCRETE AND CONTINUOUS OPTIMUM DESIGN OF STEEL TRUSSES

Has been accepted as a thesis of

MASTER OF SCIENCE

after the Examination by the Jury Assigned by the Administrative Board of the Graduate School of Natural and Applied Sciences with the Decision Number 1734 dated

22 / 01 / 2018

Approved By

Chairman	**:**	**Prof. Dr. Ayşe DALOĞLU**	……..……..……
Member	**:**	**Assoc. Prof. Dr. İlker USTABAŞ**	……..……..……
Member	**:**	**Assoc. Prof. Dr. Korhan ÖZGAN**	……..……..……

Prof. Dr. Sadettin KORKMAZ

Director of Graduate School

ACKNOWLEDGEMENTS

I would like, through this acknowledgement, to express my gratefulness to all the persons who didn't spare no effort to help me during the process of completing my master degree.

First of all, I wish to express my sincere thankfulness to my thesis advisor, Prof. Dr. Ayşe DALOĞLU, who has never grown tireless of providing me invaluable guidance, manifesting undeniable support or encouraging approaches, appreciating any single good work done and showing enlightening guidance during the progress of the thesis, and finally providing some constructive advices to my address.

My deepest gratitude to my beloved family, parents and all relatives, who have always given to me unceasing encouragements, understanding, inspiration and constant love. They have always been my source of motivation, and I acknowledge them for their never-ending multiples and various supports and prayers. Words cannot describe how proud I am for the belief they placed on me.

Special thanks to the university, Karadeniz Technical University, for allowing me to pursue my master degree in their prestigious school. Great thanks to all the professors and assistants who have shared their knowledge with me through the lectures and practical words.

I am grateful to all my friends for their endless contributions to make my life more colourful. In all the difficulties encountered in the accomplishment of my research work they have always stand by my side to comfort and ensure me that I can come over obstacles I am facing.

I could not end this acknowledgment without making a reference to my grandfather, Sawadogo Landaogo Pierre, whose blessings for my success never leave me in his lifetime. May him rest in peace.

Patrick Jean de Dieu OUEDRAOGO

Trabzon 2018

THESIS STATEMENT

After an architectural conception of a structure, it has to go under structural design. If the design is conducted by different engineers, many solutions could be proposed at the end their studies. Among those studies that all satisfy common rules of engineering design, some may appear to be better than others regarding the structural weight that directly impacts on the execution cost. The design that will bring out the lightest structure is called an optimum design. This is the aim of this thesis, to conduct optimization process on steel trusses. Before starting the study, size optimization and optimization algorithms employed, previous performance of well-known examples are analysed. Nevertheless, structural design through the use of Finite Element Methods, the use of Matlab and Structural analysis program Sap2000 should be mastered.

I hereby declare that "Artificial Bee Colony (ABC), Harmony Search (HS) and Particle Swarm Optimization (PSO) based discrete and continuous optimum design of steel trusses" thesis work has been presented in accordance with academic rules and ethical conduct under the supervision of Prof. Dr. Ayşe DALOĞLU. I also declare that references contain materials and results that are not original to this work.

Patrick Jean de Dieu OUEDRAOGO

TABLE OF CONTENTS

Page No

Master Thesis

SUMMARY

ARTIFICIAL BEE COLONY (ABC), HARMONY SEARCH (HS) AND PARTICLE SWARM OPTIMIZATION (PSO) BASED DISCRETE AND CONTINUOUS OPTIMUM DESIGN OF STEEL TRUSSES

Patrick Jean de Dieu OUEDRAOGO

Karadeniz Technical University
Graduate School of Natural and applied sciences
Civil Engineering graduate program
Supervisor: Prof. Dr. Ayşe DALOĞLU
2018, 83 pages

Steel structural designs are done based on many considerations that have been implemented over past years by civil engineers and research organizations. In this study, the design verifications assumed are based on prescriptions of American Institute of Steel Construction (AISC) that proposes to main ways that are Allowable Stress Design (ASD) and Load Resistance Factor Design (LRFD). Element internal forces used for the verifications are evaluated from different methods among which Finite Element Method (FEM) is the most common and also implemented in usual civil engineering programs. However structural design is not only limited in satisfying stability and reliability conditions but also in finding the most efficient and economical design. In this thesis, 2D and 3D steel trusses under discrete and non-discrete optimizations will be implemented using three specific metaheuristic algorithms, namely Artificial Bee Colony (ABC), Harmony Search (HS) and Particle Swarm Optimization (PSO). Given that analysis method is based on FEM, problems will be solved using two joint procedures while applying stress and displacement constraints. The first one consists of a script completely written in MATLAB that solves the whole problem, whereas the second retrieves the analysis results from SAP2000 and performs the optimization on MATLAB through the Open Application Programming Interface (OAPI) facility. The run of several examples demonstrates the effectiveness, robustness and fast convergence of employed algorithms.

Keywords: Optimum design of trusses, Artificial bee colony (ABC), Harmony Search (HS), Particle Swarm Optimization (PSO), Size optimization of steel trusses, Sap2000 OAPI, MATLAB, Discrete Variables, Non-Discrete

Yüksek Lisans Tezi

ÖZET

YAPAY ARI KOLONİSİ (ABC), HARMONİ ARAMA (HS) VE PARÇACIK SÜRÜSÜ OPTİMİZASYONU (PSO) İLE ÇELİK KAFES SİSTEMLERİN AYRIK VE SÜREKLİ OPTİMUM TASARIMI

Patrick Jean de Dieu OUEDRAOGO

Karadeniz Teknik Üniversitesi
Fen Bilimleri Enstitüsü
İnşaat Mühendisliği Anabilim Dalı
Danışman : Prof. Dr. Ayşe DALOĞLU
2018, 83 pages

Çelik yapısal tasarımlar, inşaat mühendisleri ve ilgili alanda çalışan araştırma enstitüleri tarafından gerçekleştirilen süren gözlemlere dayanarak yapılır. Bu tez çalışmasında, göz önünde tutulan tasarım kuralları, Amerikan Çelik Konstrüksiyon Enstitüsü'nün (AISC) yayınladığı, Güvenlik Katsayıları ile Tasarım Kuralları (GKT) ve Yük ve Dayanım Katsayıları ile Tasarımı (YDKT), yönetmenliklerine uygun yapılmıştır. Yapıların dayanımını kontrol etmek için kullanılan eleman iç kuvvetleri, inşaat mühendisliği alanında kullanılan farklı yöntemlerle yapılabilir ancak Sonlu Elemanlar Yöntemi (SEY) bunlar arasında en yaygın olan ve inşaat mühendisliği alanındaki geliştirilen bilgisayar programlarında da kullanılandır. Bununla birlikte, yapısal tasarım sadece sağlamlık ve güvenilirlik koşullarını tatmin etmekle kalmaz, aynı zamanda en verimli ve ekonomik tasarımı da bulmayı gerektirir. Bu tezde, 2 boyutlu ve 3 boyutlu çelik kafes sistemlerin ayrık ve ayrık olmayan (sürekli) tasarım değişkenleri ile optimizasyonu üç farklı meta sezgisel algoritma kullanılarak; Yapay arı Koloni (ABC), Harmoni Arama (HS) ve Parçacık Sürü Optimizasyonu (PSO), gerçekleştirilmektedir. Analiz yöntemi için SEY kullanılmakta, gerilme ve yer değiştirme sınırlayıcıları uygulanmakta ve problemler iki farklı yöntemle çözülmektedir. Birincisi, tamamen MATLAB'de kodlanan bir program ile, ikincisi SAP2000'den analiz sonuçlarını alarak ve Açık Uygulama Programlama Arayüzü (OAPI) özelliği aracılığıyla, MATLAB üzerinde optimizasyon gerçekleştirir. Çeşitli örneklerle uygulanılan algoritmaların etkililiği, sağlamlığı ve yakınsama hızı gösterilmektedir.

Anahtar Kelimeler: Kafes sistemlerin optimum tasarımı, Yapay arı koloni (ABC), Harmoni Arama (HS), Parçacık Sürü Optimizasyonu (PSO), Çelik kafeslerin boyut optimizasyonu, Sap2000 OAPI, MATLAB, Ayrık Değişkenler, Ayrık Olmayan Değişkenler

LIST OF FIGURES

Page No

LIST OF TABLES

Page No

ABBREVIATIONS AND SYMBOLS

Abbreviations

2D	2-Dimensionnal
3D	3-Dimensionnal
ABC	Artificial Bee Colony
AISC	American Institute of Steel Construction
ARCGA	Adaptive Real-Coded Genetic Algorithm
ASD	Allowable Stress Design
BA	Bees Algorithm
BW	Band Width parameter
CA	Cultural Algorithm
CS	Cuckoo Search
CSI	Computers and Structures, Inc.
DE	Differential Evolution
DHPSACO	Discrete Heuristic Particle Swarm Ant Colony Optimization
EA	Evolutionary Algorithm
EF	Employed Foragers
FEA	Finite Element Analysis
FEM	Finite Element Method
GA	Genetic Algorithm
HM	Harmony Memory
HMCR	Harmony Memory Consideration Ratio
HMS	Harmony Memory Size
HPSO	Heuristic Particle Swarm Optimizer
HS	Harmony Search
KKT	Karush–Kuhn–Tucker
LRFD	Load Resistance Factor Design
MABC	Modified Artificial Bee Colony
OAPI	Open Application Programming Interface
OC	Optimality Criteria

PAR	Pitch Adjusting Ratio
PSO	Particle Swarm Optimization
PSOPC	Particle Swarm Optimizer with Passive Congregation
R	Recruit
S	Scouts
SGA	Steady-State Genetic Algorithms
RO	Ray Optimization
TLBO	Teaching Learning Based Optimization
UF	Unemployed Foragers
VBA	Visual Basic Applications

Symbols

A_i	cross-section area of the i^{th} member
Amin	Lower bound of the design variables
Amax	Upper bound of the design variables
C_1	cognitive parameter
C_2	social parameter
C_c	Decision Coefficient
E	Elasticity Modulus
f_1	External nodal forces at node 1
f_2	External nodal forces at node 2
fi	Objective function of the food source Ai
fmin	Lowest objective function
FW	Fret Width (Bandwidth)
FW_damp	Width Damp Ratio
Fpenalty	Penalty Term
f (x)	Objective function
Fy	Yield stress
K	Stiffness Matrix
K_G	Stiffness matrix in global coordinates
k	Effective length factor
L_i	Length of the i^{th} member

MaxIt	Maximum Number of Iterations
n	Number of variables
nNew	Number of New Harmonies
nOnlooker	Number of Onlooker Bees
NP	Number of Particles
nPop	Population number
nVar	Number of design variables
pi	probability of the food source i
$r_{1j}(t)$, $r_{2j}(t)$	Stochastic components
ri	Radius of gyration
T	Transformation matrix
u_1	Displacement at node 1
u_2	Displacement at node 2
$v_{ij}(t)$	Velocity in dimension j at time t
Vmax	Maximum Velocity
W(A)	Weight of the Structure
wdamp	Weight Damping Ratio
$x_{ij}(t)$	Position in dimension j at time t
$y_{ij}(t)$	Personal best positions in dimension j
$\hat{y}_j(t)$	Neighbourhood best positions in dimension j
ρ	Weight density of material
ω	Inertia Weight
λi	Slenderness ratio
σ_i	Constraint

1. INTRODUCTION

1.1. General Overview

Steel has many desirable characteristics as one of the most common construction materials in civil structure. Some of the properties of steel that make it desirable for engineering are its malleability as well as high strength to mass ratio. Steel structures are commonly used as warehouses, factories, and housings because they mainly provide shelter and personal space for individuals. Their design conducted by committed engineers applying common rules of steel construction needs to bring out variants of the structural system that is some well-defined sense the “best” solution regarding to costs or weight while satisfying all relevant engineering constraints, such as maximum allowable stress in each member or displacements at elements end nodes. Undeniably the material cost represents the foremost driving element in the construction of engineering structures and it can be minimized by abating the structural system weight or volume through an optimizations process.

Over the past several decades, due to its direct applicability to the design of structures, structural design optimization has become a critical and challenging activity that has gained substantial concern (Lee and Geem, 2004). Optimum design of truss structures is an approach to find a minimum weight of truss structure with no violation of certain constraints. Optimal designing of structures has a major role in reduction of material usage which allows project constructors to save considerable money from the initial provisional cost. This is one of the main objectives of each construction project, to complete the construction with the least possible cost. In general, optimization of truss structures can be considered as size optimization which deals with cross-sectional areas of members. Each optimization problem requires a clearly defined objective function, design variables and constraints for the states of problem formulation. Depending on the class of a given problem and the specific needs several objective functions can be identified and a range of design variables considered. In this present study, two types of optimizations will be performed based on cross sections selection. When the variables are selected randomly between an upper bound and a lower bound without any reference to commercialized steel bars the optimization is called non-discrete or continuous. Because this optimization

method treats the design variables as continuous by ignoring their discreteness, solutions produced may either be far from optimum, or sometimes result in infeasible values. But in the optimisation of steel truss structures one of the most important considerations is that the design variables have to be selected from a list of discrete values due to the reason that member cross sections are only available in discrete standard sizes because of industrial steel manufacturing practices. This leads to a discrete optimisation problem, in which continuous optimisation techniques cannot be used, which makes it a more difficult task to solve. Through implementing these optimizations, designers are able to yield better design while saving time and money, therefore a number of optimization techniques has been implemented to solve structural problems. Generally, structural optimization methods are classified into two main groups that are modern techniques, recently developed and traditional methods used by optimization pioneers. In traditional structural optimization methods, optimality criteria approach and mathematical programming are explored, whereas modern structural optimization methods regroup heuristics and metaheuristic search techniques. Mathematical programming and metaheuristic techniques will get more attention in this study.

Mathematical Optimization has been used through its multiple Numerical Methods to solve recurrent engineering problems for many years. For weight and member sizing of truss structures optimization problems, the objective functions are defined in the design space, while the constraints imposition are made based on the behaviour response space, and the structural analysis is in charge of relating the two spaces. Traditionally, the optimisation problem solved by trial-and-error, is dictated by some design specifications and guided by the practice and intuition of the designer, has worked well as evident from the existence of many fine buildings and other structures. However, these technics have shown some limits to researchers that were relying on the experienced results.

The recent development of high-speed electronic computers technology has made the analysis and optimal design of problems much more accurate than ever before, which in turn has led to an increased use of structural optimisation research to achieve more efficient and economical design. The efficiency of metaheuristic’s performance relies on their capability to balance intensification and diversification during the search. In order to carry out global search, modern metaheuristic algorithms have been evolved, with three main purposes: solving problem faster, solving large problem and obtaining robust

algorithm. Metaheuristic optimization techniques have received then considerable attention from engineering researchers. The usage of optimization schemes in civil structure design helps designers to save time, material and cost. Using optimization schemes, designers can avoid the troublesome process of deriving designs through trial-and-error. Application of optimization schemes to civil structures is a tedious process as a large amount of iteration is required to be computed before the solution converges. These new metaheuristic optimization technologies developed during past three decades enable designers, precisely engineers, to find the most efficient and suitable solution amongst a multitude of design alternatives. Research is still vigorously pursued for many reasons, based on the need to fix a widened class of problems, take in consideration realistic definition of design variables, find methods to locate the global optimum and to reduce the designing time, and finally produce a continuous improvement of the technic's efficiency. There exist several optimization algorithms that have been developed over past years and these examples are the illustration: Genetic Algorithm (GA) (Holland, 1975), Particle Swarm Optimization (PSO) (Kennedy and Eberhart, 1997), Differential Evolution (DE) (Storn and Price, 1997), Artificial Bee Colony (ABC) (Karaboga, 2005), Bees Algorithm (BA) (Pham et al., 2006), Firefly Algorithm (Yang, 2009), Cuckoo Search (CS) (Yang and Deb, 2009), Water Cycle Algorithm (Eskkandar et al., 2012).

1.2. Research Objective

The main aim of the current study is to develop a computer design model which implements a structural size optimization under stress and displacement constraints. The objective of optimization procedure is to minimize the weight of the structure, such that the stress is not over the allowable stress in any element of the structure and the displacement at each single node does not exceed the maximum displacement allowed. The weight of the structure is obtained by assigning a cross-sectional area for each structural member satisfying the limitations as prescribed by AISC – LRFD and ASD specifications. This is done by associating optimization algorithms, Artificial Bee Colony (ABC), Harmony Search (HS) and Particle Swarm Optimization (PSO), chosen as the solution methods for their great flexibility and versatility, and the structural analysis process conducted either by MATLAB script or engineering software SAP2000. Upon completion of each case study, the program will provide an optimized steel truss design for the prescribed loading

condition, bringing out the smallest mass and the graph of convergence history during iterations.

The following sub-objectives were considered to achieve the main one:

- Review the existing methodologies to optimize truss structures.
- Develop a computer program to solve steel truss structures using FEM based on a MATLAB script.
- Create a computer program to solve steel truss structures using SAP2000 OAPI an MATLAB.
- Build up ABC, HS and PSO algorithms and connect them to the two design models enumerated above.
- Carry out validation and verification of the developed models.
- Compare the results obtained in this study with the established results available from previous researches.

1.3. Research Scope and Limitation

The scope area of this study is the weight optimization of 2D truss implemented in a MATLAB script for structural analysis and metaheuristic algorithms ABC, HS and PSO represented by 4 examples. Further into the thesis it is the optimization of 3D truss that will be studied by employing MATLAB and SAP2000 OAPI tools considering 3 study cases.

Limitation is mainly related to cross sectional optimization of truss elements. It is not extended to shape and topology optimization that are also contribute to reduce the structural weight.

1.4. Study Outline

The study reported in this present thesis consists of four chapters as respectively briefly described below:

Chapter 1 is a presentation of the thesis. It started with a general overview of the topic followed by the research objective. Afterwards the research scope and limitation were evoked with a final look at the study outline.

Chapter 2 will be dedicated to the literature review. It provides an overview of the concept of optimization, and truss size optimization respectively. The optimization problem of the structure is defined, in other words the objective function and the design constraints are detailed with a particular attention to the penalty function considered in constraint violation cases. After describing optimization by traditional method (mathematical method and optimality criteria approaches), optimization by metaheuristic algorithms is introduced with regards to ABC, HS and PSO.

Chapter 3 is an overview of 2D truss optimization based on codes entirely written in MATLAB. Therefore, some explanations of FEM will be given before presenting the 4 numerical examples.

Chapter 4 is devoted to 3D truss optimization process done by combining MATLAB and SAP2000 through OAPI. The description of OAPI principle will be followed by the analysis of 3 study cases.

Finally, Conclusions and research findings are summarized in Chapter 5. Based on the results of the study, recommendations for future work are also presented.

2. LITERATURE REVIEW

In this chapter some common concepts will be defined. After explaining the global comprehension of optimization and how it was introduced to human daily problems, the following part will focus on truss size optimization. Afterwards the parameters governing the optimization process such as design variables, objective function, constraints, penalty function are described. Traditional structural optimization methods, and modern structural optimization methods that contains Artificial bee colony (ABC), Harmony Search (HS) and Particle Swarm Optimization (PSO) algorithms will be the last interest sub-topic of the chapter.

2.1. Optimization

Optimization is a concept integrated to our daily lives. It can be defined as the science of evaluating the best solution of a problem defined mathematically, that commonly refers to a physical reality model. Its concern is to increase a company profit that implies an objective of economy and efficient use of available resources. It is the tool used to achieve the best conception in a timely and economical way.

Since the earlier history, human beings have always sought to maximize their profit. When from a time, they noticed that resources in the nature become limited, they started to economize the available energy, keep its outgoings, discomforts and reduce pain at minimum. This phenomenon is only possible if the best choice amongst all the feasible ways to accomplish the day-to-day event tasks, is made. Therefore, the optimal pattern of a process should be decided. The optimization process, from a mathematical point of view, is an application committed to achieve the best outcome of an operation, with respect to certain restrictions.

The first problems of optimization would have been formulated by Euclid, in the third century before our era, in his historical work « Elements ». Three hundred years later, Heron of Alexandria in « Catoptrica » states the "principle of the shortest path" in the context of optics. In the seventeenth century, the appearance of differential calculus led to the invention of optimization techniques, or at least stimulated its necessity. Newton developed then an iterative method to find the local extremums of a function by using the

notion of derivative, resulting from his collaboration with Leibniz. Problems are reduced to the search of derivative roots, and this new concept allows great progress in the optimization of functions. During the eighteenth century, the work of the mathematicians Euler and Lagrange led to the calculation of variations, established a technique of optimization under constraints called the "Lagrange multipliers". The nineteenth century has seen the growing interest of economists in mathematics. They set up economic models that should be optimized, which accelerated the development of mathematics. Since that time, optimization has become a pillar of applied mathematics and the expansion of techniques is such that many improvements have been made. This shows that the concept of “better design” is very old and man has always wanted to do better. But even if the idea of optimal design is probably very old, it is only possible in recent decades thanks to the development of the computer technology.

Optimization techniques are nowadays used in a very large number of interest areas including logistics, production management, finance, bank and insurance, the information transport protocols of computer networks, the transport of energy in power grids, military strategies, air and rail transport among others. And of course, these tools are used in mechanical engineering offices, in civil engineering, shipbuilding, aeronautics, automotive and so on. It is no longer a task reserved to specialists as it was in the 1980s. These methods can no longer be ignored and their understanding is getting more and more essential. Some optimization tools are specific to a given problem and have been developed to meet a singular need, while others are more general. Specific tools have seen their scope expand progressively. It should be underlined that all the optimization tools have one common characteristic: they are built on some mathematical fundaments and their implementation is done according to the Figure 2.1. below (Bruyneel, 2014). On the basis of an initial design defined by a number of variable values, called design variables, the optimization aims to automatically determine which design suits the best to the Criteria related to structural performance. The solution qualified of optimal design is found by this iterative process, alternating structural analysis and application of a technique of optimization. When applied to steel truss structures one can perform three main categories of optimization, size, topology or shape optimization regarding the type of variables considered. A size variable dictates the size of a member, which could be the area of the cross-section, or the inertia moment. Topology variable are related to the presence or

absence of members in the structure. Shape variable dictates the joint coordinates of the structure. In this study considerations are made for size optimization.

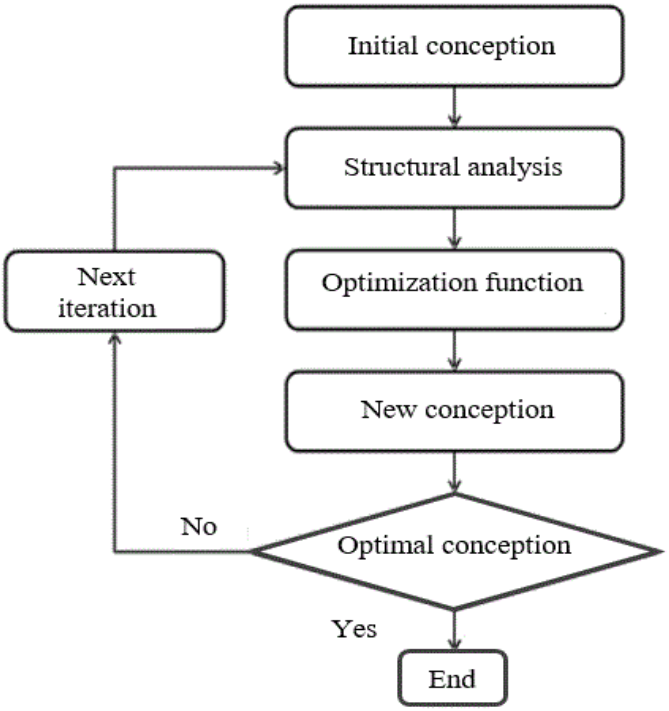


Figure 2.1. Optimization scheme

2.2. Truss Size Optimization

Truss structures is a part of skeletal constructions, a board category of man-made structures, regrouping bridges, water tower, cranes, roof support trusses, temporary construction and frameworks. Trusses distinctive look and utility are generated from their simple construction: bar elements resisting to axial forces, connected concentrically with joints.

The sizing optimization of these structures focuses on cross-sectional structural elements that are the variables. In the size optimisation of truss structural problems, the cross-sectional areas of all member elements are modified to meet the design requirements. Size optimization is done by assuming fixed topology (nodes connectivity) and geometry (nodes coordinates). Sizing optimization begins then at a structure in which the geometrical configuration is already defined. It seeks the optimum combination of element size specially the cross-sectional area. The geometry change is neglected when varying

these design variables, so it does not involve the redefinition of outer boundary shapes neither structural inner holes. The sizing of the structural elements in this manner is approached using methods such as performance-based design or strength-based design (Connor, 2003).

Realise a size optimization is to determine the ideal thickness of each structural element, regarding the performance goals and the forces that will be subjected to them during their life time. In a more global optimization process, size optimization is generally introduced after freeform optimization, that is to say, once the initial geometry of the all components have been defined and interpreted. One of the primary disadvantages of sizing optimization is that the topology of the structure remains fixed throughout the optimization procedure. Therefore, if a sub-optimal topology is chosen when formulating the optimization problem, the resulting structure will also be sub-optimal. The optimal design of a size optimization is the best design that comes out of the predefined structural geometry.

In structural design optimization, instead of defining the design variables by continuous values, they should be described by discrete variables, since cross-sectional areas usually belong to a certain predetermined set of values, provided by the manufacturers. Both discrete and continuous optimization are performed in many examples for the purpose of this study. The common formulation of size optimisation problems for truss structures with discrete and continuous variables are performed with considerations where the minimum weight is taken as the objective function, stress and displacement limitations as constraints with a special consideration for a penalty function governing constraints handling.

2.3. Design Variables

Each single optimization problem includes a number n of design variables. Design variables are quantities that appear in the problem definition and whose optimum value is sought. These are the unknowns of the optimization problem, here size design variables are considered and they can be related to a truss member's cross-sectional area, an inertia moment of a flexural element, an area of a beam, a thickness of a shell or a plate. This study focuses on size optimization of trusses, hence relies on members cross section.

Usually several ways may be employed differently to choose design variables in an optimization problem, and the choice is done accordingly to the nature of the problem and dictates which algorithm is applicable. The optimization problem is solved using the well-known optimization methods by integrating two kinds of design variables. First of all, there is discrete optimization (or combinatorial) that picks its values from a set of ready commercial cross sections. In this optimum design process of steel trusses, steel profiles are selected from an available list of practical sections. Even though optimization problems with discrete design variables are more difficult to handle than those with continuous design variables, they remain the most suitable solution for engineering design problems.

In the other hand, a design variable can generally be defined by a lower bound and an upper bound, explicitly described in the problem statement. It is evident that the lower bound is greater than zero, which, however, must be defined in the algorithms for solving the problem. The limits of the interval where the values are chosen define the boundary constraints of the optimization problem and they are related to the design variables A_i. For a problem involving n variables, the relations are stated this way:

$$Amin \leq Ai \leq Amax \quad i = 1, \ldots, n;$$

With A_{min} = lower bound and A_{max} = upper bound.

Design variables are essential optimization parameters used to formulate the objective function retrieved from the structural system definition. The objective function depends on these design variables and is written as follows: $f = f(x) = f(A_1, A_2, \ldots, A_n)$.

2.4. Objective Function

Whatever the field studied and whatever the study is to improve the performance of a structure or the optimization problem to be solved, several notions are systematically present. In any problem improvement, more than one objective could be formulated, because the designer may want to optimize many variants of the problem simultaneously, and this process is called multiobjective or multicriterial optimization. Though, multiobjective optimization algorithms are reputed to be more complex, computationally time consuming and expensive. Therefore, single criterion optimization is performed in most cases as it is for this present study. An optimization problem is defined as research, among a set of solutions possible solutions (also called decision space or search space), of the solution that minimizes (or maximizes) a function that measures the quality

of this solution. This function is called objective function or cost function. The objective function of a mathematical problem is what an optimization procedure uses to select better solutions over poorer solutions. Differently said, objective function will determine the effectiveness of the design under certain considerations. Optimization problem becomes then either a minimization or maximization depending on the problem's objective. For instance, if a structural design problem aims to bring out the smallest deflection of a beam, it is imperative to carry out the maximization of the beam's the stiffness. In such a problem, the objective function will be maximized. In the other hand, if the problem consists of making the project the least expensive, objective is to maximize the weight, then the procedure tries to move in the direction of solutions that decrease weight while still remaining feasible.

When an optimization problem is to be solved, the best possible solution to this problem is sought, which means the global optimum. However, there may be intermediates solutions, which are also optimums, but only for a limited subspace of the search space: these are called local optimums. This notion is illustrated in Figure 2.2. (Boussaid, 2013).

The problem of sizing optimization of truss structures involves optimizing cross sections A_i of the bars such that the structural weight W is minimized. Therefore, the global optimum is considered as the objective function. The mathematical formulation of the problem can be expressed as follows (Kirsch, 1982):

Find a design vector $A = [A_1, A_2, \ldots, A_n]$, (1)

To minimize f(x),

For weight optimization; $W(A) = \sum_i^n A_i L_i \rho$ (2)

Eq. (2) defines the weight of the truss structure, n is the total number of elements in the structure, A_i and L_i are, respectively, the cross-section area and length of the i^{th} member and ρ is the weight density of material.

The minimum design of weight should satisfy inequality constraints for the size of design variables and structural responses limits (Lee et al., 2005).

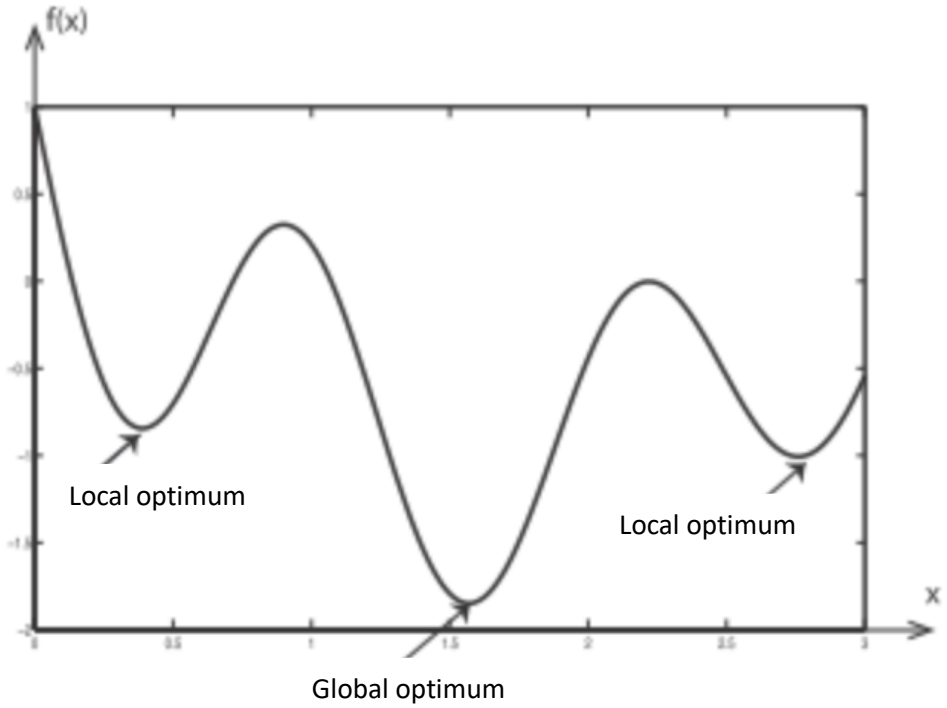


Figure 2.2. Local optimum and Global optimum

2.5. Constraints

Problems in the real world are often constrained. Several classical and evolutionary methods have been developed to take some restrictions in consideration. The restrictions to be satisfied for an acceptable design, formulated explicitly are called constraints. Constraint is generally defined as a condition of an optimization problem that the solution must satisfy. Constraints are functions that will restrict the search space. They will verify if the solution is feasible, but will not measure its quality. There are several types of constraints: firstly equality constraints, in second position integer constraints and lastly inequality constraints that are considered in this study. The set of candidate solutions that satisfy all constraints is called the feasible set.

In structural optimization problems, constraints are categorized in two main groups: design constraints also known as side constraints and behaviour constraints. Size constraints refer to functionality, fabrication, or aesthetics limitations. They are especially related to the lower and upper bounds of bars the cross sections.

Behaviour constraints deal with the stability of the structural system. These constraints may be limits upon parameters such as section stresses, nodal displacements, natural frequencies, and stability. Constraints considerations are made here for both,

displacements that are required not to be larger than fixed limits in order to preserve the serviceability conditions and stresses so that element's strength can be sufficient to resist the internal forces.

If a solution does not satisfy at least one of the constraints, it is said to be an infeasible solution, unlike the feasible solutions, which verify the set of constraints. The set of feasible solutions constitutes the eligible domain of the research space or the feasible space. The ratio between the size of the feasible space and the search space can be used as an index of difficulty for the problem (Michalewicz,1994).
The general constrained minimization design problem is defined as

Minimize W(A)

subject to $G_i(A) \leq 0$, i = 1, 2, ..., p

where, W(A) is the objective function. Functions $G_i(A)$ are the set of inequality constraints.

2.6. Constraint Handling/Penalty Function

The greatest part of engineering design problems involves some constraints prescription. Several techniques have been designed to resolve constrained optimization problems. The challenge of engineers in these kind of problems is how to optimize the objective function value avoiding at the same time its constraint violations. Thus, finding the appropriate method to handle constraints is extremely important for any optimization mechanism and design space exploration according to the studies conducted by (Coello Coello, 1999; Gen and Cheng, 1996). This present work interest is based on evolutionary algorithms, and because their operations do not always preserve feasibility, they consume considerable computational energy to seek for infeasible solutions. Considerable research has focused on constraint-handling techniques for evolutionary algorithms. Each of the approaches developed falls into one special category, cause these approaches can be grouped in four major categories (Michalewicz and Schouenauer, 1996). The categories evoked are: Penalty functions, Special representations and operators also called preserving feasibility of solutions, Separation of objectives and constraints or search of feasible solutions, and Hybrid methods. There are other classification schemes of constraint handling methods and Coello Coello (1999) state a fifth category that is, Repair

algorithms. Kicinger et al. (2005) have presented the advantages and disadvantages of these constraint handling techniques in their review papers.

The most recurrent approach in the evolutionary algorithm (EA) universe for constraints handling (particularly, inequality constraints) remains the use of penalties. Penalty functions have originally been proposed by Courant (1943) and were later more explored by Carroll (1961). In the penalty-function implementation techniques, each infeasible solution is penalized by the constraint violations magnitude. Many researchers tried to manipulate penalty function coefficients with the aim to balance the objective function with its constraint violations. The standing of penalty function methods as the most popular approach is due to their simplicity and easiness of application. Furthermore, they are functioning with a generic approach and have no limit (could be applied to linear or nonlinear problems). In optimization problems the role of penalty functions is to effectively transform into an unconstrained design a problem initially constrained, by adding to the objective function a value that determines the percentage of constraint violation in a particular solution namely a penalty term (Coello Coello 2000). The objective function of the problem is then replaced by the following function to optimize:

$$\text{Fitness}(A) = f(A) + F\text{penalty}(A)$$

In classical programming, exterior and interior penalty functions are considered. In the case of exterior method, the optimization first value is an infeasible solution that moves progressively to reach the feasible region. Evolutionary design has always exclusively focused on exterior penalty functions in which the penalty term is a small value chosen at some points distant from the constraint boundaries that will converge to infinity as the constraint boundaries are approached.

The method proposed for penalty function application uses a tournament selection operator, while comparing at the same time two solutions, with respect to the three rules enumerated below:

- Rule 1: Any infeasible solution is neglected to the profit of feasible solution.
- Rule 2: Among two feasible solutions, the one with the best objective function is selected.
- Rule 3: Among two infeasible solutions, the one that violates less the constraints is preferred.

By using the first and second rules, the search tends to the feasible region than infeasible region, and by employing second rule, the research tends to the feasible region with good solutions.

After engineers have studied multiple penalty functions, a general classification of the common used ones is presented as follows:

- Static penalty functions which have a constant value during an entire optimization process (Carlson, 1995).
- Dynamic penalty functions which are changing constantly throughout the runs (usually increase over time) (Joines and Houck, 1994).
- Annealing penalty functions which base its techniques on simulated annealing (Michalewicz and Attia, 1994).
- Adaptive penalty functions which operate change regarding the feedback retrieved from the optimum global search (Nanakorn and Meesomklin, 2001).
- Coevolutionary penalty functions which principle is to evolve solutions and penalty factors in different populations (Coello Coello, 2000).
- Death penalty functions which do not take in consideration infeasible solutions (Schwefel, 1981).

In this present thesis both static and death penalty functions are implemented because the examples studied are all relevant from previous works and the convergence of the results leading to similar values.

All these constraints handling processes are an integral part of the large repertory of optimization techniques that are going to be analysed in the following lines.

There is large amount of optimization techniques developed that can be used to determine the optimum design of structural systems. These are in general classified in two categories. The first one is the analytical methods, based on theory of calculus and variational methods. They can provide the exact optimal design solution by solving the system of equations that represent the optimality conditions. However, they are not suitable for solving large scale structures but are more convenient for studying a single structural component, reason why they are not considered in this study.

The second one is the numerical optimization techniques that are divided into traditional structural optimization methods and modern structural optimization methods also known as metaheuristic. They are adapted to computer programming and applicable to solve any kind of optimization problem. The optimization process starts with a selected initial design which is iteratively improved until reaching an objective value without any violation of the constraints. The process ends when the convergence criteria is satisfied bringing out the optimum design found.

2.7. Traditional Structural Optimization Methods

In this category of optimization methods two sub categories exist and they both use continuous design variables and the objective function as well as constraints are expressed as functions of the variables, however, they can also solve discrete optimization problems with a couple of alterations. These traditional methods are enumerated as follows:

2.7.1. Mathematical Programming

Mathematical programming techniques are amongst the well-known classes of structural optimization techniques which work based on gradient vector and the first derivatives of the objective function and the constraints regarding the design variables. This optimization approach, pioneered by Schmit et al. (1960), has actually enjoyed considerable success in a wide range of practical design problems. Many studies were done on the concern of employment of mathematical programming methods in the field so called, design optimization of engineering structures (Erbatur and Al-Hussainy, 1992). The basic idea of its implementation is to move to the left when the gradient of the objective function moves to the right and vice versa, to find a more convenient design and its most attractive feature is drawn from its generality in the sense that a broad class of structural optimization problems can, in principle, be treated in a unified manner.

Mathematical programming techniques can be globally classified as linear programming and nonlinear programming. The particular characteristic of linear programming is that the constraints and the objective function are expressed as linear functions of design variables whereas for nonlinear programming either constraints and/or the objective function are nonlinear functions of design variables. In nonlinear programming, for a solution to be optimal, the Karush–Kuhn–Tucker (KKT) conditions, generally described as the Kuhn–

Tucker conditions, are necessary conditions that should be provided to satisfy some regularities, furthermore it is an approach used in theoretical models to obtain qualitative results. With the necessity to allow inequality constraints in nonlinear programming, The Lagrange multipliers method which works only with equality constraints has been generalized into the KKT approach. The mixed system of equations and inequalities of the KKT conditions can't usually be solved directly, except in certain cases where a closed-form result is found by an analytical derivation (Leu and Huang, 2000).

Mathematical programming has been used by many researchers however; by about 1970 it became apparent that the application of this method, in combination with finite element structural analysis, to large scale structural optimization problems, required a large number of analyses and long run times to solve problems of only modest proportions (Saka, 2007). This situation led some investigators to abandon the mathematical programming approach and direct renewed effort toward implementing recursive redesign procedures based on optimality criteria.

2.7.2. Optimality Criteria Approaches

One of the most important research efforts to focus attention on the optimality criteria approach was reported by Venkaya et al. (1973). Other Early works are due to Venkayya et al. (1968). These early efforts have in recent years been followed by several notable studies, which pursue this same basic line of investigation. At approximately the same period, this design concept relatively simpler, based on a Lagrange multiplier technique and more specialized for structural design applications, got increasing interest in solving certain types of structural optimization problems (Berke and Khot, 1974). These studies have shown that the optimality criteria approach, including fully stressed design concepts, is well suited to achieving high efficiency in appropriate specialized situations. This has motivated for example the application of multiple variants of the optimality criteria approach to optimize the design of pin-jointed and frame structures.

The optimality criteria technique is an alternative way to produce optimum design. The rational use of optimality criteria is investigated for a class of structural synthesis problems where materials, configuration and applied load conditions are specified, and the minimum weight design is to be determined. Typically, in optimality criteria methods first

some derivatives are generated for a general case and then modified for each type of constraint. Next, in order to obtain an optimum design, the structural analysis is reduced by using first order Taylor series expansions to approximately evaluate the interdependence of the constraints. The optimality criteria update both the Lagrange multipliers and design variables by applying several iterative procedures. The update rules integrate some modifications to make more general the original formulas for frequency, displacement, and stress constraint (Berke and Khot, 1987). These rules are known as the Lagrange multiplier update rule, the design variable update rule and finally the hybrid design variable update rule.

The optimality criteria computer code has been developed based on already existing interfaces of routines analysis and was integrated into an optimally structural design test bed that is CometBoards. This code is composed of three modules and represented in Figure 2.3. (Patnaik et al., 1993):

- The optimization module,
- The analysis module,
- The interface module.

Gradient based formulations methods evoked above connote the use of different kind of approximations which do not reflect always the reality. Therefore, due to the drawbacks of these traditional methods to handle real world optimization design, the area of metaheuristics has considerably grown in the last two decades and became an unavoidable solution to optimize design problems. They have the ability to perform well in some contexts where exact optimisation techniques could not produce expected results.

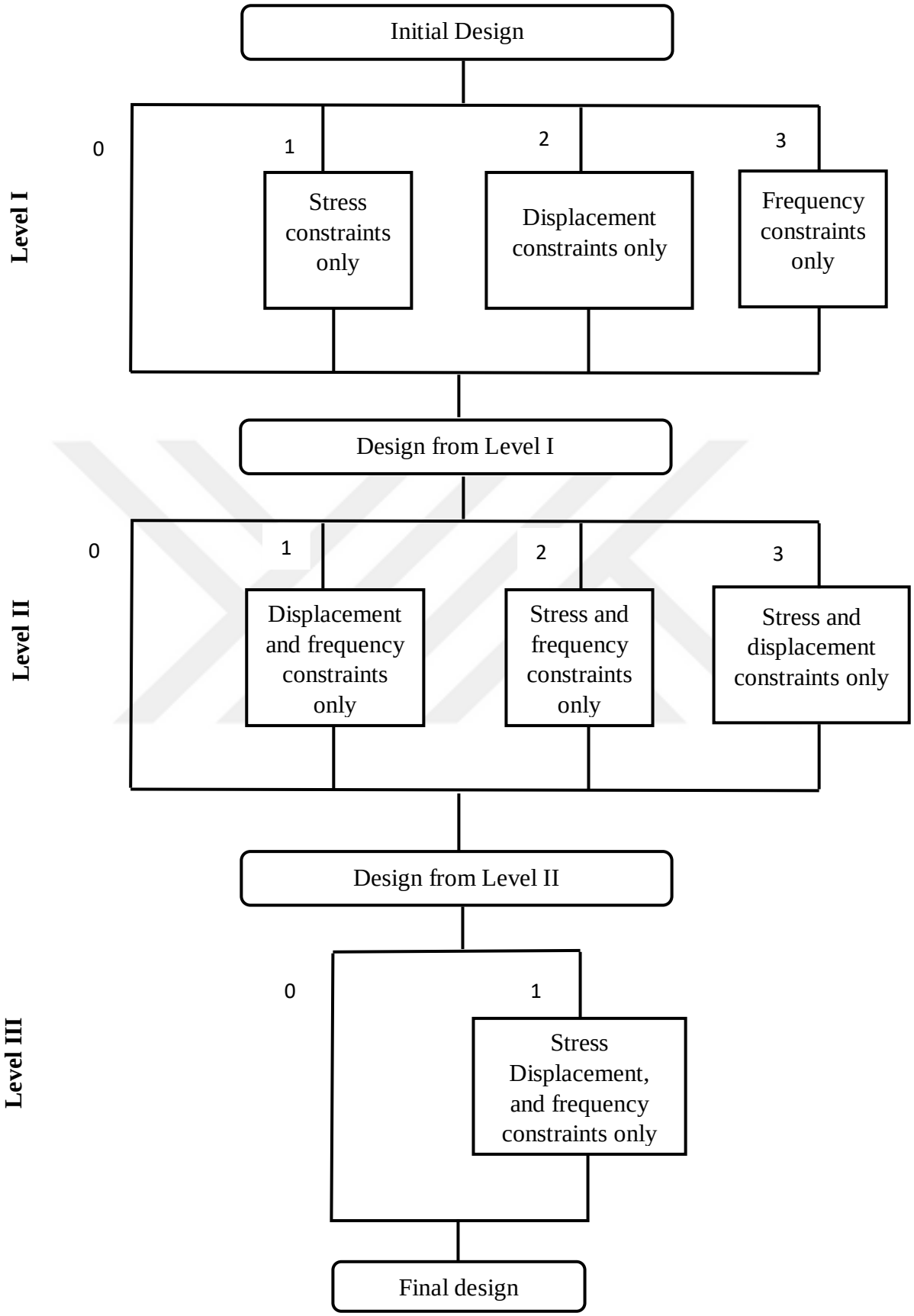


Figure 2.3. Optimality criteria code levels and constraint inclusion strategies

2.8. Modern Structural Optimization Methods: Metaheuristic

The word metaheuristic is derived from the composition of two words: meta, from the Greek “πέρα” meaning "beyond" (or "at a higher level") and heuristic from Greek “ευρισκειν” which means "to find" from where the famous Eureka of Archimedes. The term metaheuristics was introduced by Glover (1986). Metaheuristics are not problem or domain specific and can be applied to any optimisation problem. In fact, the algorithms based on this concept are elaborated as methods that can optimize a wide range of different problems, without important changes in the algorithm employed. Metaheuristics are in general non-deterministic, that means they are based on probabilistic transition rules and are optimisation methods that deliver reasonably good solution in a reasonable amount of time (Osman and Laporte, 1996).

Metaheuristics are stochastic search algorithms that use the ideas taken from the nature and do not implicate gradient computations of the constraints and the objective function. Most metaheuristics use random and iterative processes as a means of gathering information, exploring space research and deal with combinatorial problems. They can make use of the accumulated experience during the search for the optimum, for a better guide of the remaining research process. Hence many successful applications have demonstrated that metaheuristics is reliable in various contexts, either through comparison with other algorithms and/or applications to recurrent problems.

A wide range of metaheuristic algorithms have been developed over the past two decades and one of the stakes of the conception of metaheuristics is thus to facilitate the choice of a method and the adjustment of the parameters to suit a problem. There are different ways to classify metaheuristic algorithms based on characteristics but the most common one is “Population-based versus Single-solution-based”. Single-solution-based methods, also called trajectory methods, manipulate a single solution and are intensification oriented while population-based methods iterate and manipulate a whole family of solutions and are more focused on exploration of search space.

2.8.1. Artificial Bee Colony (ABC)

The Artificial Bee Colony (ABC) algorithm is a swarm based meta-heuristic algorithm that was first presented by Karaboga (2005) to optimize numerical problems. Later on, some modifications and improvements have been carried out by Karaboga and Basturk (2007) that detailed the main outlines of the ABC algorithm. Further applications have been presented by Karaboga and Basturk (2008) followed by Karaboga and Akay (2009). Later Hadidi and Kazemzadeh (2010) also studied some specific aspects of the algorithm.

ABC is a part of the most recently improved algorithms. The fundaments of the algorithm are specifically retrieved from the model proposed by inspired by the intelligence of honey bee colonies foraging behaviour and has demonstrated sustainable search abilities on many optimization examples. ABC algorithm tries to establish the natural behaviour of food foraging process from the honey bees. Honey bees base their search mechanisms on waggle dance to locate the optimal food source and permanently identify new ones.

2.8.1.1. Honey Bee's Colony

It is a well-known fact that honey bees are social insects that live in colonies consisting of thousands semi-sterile female workers, few thousand males (drones) and only one single queen. In the colony, everyone's task is clearly defined. The queen creates a new colony by mating with the males. All the other tasks are reserved to female workers that have the responsibility to process and store food, clean cells, feed larvae, secret wax and construct comb, and finally they are responsible of guarding the entrance (Ministry of Agriculture and Lands of British Colombia 2010). When the new born workers are about 3 weeks of age, the female workers previously in charge of the tasks within the hive enumerated above, will stop performing them and be reconverted to foraging duty for the rest of their lives (Honey Bee Biology 2010). These bees are the ones considered in ABC algorithm. Foragers as they are called search for promising nectar sources from different flowers in diverse directions up to 12 km from the hive, but their average fly radius is about 3km. When the foragers discover a food source, they load up the nectar and unload it once they return to the hive. This interactive behaviour of bees during food procuring is the waggle dance represented in Figure 2.4. (Lemmens et al., 2007) and Figure 2-5. By

performing this dance, successful foragers share with their hive mates the information about the amount of nectar within a flower, the direction to patches of flower and finally the distance between the hive and the food source. Most of the bees who watch the waggle dancers will decide to take the path of their nest mates, while some will independently leave the hive to an unknown direction, in the aim of finding a better food source than the one advertised in the dance. When foraging bees are performing the waggle dance, their direction reveals the position of the food source with a certain correlation to the Sun movement; the dance time indicates the nectar amount on the related food source and the waggles intensity, how far away it is.

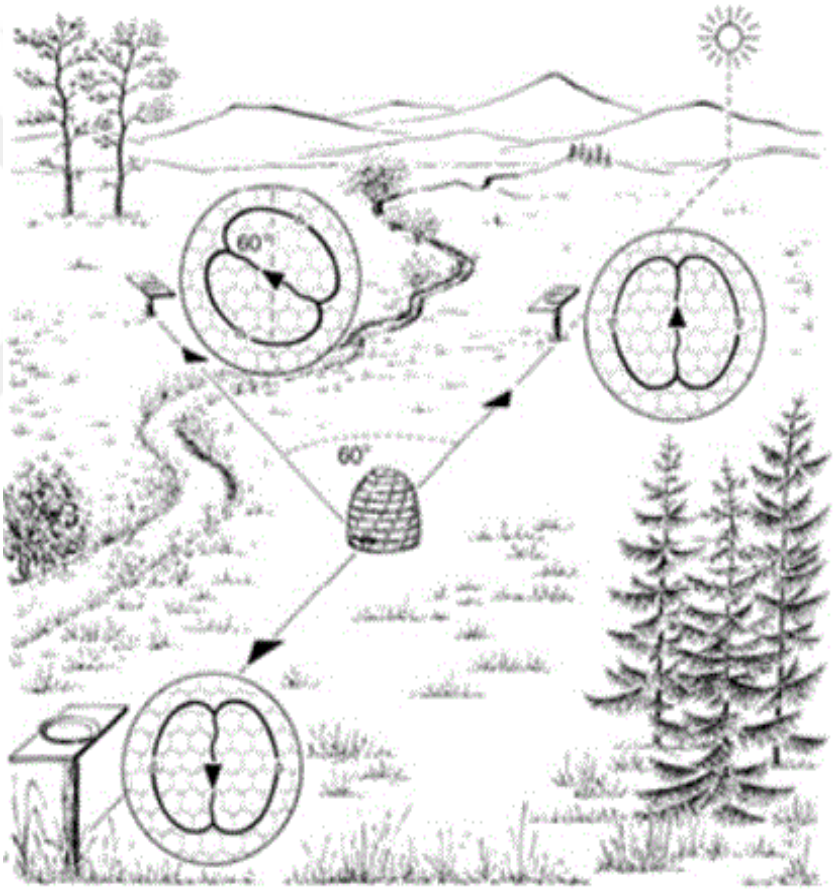


Figure 2.4. Waggle dance

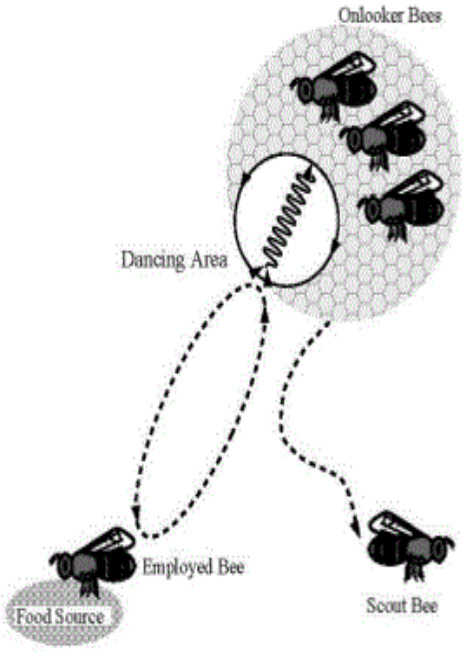


Figure 2.5. Honey bees foraging behaviour

2.8.1.2. Steps of ABC Algorithm

The ABC algorithm model consists of two essential components: employed bees, and unemployed bees divided into onlooker bees, and scout bees. In the ABC algorithm implementation, first half of the population represents the employed artificial bees and the second half, the unemployed bees (especially onlookers). The assumption is made such that the employed bees in the colony be equal to the number of food sources explorable around the hive. The employed bees are committed to seek for food in the food source memory meanwhile they share the available information to the onlooker bees. The onlooker bees then tend to choose from food sources found by the employed bees a good food source that suits them. When a food source has a high quality (fitness), it gets more attention from the onlooker bees than the one with lower quality left with an infinitesimal chance to be selected. The employed bees whose food sources have been abandoned become the scout bees that start investigating for new food source. The dances executed by employed bees are the food sources selection criteria for onlookers because after food foraging process employed bees come back to dance on the hive area. Karaboga (2005) described Tereshko model as Behaviour of real bees shown in Figure 2.6. to explain the

connection of the aforementioned models. It is Assumed that there are two discovered food sources: A and B. Employed bees are represented by EF (Employed Foragers), unemployed bees by UF (Unemployed Foragers), S is used for Scouts and R for Recruit (after watching the waggle dances they start searching for a food source).

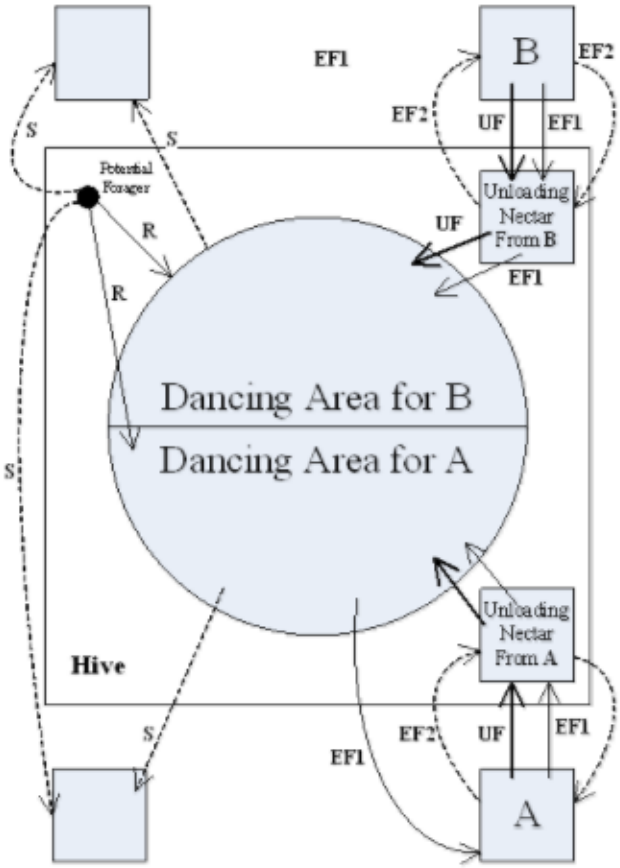


Figure 2.6. Connection between bees

Since ABC proved to perform well in different optimization problems, it has been used as an optimization algorithm in this study. Many parameters control the good performance of the algorithm. In total five parameters are necessary to the algorithm for finding optimum solutions. First there is the “number of bees in the colony” that determines the number of investigated simultaneous solutions. In second position comes the “improvement limit for a solution”, a very sensitive parameter, which affects how deep a bee tries to search the vicinity of a given solution and is used to escape from being stuck at local minimums. Third the “maximum number of iterations” is used to set the upper most limit of single studies that should be performed. Fourth an importance is given to “variable changing percentage” that indicates the total number of variables modified in

each iteration to bring out an improved solution. Last and none the least “number of independent runs” is used when large design spaces are involved.

The main steps of the algorithm defined by (Hadidi and Kazemzadeh, 2010) for solving optimization problems are stated as follows:

- Initialization.
- Repeat stage.
  - ✓ Place the employed bees on the food sources and determine their nectar amounts.
  - ✓ Place the onlooker bees on the food sources and determine their nectar amounts.
  - ✓ Scouts bees are sent to search new food sources.
  - ✓ The best food source found so far is memorized.
  - Stopping stage (when requirements are met).

(1) Generate the initial population x_i $(i = 1, 2, \ldots, SN)$
(2) Evaluate the fitness (fit(x_i)) of the population
(3) Set cycle to 1
(4) Repeat
(5) For each employed bee {
Produce new solution v_i by using (2)
Calculate its fitness value fit(v_i)
Apply greedy selection process}
(6) Calculate the probability values P_i for the solution (x_i) by (3)
(7) For each onlooker bee {
Select a solution x_i depending on P_i
Produce new solution v_j
Calculate its fitness value fit(v_j)
Apply greedy selection process}
(8) **If** there is an abandoned solution for the scout,
then replace it with a new solution which will be randomly produced by (4)
(9) Memorize the best solution so far
(10) Cycle = cycle +1
(11) **Until** cycle = MEN

Figure 2.7. Pseudo code of the ABC Algorithm

The main steps of the algorithm mentioned above are expressed through a MATLAB scripts in accordance with the pseudo code of the ABC Algorithm in Figure 2.7. and also, a

global outline of ABC is presented Figure 2.8. An important parameter governing the food source selection in this process is the probability (pi) of the food source i determined by Karaboga (2005). The unemployed bees pick a food source in accordance to the quality and quantity of the nectar found by the employed bees that visited that site. This means that there is a principle governing the food selection of unemployed bees that is based on the probability of nectar amount in the food source. The probability pi for ith food source is evaluated according to the following formula:

$$P_i = 0.9 \times \frac{fmin}{fi} + 0.1 \quad (3)$$

Where fi represents the objective function, more precisely weight (W), of the food source Ai and fmin is the lowest objective function value between all the evaluated solutions. So far, the higher the quality of a solution, the lower the weight and the higher the value of the resulting probability. After selecting a food source, the onlooker bee automatically creates a new food source that is evaluated and a greedy selection will be applied, in the same way the employed bees do.

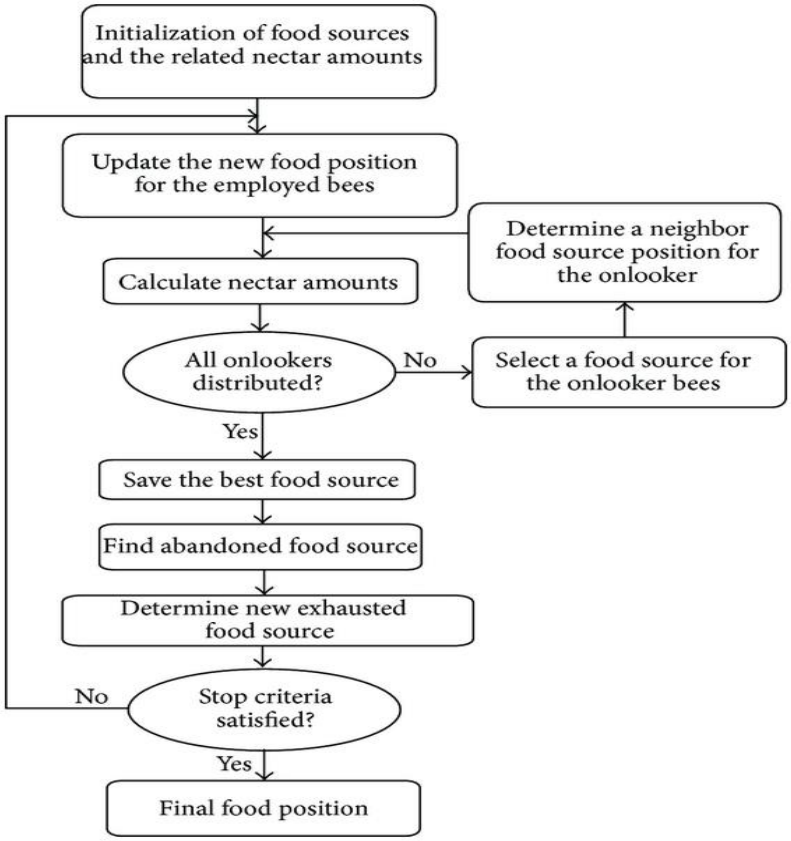


Figure 2.8. Flowchart of ABC algorithm

2.8.2. Harmony Search (HS)

While Harmony in nature is the emanation of a special relationship between several sound waves that are spreading with different frequencies and the human ears accept these musical tones reflexively, music harmony is considered as a combinatorial creation of sounds evaluated as pleasing to a group of listeners. This relationship was first developed by Pythagoras (582 BC-497 BC) a Greek philosopher and mathematician, in his famous experiment on stretched string, and later many researchers have focused on this phenomenon. Jean-Philippe Rameau (1683-1764), a well-known French composer and musicologist established the classical harmony theory (Kirkpatrick et al., 1983) and later the musicologist Tirro (1977) has produced a documentation on the history of American jazz.

Harmony search (HS) Algorithm, was introduced by Lee and Geem (2004) as a new metaheuristic algorithm that bases its operation on the music improvisation process, a search for a perfect state of harmony. HS is one of the most recent methods derived from the natural phenomena of musician's behaviour during a collective performance with their musical instruments considered as population members, to bring out some pleasing harmonies represented by optimal solutions. The considerations are made for a jazz improvisation during which a musician searches for a better state of notes combination. The major aim of a Jazz improvisation is to find musically pleasing harmony, the perfect state determined by each pitch of a musical instrument based on an aesthetic standard.

Recently, HS has become a popular algorithm in the evolutionary computation field and has successfully been applied to a wide variety of optimization problems due to its superiority to many other algorithms regarding its easiness of implementation, few parameters considered in adjusting and finally the simplicity of the concept. Some of its advantages are also summarized by Geem (2007), make it very suitable engineering applications. HS algorithm has been employed to solve large scale of problems, such as water distribution and games, and was revelled to have better performances while compared to other optimization techniques. During the recent years, HS was used in interest areas such as function optimization, pipe network optimization, mechanical structure design (Lee and Geem, 2004), stochastic equilibrium network design, and optimization of data classification systems (Wang et al., 2009).

Harmony is characterised by the simultaneous production of sounded musical notes regarding some chords with the unique objective to provide a pleasing effect to ears. Do, Re, Mi, Fa, Sol, La, and Si are the notes from which combinations are made in with any music instrument to produce a specific melody. Musicians are always striving to find a better harmony, and they only succeed by accomplishing numerous practices of changing the notes that are played. The improvisations of all the musicians on a given project are considered as a single harmony vector. During a trial performance if the pitches are evaluated to be a good harmony, it is stored in each player's memory with a high probability to produce a better harmony in their next try. This principle is similarly applied in civil engineering optimization. Within a range of values, each decision variable makes a choice, and all the chosen values are stored as a solution vector in each variable's memory as long as this vector is evaluated as a good solution. The probability in getting a better solution is increased during the next experience. This analogy between music improvisation and engineering optimization is described in Figure 2.9. (Alatas, 20010) and the comments below.

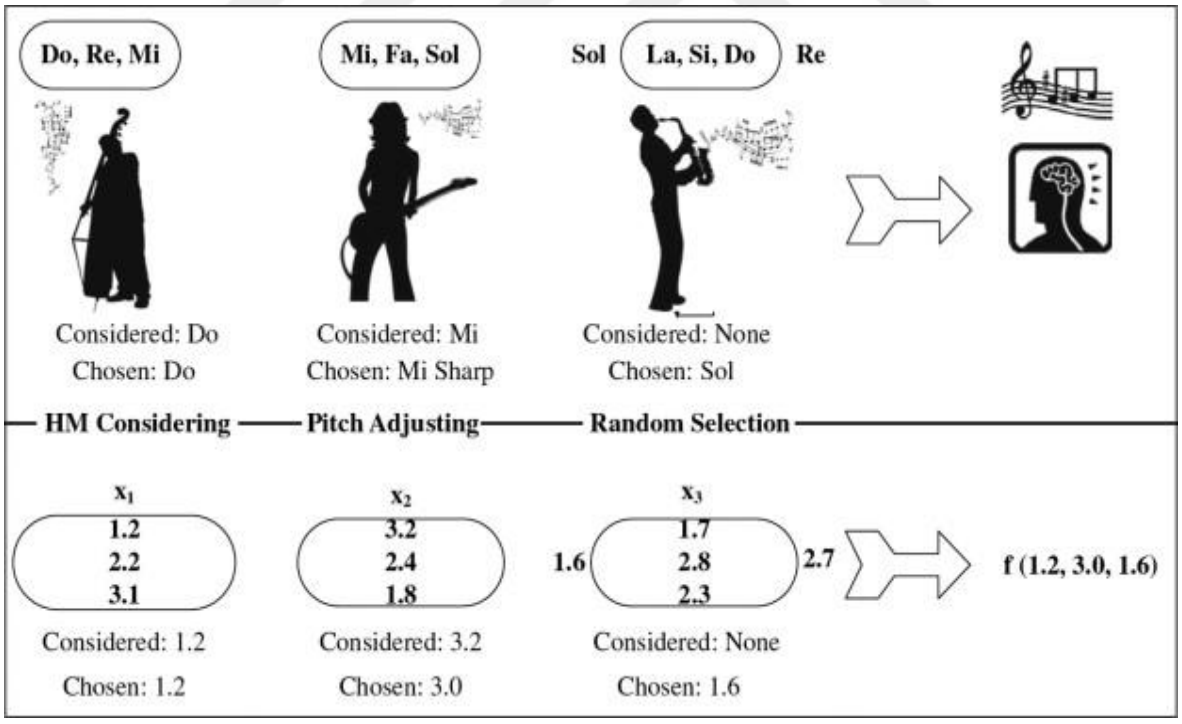


Figure 2.9. Analogy between music improvisation and engineering optimization

The figure above illustrates the analogy once again and the details stated can be understood through the explanations gave by Lee and Geem (2005) based on this illustration of a trio playing jazz music. The method employed by jazz musicians to select their notes is partitioned into three different phases: first select a pitch from the memory, second generate a nearby pitch to an existing one or finally play a random pitch from possible sounds range. These three processes constitute the fundament of the harmony search algorithm. The example established on a trio that is considered here, gives to each musician a certain amount pitches as it follows: guitarist {Mi, Fa, Sol}, saxophonist {La, Si, Do} and {Do, Re, Mi} for the double bassist. Out of the musical harmonies that are assigned to them, if the guitar brings out {Mi Sharp}, the saxophone produces {Sol}, and the double bass releases {Do}, {Do, Mi Sharp, Sol} is the new harmony vector. In the case this new harmony vector is better than another harmony in the Harmony Memory(HM), the new one is integrated in the HM while the existing worst one is simply excluded from the HM. The process is repeated until the best harmony is found.

In real optimization problem, musicians are replaced by the decision population and the sounds pitches replaced by the design variables. This study optimizes the cross-sectional areas of the truss bar elements; therefore, the design variables are the areas. If the first design variable selected is {1.2}, the second is {3.0} and the third one {1.6}, the new solution vector is {1.2, 3.0, 1.6}. And in accordance with principle employed for the sound pitches if the new solution vector is better than an existing one, the worst vector is just replaced by the new one. The process is performed many times until the best solution is obtained.

2.8.2.1. Harmony Search Parameters

The harmony search performance is tightly depending on many parameters as: harmony memory consideration ratio (HMCR), harmony memory size (HMS), pitch adjusting ratio (PAR), and a band width parameter (BW). Another parameter to take in consideration is the harmony memory (HM) that was employed earlier in the redaction. HM is the set of sounds combinations (designs variables) that are stored and will be used for generating new chords (designs variables). HMS is a parameter that represents the number of chords (designs variables) that are found in the memory. Depending on the problem to optimize, most examples in literature, made HMS vary between 10 and 50. The

HMCR is a parameter picked from the interval [0,1] that indicates the probability to be drawn a design variable from the HM. For instance, a HMCR value of 0.70 means that the probability of selecting a value from the HM is 70% and the probability for the variable to be randomly picked out of the HM is 30%. Once a decision is made to pick a value (pitch) from the HM, another test is needed to see whether the picked value should be pitch-adjusted or not. PAR is responsible of pitch adjustment which seeks for better design in the neighbourhood of the current design. The process of adjustment depends on a probability evaluated according to the formulas $HMCR\times\ PAR$ if a new design variable is to be selected, and $HMCR\times\ (1-PAR)$ when the design value is to be kept. PAR is also a value between 0 and 1. BW is an important factor that influences considerably the convergence rate toward optimal solutions.

2.8.2.2. Harmony Search Steps

Harmony search consists of five main steps as represented by the flowchart in Figure 2.10. (Lee and Geem, 2005) and are implemented trough computer algorithms following descriptions of the pseudo code in Figure 2.11.**:**

1. Formulate the optimization problem and Initialize the algorithm parameters values (HMCR, PAR, HMS, BW).
2. Initialize the harmony memory.
3. Improvise a new harmony and evaluate its performance.
4. Update the harmony memory as far as it is necessary.
5. Check for the termination criteria.

All these steps are reviewed in details in the following lines.

Step 1: Optimization problem formulation and HS parameters specification.

The HS algorithm parameters stated above are chosen in this step and their selection is made depending on the type of problem. The objective problem is to minimize f(x) with the variables limited by a lower bound and an upper bound. Harmony search parameters should be specified with care, especially for HMCR and PAR that are very important in improving the solution vectors. For example, a relatively high HMCR value that gets closer to one reduces the probability of generating new values in the HM.

Conversely a small HMCR value close to zero reduces the exploitation of the combinations stored in HM. Therefore, a good set of parameters will increase the capacities of the algorithm to look for the optimal solution.

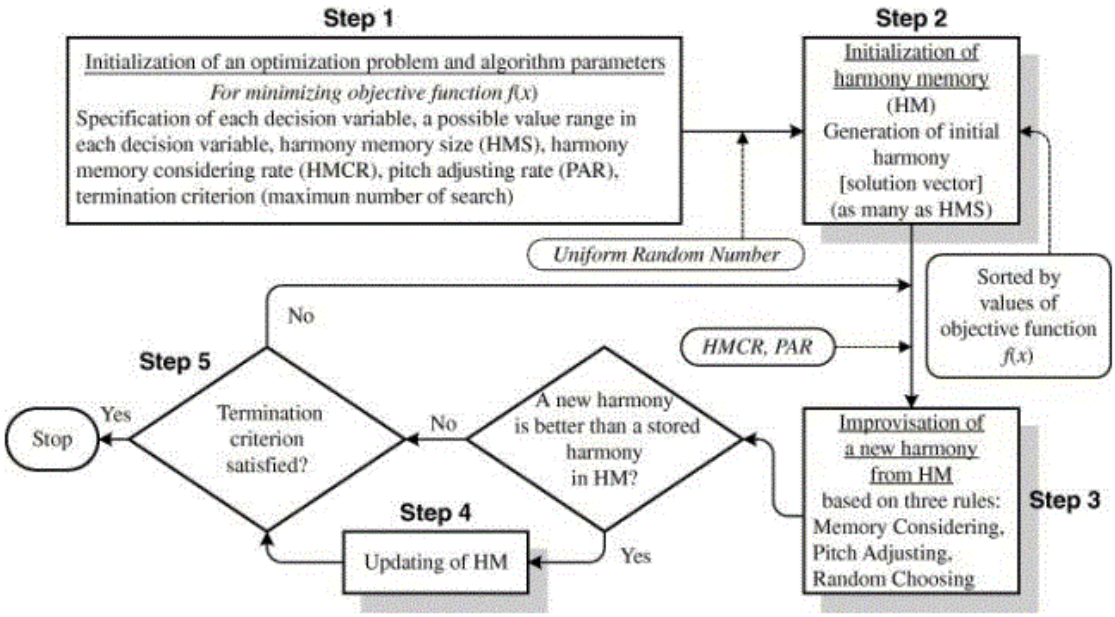


Figure 2.10. Flowchart of Harmony Search algorithm

```
// Initialize
Initialize parameters
Initialize the HM
Compute the fitness of each vector
// Main loop
while (not_termination)
   for j = 1 to number of decision variables (N) do
         r1 = uniform random number between 0 and 1
         if (r1 < HMCR) (memory consideration)
            x[i,j] will be randomly chosen from HM
            r2 = uniform random number between 0 and 1
            if (r2 < PAR) (pitch adjustment)
               r3 = uniform random number between 0 and 1
               r4 = uniform random number between 0 and 1
               if (r3 < 0.5)
                  x[i, j] = x[i, j] - r4*bw
               else
                  x[i, j] = x[i, j] + r4*bw
               end if
            end if
         else (random selection)
               x[i, j] = x ∈ Φ (Φ = Value Set)
         end if
      end do
   if x[i, :] violates problem constraints
      Handle constraints
   end if
   // Evaluate the fitness
   fitness_x[i] = evaluate_fitness(x[i, :])
   // Update HM if applicable
   update_memory(x[i, :], fitness_x[i])
end while
```

Figure 2.11. Pseudo code of HS

Step 2: Initialize the harmony memory.

After the parameters are set the matrix of harmony memory is generated and filled with some specific solutions that are either randomly generated or selected from a population of discrete values with a total number equal to the harmony memory size. Each row of the HM matrix contains the values affected to the design variables. The harmony memory matrix is represented in Figure 2-12 (Lee and Geem, 2005).

$$HM = \left[\begin{array}{cccccc|c} x_{1,1} & x_{1,2} & \cdots & \cdots & x_{1,N-1} & x_{1,N} & f(x_{1,:}) \\ x_{2,1} & x_{2,2} & \cdots & \cdots & x_{2,N-1} & x_{2,N} & f(x_{2,:}) \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{HMS-1,1} & x_{HMS-1,2} & \cdots & \cdots & x_{HMS-1,N-1} & x_{HMS-1,N} & f(x_{HMS-1,:}) \\ x_{HMS,1} & x_{HMS,2} & \cdots & \cdots & x_{HMS,N-1} & x_{HMS,N} & f(x_{HMS,:}) \end{array}\right]$$

Figure 2.12. Harmony memory form

The design variables stored in the HM matrix are analysed to produce the results of the corresponding objective functions that are shorted in descending order according to the solution.

Step 3: Improvise a new harmony.

In this third step, a new harmony vector will be generated on the basis of three rules as specified in the concept of jazz music players: memory considerations, pitch adjustment ratio and selection done randomly. For instance, the first value of the new vector can be chosen from the existing HM and so on for the other variables. This process has a governing factor named HMCR that is also responsible for determining if the value has to be generated randomly. If the component's value comes from the HM it may be pitch adjusted depending on the value of the PAR. The pitch adjustment is different for continuous variables and discrete variables. The concept of improvement is described in Figure 2.13.

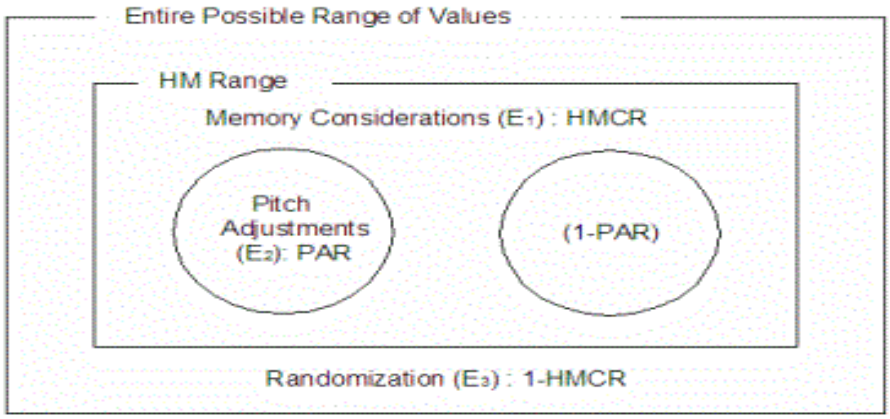


Figure 2.13. New harmony improvisation concept

The explanation of the process in which new harmony is generated is referred in Figure 2.14.

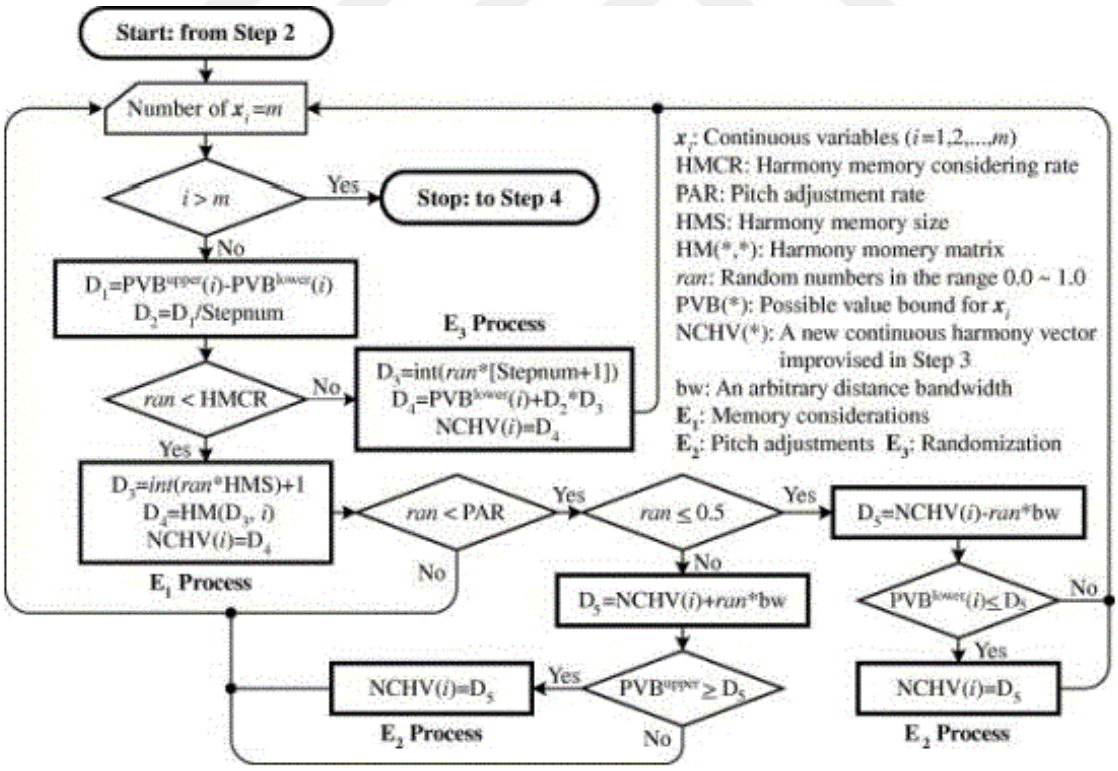


Figure 2.14. New harmony improvisation flowchart

Step 4: Update the harmony memory.

If the improvised harmony has a better performance than the old harmony regarding the objective function value, and the prescribed constraints violation, this newest harmony takes place in the HM while the old harmony considered as worst is excluded from the HM. Otherwise the improvised harmony is simply neglected.

Step 5: Termination criteria.

In the final step, steps 3 and 4 of the harmony search are repeated until the termination criteria are satisfied. In the early formulations of HS algorithm only one termination criterion was used and relies on the maximum number of iterations. With the progress, a new factor initiated by Cheng et al (2007) is integrated to stop the computation process. This second criterion stops the algorithm before reaching the imposed iteration number if and only there are no improvements in the solution.

2.8.3. Particle swarm optimization (PSO)

Particle swarm optimization (PSO) is a population-based stochastic random search algorithm that was established by Kennedy and Eberhart (1997) and later modifications suggested by Pedersen (2010) were integrated. PSO was developed as a search and optimization method that is a biologically and sociologically inspired swarm intelligence method, based on a simulation of some social animal's behaviours and dynamic movements such as insect communications, birds flocking and fish schooling. In the PSO algorithm, each particle represents a candidate solution to the optimization and the population individuals are considered as weightless and volumeless particles that move at a given speed. Each particle can adjust its speed dynamically according to other particles and also referring to the group experience to progress in synergy so that they all can reach the optimal location gradually. The velocity of a particle is evaluated based on its attraction towards two main positions in the research space that are the best position localized by the particle and the best position in its neighbourhood. The original PSO algorithm can be implemented through several swarm neighbourhood topologies constructed based on the index of each particle. Six main categories are stated to be used in general cases (Medina et al., 2009): ring, fully connected, mesh, toroidal, tree, and star topology as shown in Figure 2.15. This present study focuses on two strategies, namely a star topology and a ring topology. In the star topology neighbourhood is the entire swarm, whereas in the ring

topology neighbourhood consists of a restrained group made of the immediate neighbours. The two topologies evoked above are respectively determined as global best and local best, and no significant difference is noticed regarding their performances when evaluated with a sample of benchmark functions. For the purposes of this study, the methodology employed is a global-best topology. The PSO algorithm is well known for its ability to avoid difficulties faced in traditional optimization algorithms, to achieve a better optimization solution within a small-time amount when solving high dimensional complex functions (Tunchan, 2009). The concern of PSO behaviour is to use social forces to make some grouping depending on the memory of each individual particle as well as the knowledge of the whole swarm gained through the years. Swarm intelligence is defined as the phenomenon leading this behaviour.

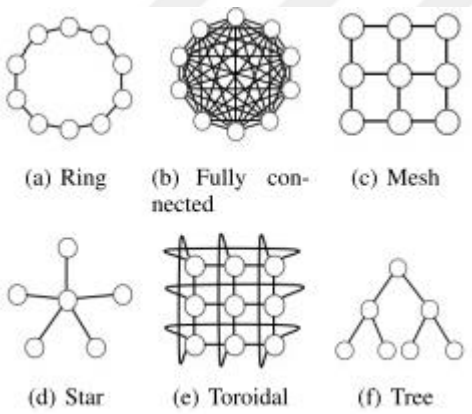


Figure 2.15. Neighbourhood topologies

Swarm intelligence has been marked as a multidisciplinary character since it was first used in cellular robotic systems by Hackwood and Wang (1988). It's a discipline that makes a focus on the collective behaviour of individuals with each other and with their environment. School of fish, flock of birds, herd of land animals are some examples of swarm intelligence that are illustrated in Figure 2.16. Swarm intelligence is built up with four basic principles. The first principle, so called proximity principle, sustains that space and time computations are elementary tasks swarm should be able to accomplish. The second one is the quality principle that admits that the swarm is capable of responding to quality factors such as food stuffs or safety of the location. The third principle is diverse principle that makes the swarm distribute resources along many modes. And finally, the stability principle with its declaration that the swarm's behaviour should not undergo some changes from one mode to another upon every fluctuation of the environment.

a) Bird flocking

b) Fish schooling

c) Animal herding

Figure 2.16. examples of swarm intelligence

2.8.3.1. Control Parameters

The basic parameters of PSO algorithm are divided into many parts but only the most important are explained as follows:

- Cognitive and social parameters, C_1 and C_2

Also called acceleration constant, they represent "trust" settings that indicate the confidence degree, and are very important when it comes to identifying the trajectory of

particles. In the process of finding the best solution C_1 (cognitive parameter) indicates the percentage of the confidence the particle places in itself, while the percentage of confidence by the whole swarm is expressed by C_2(social parameter). Usually C_1 and C_2 are taken equal 2, but diverse values could also be considered, with respect to $0 < C_1 + C_2 < 4$ (Perez and Behdinan, 2007)

- Vector containing the maximum allowable velocity for each dimension during one iteration, Vmax

The velocity of the particles is calculated based on the inertia weight model of Shi and Eberhart (1998) expressed by the formula below.

$$v_{ij}(t+1) = \omega v_{ij}(t) + c_1 r_{1j}(t)\left(y_{ij}(t) - x_{ij}(t)\right) + c_2 r_{2j}(t)\left(\hat{y}_j(t) - x_i(t)\right),$$

where

$v_{ij}(t)$: the velocity in dimension j at time t,

$x_{ij}(t)$: the position in dimension j at time t,

ω : the inertia weight,

c_1 , c_2 : the cognitive and social coefficients, respectively.

$r_{1j}(t)$, $r_{2j}(t)$: the stochastic components of the algorithm and

$y_{ij}(t)$, $\hat{y}_j(t)$: personal and neighbourhood best positions in dimension j, respectively.

The maximum velocity value is usually defined as the equivalent of half the length of the search dimension's interval that is the mean value of the lower bound and the upper bound. It is very useful to regulate the trajectory modifications of each particle during the consecutive iterations, and also represents a mean to prevent particles wider expansions in the problem search space. To adjust exploration and prevent probable explosions in the particle's search, it's better to assign small values to Vmax. Implementation of velocity clamping is represented as follows:

$$v_{ij}(t+1) = \begin{cases} -v_{max} & \text{if } v_{ij}(t+1) < -v_{max} \\ v_{max} & \text{if } v_{ij}(t+1) > v_{max} \\ v_{ij}(t+1) & \text{otherwise} \end{cases}$$

- Inertia weight, w

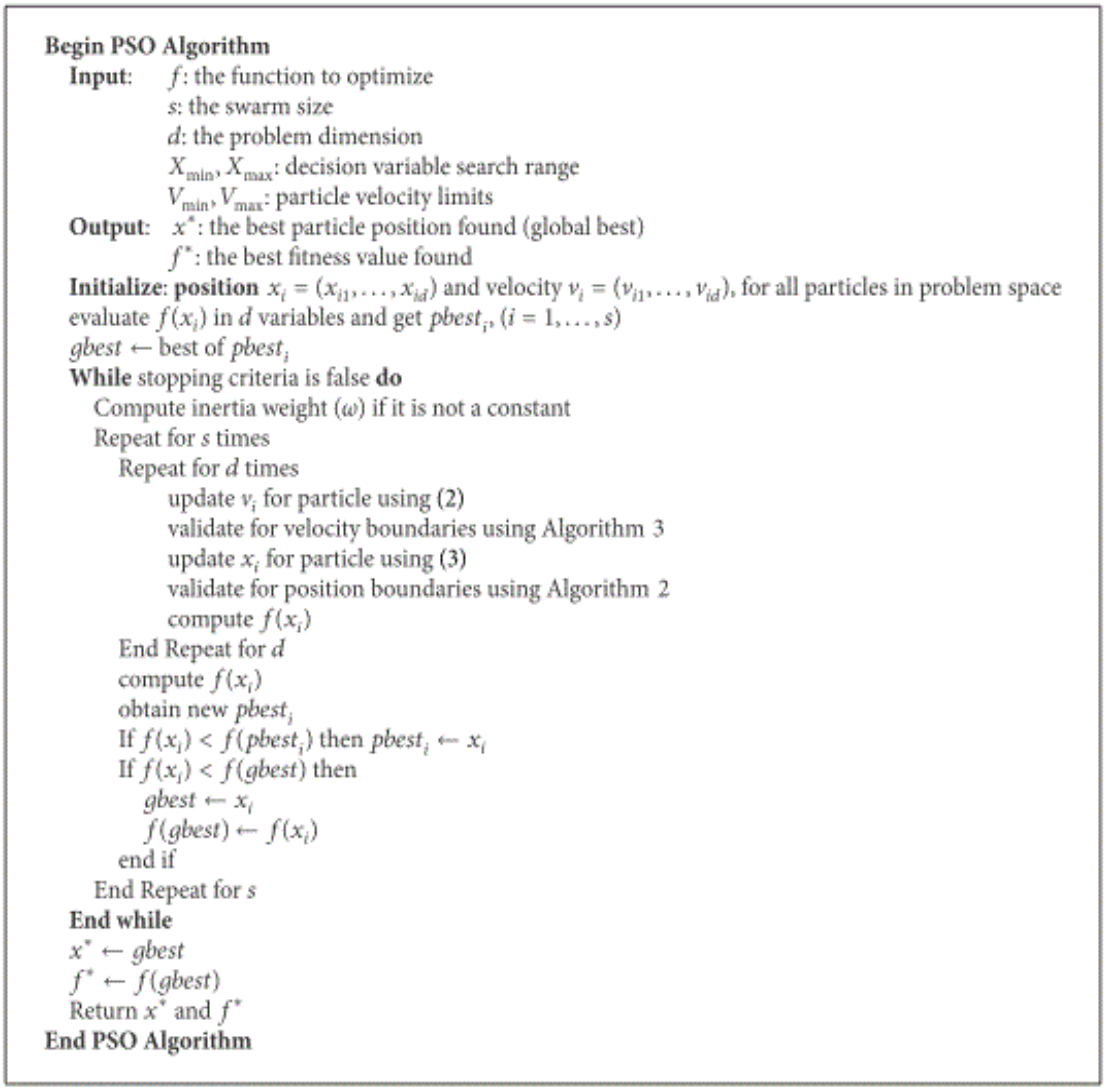
The inertia weight is generally defined as a scale factor linked to the velocity during the previous steps in the velocity update equation and is usually set to be less than 1 but can also be updated during iterations. In other words, it is required to control the impacts of previous velocities on the current iteration's velocity. The field of inertia weight control strategies is very large, with no clear declaration on which strategy has the best performance. The previous studies conducted on the inertia weight were not able to point out which strategy indicates how and when each of the multiple control strategies, adhere the most to the theoretical convergence criterion the best. 18 inertia weight control strategies have been examined and dissected analytically to determine their convergence behaviour through empirically investigations on a suite of 60 boundary-constrained benchmark problems. Commonly, PSO with a decreasing inertia weight is used by major searchers. All the strategies have revealed that if a large inertia weight is employed the global exploration is made easier, whereas a small inertia weight facilitates local exploration. An appropriate selection of inertia weight parameter performs the optimization process with less iteration.

- Number of particles, NP

Typically, values vary from 10 to 40, but for most problems 10 particles are sufficient to produce good results. For complex problems, the number can be increased to 50-100.

2.8.3.2. Particle Swarm Algorithm's Implementation

The PSO algorithm steps can be outlined in accordance with the pseudo code in Figure 2.17. and the flowchart of the basic particle swarm optimization technique given in Figure 2.18.



Begin PSO Algorithm
Input: f: the function to optimize
s: the swarm size
d: the problem dimension
X_{min}, X_{max}: decision variable search range
V_{min}, V_{max}: particle velocity limits
Output: x^*: the best particle position found (global best)
f^*: the best fitness value found
Initialize: **position** $x_i = (x_{i1}, \ldots, x_{id})$ and velocity $v_i = (v_{i1}, \ldots, v_{id})$, for all particles in problem space
evaluate $f(x_i)$ in d variables and get $pbest_i$, $(i = 1, \ldots, s)$
$gbest \leftarrow$ best of $pbest_i$
While stopping criteria is false **do**
Compute inertia weight (ω) if it is not a constant
Repeat for s times
Repeat for d times
update v_i for particle using (2)
validate for velocity boundaries using Algorithm 3
update x_i for particle using (3)
validate for position boundaries using Algorithm 2
compute $f(x_i)$
End Repeat for d
compute $f(x_i)$
obtain new $pbest_i$
If $f(x_i) < f(pbest_i)$ then $pbest_i \leftarrow x_i$
If $f(x_i) < f(gbest)$ then
$gbest \leftarrow x_i$
$f(gbest) \leftarrow f(x_i)$
end if
End Repeat for s
End while
$x^* \leftarrow gbest$
$f^* \leftarrow f(gbest)$
Return x^* and f^*
End PSO Algorithm

Figure 2.17. PSO pseudo code

Optimization starts from a feasible initial population, positions and velocities randomly distributed throughout the design space by a process called swarm initialization. The next step consists of evaluating the objective function values, using the design space positions. Afterwards the optimum particle position at each iteration and the global optimum particle position have to be updated. After the optimum particle position is updated the positions of all the other particles is updated as well. The last step concerns velocity vector update for each particle. As a result, optimization is terminated after convergence check.

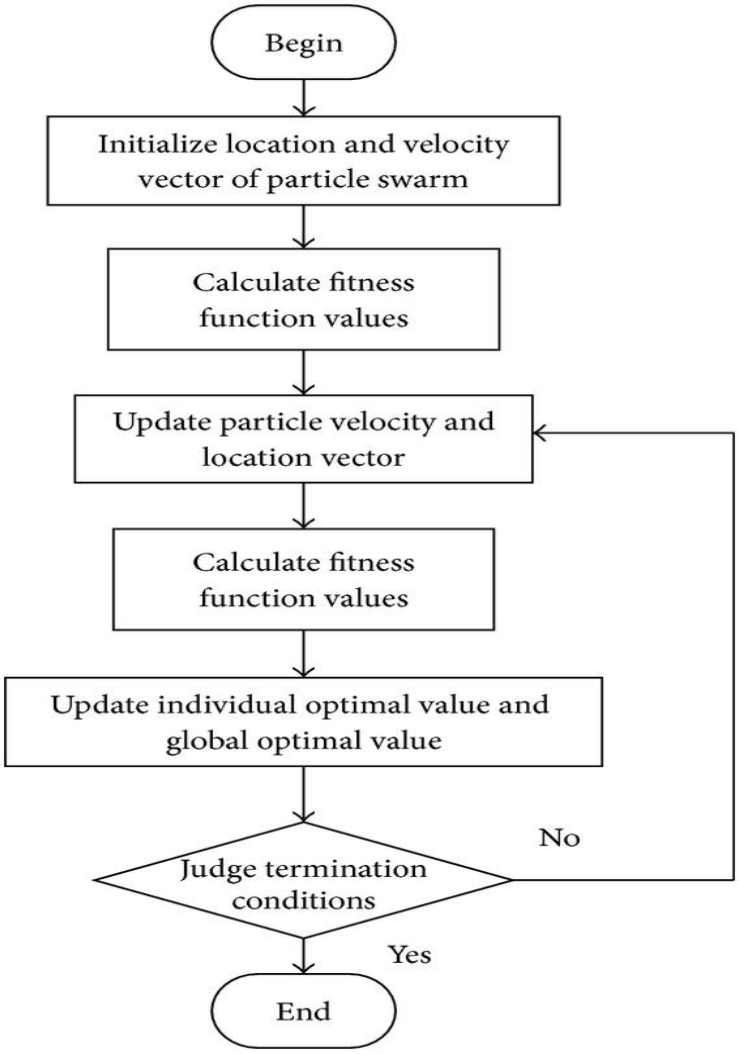


Figure 2.18. Flowchart of PSO algorithm

3. MATLAB BASED 2D TRUSS OPTIMIZATION

Chapter 3 is dedicated to show the plane trusses optimization using MATLAB program. Therefore, the use of MATLAB program will be briefly developed and the Finite Element Method (FEM), fundament of the structural analysis is also evoked. In total four case studies will be implemented and the results compared with previous studies after the whole program working pattern has been defined.

3.1. MATLAB

MATLAB is a high-level programming language and an interactive environment for numerical computations, graphics, and visualization that has been commercially available since 1984 and is widely used for teaching and research in industry and academia. Furthermore, MATLAB is modern programming software with an easy-to-use editing system, powerful data structures, customizable graphics, debugging tools that expresses problems and solutions in familiar mathematical notation. In addition to these previous factors that make MATLAB an excellent tool, it has many more advantages compared to conventional computer languages such as C, C++, Java or FORTRAN for solving technical problems. One of its distinguishing features is that It has literally hundreds of built-in functions and many toolboxes with concise Codes for solving specific problems, including statistics, partial differential equations, data analysis, optimization and several other fields of applied science and engineering. By using MATLAB, one can also develop algorithms, or create models and applications.

The name MATLAB stands for MATrix LABoratory, and the program was initially designed for matrix computations as well as providing easy access to matrix softwares LINPACK and EISPACK. MATLAB has grown greatly over its existence to become a general tool but at the beginning it was a system for solving matrices quickly and accurately. The storage of all data is made in matrix form and each single data refers to a scalar, a vector or a matrix. This makes the information manipulation easier and that is from where MATLAB picks its real power. MATLAB programming language is therefore useful in solving structural design problems implemented through the finite element method due to the existence of the extremely robust and vast predefined built-in functions.

Although MATLAB is very complete with respect to its mathematical functions some few finite element specific tasks need to be develop as additional functions. Before computing the finite elements method, one must know the fundaments of the technique and the different steps to implement.

3.2. Finite Element Method (FEM)

The finite element procedures are nowadays present and widely used as the most reliable computational tool, in science and engineering applications. The finite element literature is very large, either called Finite Element Analysis (FEA) or Finite Element Method (FEM), it is established as a powerful and popular analysis tool for solving numerical problems of continua related to solids, structures, heat transfer and fluids. The finite element procedure is based on mathematically defined differential equations and integral expressions that constitute simultaneous algebraic equations to be solved on a digital computer. In this study the term FEM is used to refer to Finite Element.

The FEM began with the advent of digital computer but it is rather difficult to give an exact date of invention. FEM as it is known today has its origin in a mathematical lecture delivered in 1941 and published in 1943. It was introduced by Courant (1943), that used the principle of stationary potential energy and piecewise polynomial interpolation to evaluate the Saint-Venant torsion by applying triangular sub-regions. The method was latter improved by engineers Argyris (1965) and Clough (1960). Since then many researchers have been devoted to the technique and large general-purpose computer programs emerged, ANSYS, ASKA, NASTRAN, ABAQUS.

The basic concept of the FEM is to solve a problem by discretizing it, in other words it's a method based on piecewise approximation functions defined over small and manageable pieces (element) obtained by dividing a complicated object. This process is called element mesh and the considerations made for it have a great importance regarding the effectiveness and the reliability of the results. A good discretization would typically lead to good solutions, while bad mesh topologies would lead to bad solutions.

The use of FEM in a structural analysis typically involves six (6) steps with the first one that has been developed above. The five (5) others are enumerated as follows:

- Evaluate each single element's stiffness matrix.

- Assemble all the matrices to get the global matrix of the whole structure.
- Apply the system's boundary conditions.
- Solve the global matrix equations obtained.
- Determine the secondary variables**.**

While considering FEM implementation in a computer program two (2) more steps are integrated. The first one should be executed after discretizing the model and the second one is the last step of computed program, there are both defined below:

- Read input data and allocate proper array sizes.
- Print or plot the desired results**.**

Some important steps that need more understanding are developed in the following lines.

3.2.1. Input Data

Generally, a truss structure possesses all the attributes necessary to illustrate a finite analysis without the need to proceed to the discretization of the model. Therefore, every bar of the structure has to be considered as an element. A member of a truss represents the simplest solid element, an elastic bar with two end nodes, referred as node 1 and 2 in Figure 3.1. These elements are connected by pin joints, only support the applied external nodal loads through axial forces and are defined by their length L, cross section A, and material's Young Modulus E as represented in Figure 3.1. a and b. Thereby after identifying the total number of elements in the system (nel) and the number of connectivity nodes (nnodes), the major input parameters needed to solve a two-dimensional steel truss are respectively inputted:

- The node coordinates: It is simply a matrix of 2×nnodes where the values of the nodes positions are stored.
- Element connectivity matrix: It is a matrix of node numbers according to the order they have been stored in the node coordinates matrix, where each line contains two rows representing the first node and the second node of an element.
- The nodal forces: They are stored in a matrix with the same dimension as node coordinates expressed by fx and fy.

The boundary conditions are input data as well but they will be developed latter.

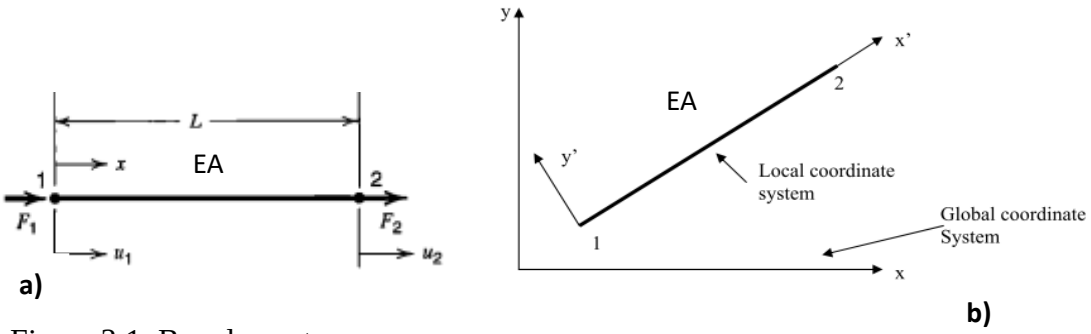


Figure 3.1. Bar element

3.2.2. Stiffness Matrix Evaluation

In most general cases, the elements of truss structures are considered to behave like linear elastic material that leads to the use of constant strain bar technique to extract the stiffness matrix with calculations based on the shape functions. The expression obtained is function of the length, area and elasticity modulus of each element.

$$[K] = \frac{EA}{L}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \tag{4}$$

Ultimately for all finite element programs the form of the linear algebraic system stated below is to be solved after collecting all the information.

$$[K]\{U\} = \{F\} \tag{5}$$

$$\begin{bmatrix} k & -k \\ -k & k \end{bmatrix}\begin{Bmatrix} u_1 \\ u_2 \end{Bmatrix} = \begin{Bmatrix} f_1 \\ f_2 \end{Bmatrix} \tag{6}$$

Equation (5) represents the shorter form while equation (6) is more detailed, and in both equations $[K]$ is the stiffness matrix with $k = \frac{EA}{L}$, $\{U\}$ represents the displacements vector with u_1 (respectively u_2) displacement at node 1 (respectively at node 2), $\{F\}$ is employed for the external nodal forces with f_1 (respectively f_2) applied force at node 1 (respectively at node 2).

The considerations made above in all the formula are only applicable to uniaxial bars conformably to Figure 3.1. a), in the case a truss bar configuration is similar to Figure 3.1. b) a transformation matrix is employed to convert the stiffness matrix from local coordinates to global coordinates.

$$T = \begin{bmatrix} c & s & 0 & 0 \\ 0 & 0 & c & s \end{bmatrix} \quad (7)$$

In expression (7) T is the transformation matrix with c = $\cos\theta$, and s = $\sin\theta$, θ is the angle between the bar and the horizontal.

The new stiffness matrix $[K_G]$ is calculated according to equation (8), while displacements and forces are decomposed as follows:

$$[K_G] = [T]^t[K][T] \quad (8)$$

$$U_1 = \begin{Bmatrix} u_1 \\ v_1 \end{Bmatrix}, f = \begin{Bmatrix} f_x \\ f_y \end{Bmatrix}$$

3.2.3. Structure Stiffness Matrix or Global Stiffness Matrix

Once, the elements matrices and vectors are computed, they need to be assembled into the global stiffness matrix and vector. To achieve that, the best method is to expand first the element stiffness matrices and vector to the full size of the structure. That means that if the structure's nnodes equals 5, each stiffness will be a 10x10 matrix filled with zeros (0) at the positions of the nodes its element doesn't include. The second step is just a regular addition operation of all the expanded matrices.

3.2.4. Boundary Conditions

In the finite element method, natural (Neumann) boundary conditions are used to form force vectors while essential (Dirichlet) boundary conditions are used to specify the value of the unknown field on a boundary state. In a more specific way, boundary considerations for truss structures are made on the fixed nodes. For plane trusses where the assembled $[K_G]$ matrix operates with u_1 and v_1, after applying the boundaries its dimension will reduce because of some columns and lines suppression. When some displacements are prescribed to be zero, the lines in the equation (6) containing that displacements are purely and simply deleted and the corresponding columns in the stiffness matrix also neglected. This is the formal way for solving manually reduced models, but it can also be used for computing when it comes to large scale problems. Other methods, as Lagrange multipliers and penalty, are preferred in many books to assign the boundaries in FEM computer codes.

3.3. Implementation

The entirety of this study is conducted using one computational tool, MATLAB, to solve the optimization study cases through automate processes of iterations. The process consists of two main parts. The first one performs the structural static analysis of the truss and the second one has the responsibility to execute optimization loop.

Practically the structural analysis performed in the first step, is governed by a script called “truss” that retrieves information it needs from some connected functions to come over expected results. The functions referred are named “inputdata”, containing nodal coordinates and connectivity, the elasticity modulus, and “loadforce” containing forces applied at the nodes.

The optimization algorithm will generate at its first steps some variables that will be assigned to the elements as cross section, in the “truss” script. When the results are ready, they are taken back to the optimization algorithm that evaluates the objective function, here represented by the structural weight. This process is repeated until the best weight is obtained.

3.4. Examples/ Case Studies

In this present case four (4) examples will be performed by employing three (3) different algorithms ABC, HM and PSO. The final results will be compared to demonstrate the efficiency of each approach by also having a look at some previous works conducted on the same topic. The parameters set for each optimization algorithm are as follows:

ABC: Number of Onlooker Bees nOnlooker=nPop, Abandonment Limit Parameter L=round(0.6*nVar*nPop), Acceleration Coefficient Upper Bound a=1. Maximum Number of Iterations (MaxIt) and Population Size (nPop) are varying according to the example but most of the time the values in parenthesis are used.

HMS: Harmony Memory Consideration Rate HMCR=0.9, Pitch Adjustment Rate PAR=0.1, Fret Width (Bandwidth) FW=0.02*(VarMax-VarMin), Width Damp Ratio FW_damp=0.995. Maximum Number of Iterations (MaxIt), Harmony Memory Size (HMS) and Number of New Harmonies (nNew) are varying according to the example but most of the time the values in parenthesis are used.

PSO: Inertia Weight w=1, Inertia Weight Damping Ratio wdamp=0.99, Personal Learning Coefficient c1=1.5, Global Learning Coefficient c2=2.0. Maximum Number of Iterations (MaxIt) and Population Size (nPop) are varying according to the example but most of the time the values in parenthesis are used.

3.4.1. Ten-bar (10) Truss Structure

The 10-bar truss structure has been one of the most popular test problems in the field of structural design for evaluating and verifying the efficiency of many different optimization methods. Therefore, many researchers have worked on the topic, such as, Camp et al. (1998), Camp and Farshchin (2014), Ringertz (1988), Rajeev and Krishnamoorthy (1992), Li et al. (2009). The geometry, support conditions and loading conditions for this 2-dimensional truss example are shown in Figure 3.2. The density of the material is 2767,990 kg/m^3 (0,1 lb/in ^{3}) and the elasticity modulus is 68 950 MPa (10 000 ksi). The allowable vertical and horizontal displacements for all nodes are equal to ±5,08 cm (2,0 in). The members stress limitations in both tension and compression are equal to ± 172,375 MPa (25 ksi). There is one load combination that is considered for the two cases in this example, $P_1 = 444{,}82$ kN (100 kips), $P_2 = 0$. The 10 design variables considered have to be picked from the variable range below:

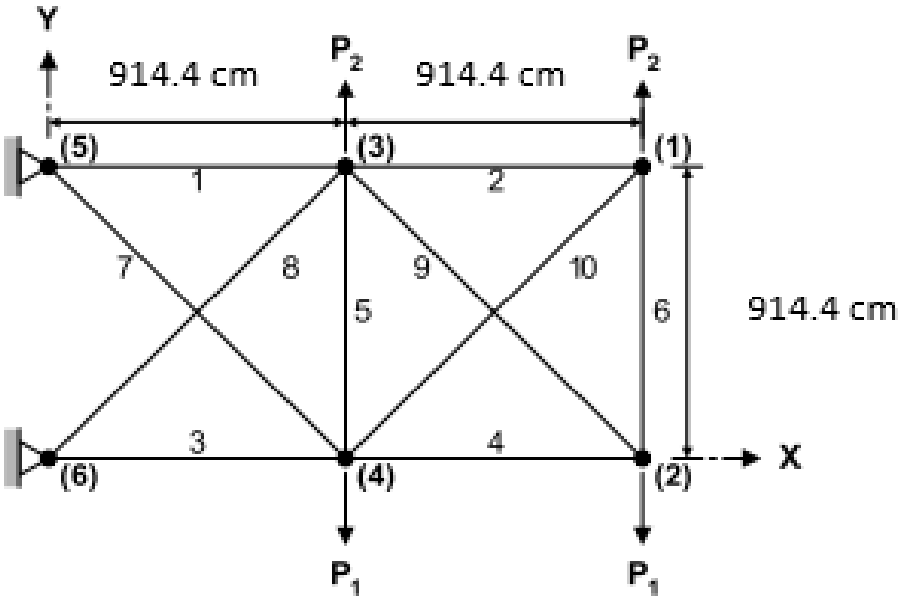


Figure 3.2. 10-bar planar truss structure

Case 1 is considered as a discrete optimization with the variable set:

D = [1.62, 1.80, 1.99, 2.13, 2.38, 2.62, 2.63, 2.88, 2.93, 3.09, 3.13, 3.38, 3.47, 3.55, 3.63, 3.84, 3.87, 3.88, 4.18, 4.22, 4.49, 4.59, 4.80, 4.97, 5.12, 5.74, 7.22, 7.97, 11.50, 13.50, 13.90, 14.20, 15.50, 16.00, 16.90, 18.80, 19.90, 22.00, 22.90, 26.50, 30.00, 33.50] (in^2).

Case 2 is considered differently by two previous studies that bring out the same result. In the first one, it is solved as a continuous optimization with cross-sectional areas varying from 0.1 in^2 to 35.0 in^2, whereas in the second one it is considered as discrete optimization with the set of variables below.

D = [0.1, 0.5, 1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0, 4.5, 5.0, 5.5, 6.0, 6.5, 7.0, 7.5, 8.0, 8.5, 9.0, 9.5, 10.0, 10.5, 11.0, 11.5, 12.0, 12.5, 13.0, 13.5, 14.0, 14.5, 15.0, 15.5, 16.0, 16.5, 17.0, 17.5, 18.0, 18.5, 19.0, 19.5, 20.0, 20.5, 21.0, 21.5, 22.0, 22.5, 23.0, 23.5, 24.0, 24.5, 25.0, 25.5, 26.0, 26.5, 27.0, 27.5, 28.0, 28.5, 29.0, 29.5, 30.0, 30.5, 31.0, 31.5] (in^2).

Results of the current study and previous studies are compared in Table 3.1. and 3.2., and Figure 3.3. a) to c), Figure 3.4. a) to c) show the convergence history of ABC, HM and PSO algorithms.

Table 3.1. Optimal design for 10-bar planar truss structure (Case 1)

Variables		Previous studies				This study		
(in^2)		GA	PSOPC	HPSO	TLBO	ABC	HS	PSO
A_1		33.00	30.00	30.00	33.50	33.50	30.00	30.00
A_2		1.62	1.80	1.62	1.62	1.62	1.62	1.62
A_3		22.00	26.50	22.90	22.90	22.90	22.00	22.90
A_4		15.50	15.50	13.50	14.20	14.20	16.00	16.90
A_5		1.62	1.62	1.62	1.62	1.62	1.62	1.62
A_6		1.62	1.62	1.62	1.62	1.62	1.62	1.62
A_7		14.20	11.50	7.97	7.97	7.97	11.50	7.97
A_8		19.90	18.80	26.50	22.90	22.90	22.00	22.90
A_9		19.90	22.00	22.00	22.00	22.00	22.00	22.90
A_{10}		2.62	3.09	1.80	1.62	1.62	1.62	1.62
Weight	(lb)	5613,84	5593,44	5531,98	5490,74	5490,74	5531,04	5507,76
	(kg)	2546,39	2537,1	2509,26	2490,56	2490,56	2508,84	2498,28

Table 3.2. Optimal design for 10-bar planar truss structure (Case 2)

Variables		Previous studies				This study		
(in^2)		Ringertz	PSOPC	GA	TLBO	ABC	HS	PSO
A_1		30.50	25.50	28.92	30.6684	31.60	31.20	30.50
A_2		0.10	0.10	0.10	0.10	0.33	0.10	0.10
A_3		23.00	23.50	24.07	23.1584	24.10	25.20	23.20
A_4		15.50	18.50	13.96	15.2226	14.50	14.60	15.30
A_5		0.10	0.10	0.10	0.10	0.15	0.10	0.10
A_6		0.50	0.50	0.56	0.5421	0.33	0.34	0.55
A_7		7.50	7.50	7.69	7.4654	8.29	7.75	7.49
A_8		21.00	21.50	21.95	21.0255	20.90	20.70	21.30
A_9		21.50	23.50	22.09	21.4660	21.00	20.60	21.20
A_{10}		0.10	0.10	0.10	0.10	0.11	0.10	0.10
Weight	(lb)	5059,90	5133,16	5076,31	5060,97	5116,20	5078,53	5061,43
	(kg)	2295,13	2328,36	2302,58	2295,62	2320,67	2303,58	2295,83

The results of case 1 show that TLBO algorithm had the best performance with a total weight of 2490,56 kg (5490,74 lb). The best result obtained in this study is performed by the ABC algorithm with a structural weight of 2490,56 kg (5490,74 lb). The parameters used for the runs are respectively, MaxIt=1000 and nPop=300 for ABC, MaxIt=1000, HMS=15 and nNew=10 for HS and finally, MaxIt=500 and nPop=25 for PSO. More importantly, the PSO algorithm exhibits improved computational efficiency when compared to other heuristic methods of this study because with a small MaxIt and nPop, it provides a result closed to the best one produced by ABC algorithm.

The results of case 2 reveal that the lightest structure is generated by the Ringertz implementation and brings out a total weight of 2295,13 kg (5059,9 lb) comparatively to the present study's lightest structure that have a weight of 2295,83 kg (5061,43 lb) performed by PSO algorithm. The parameters used for the runs are respectively, MaxIt=1000 and nPop=300 for ABC, MaxIt=500, HMS=50 and nNew=50 for HS and finally, MaxIt=500 and nPop=50 for PSO. Here again PSO algorithm is the most efficient between all employed algorithms.

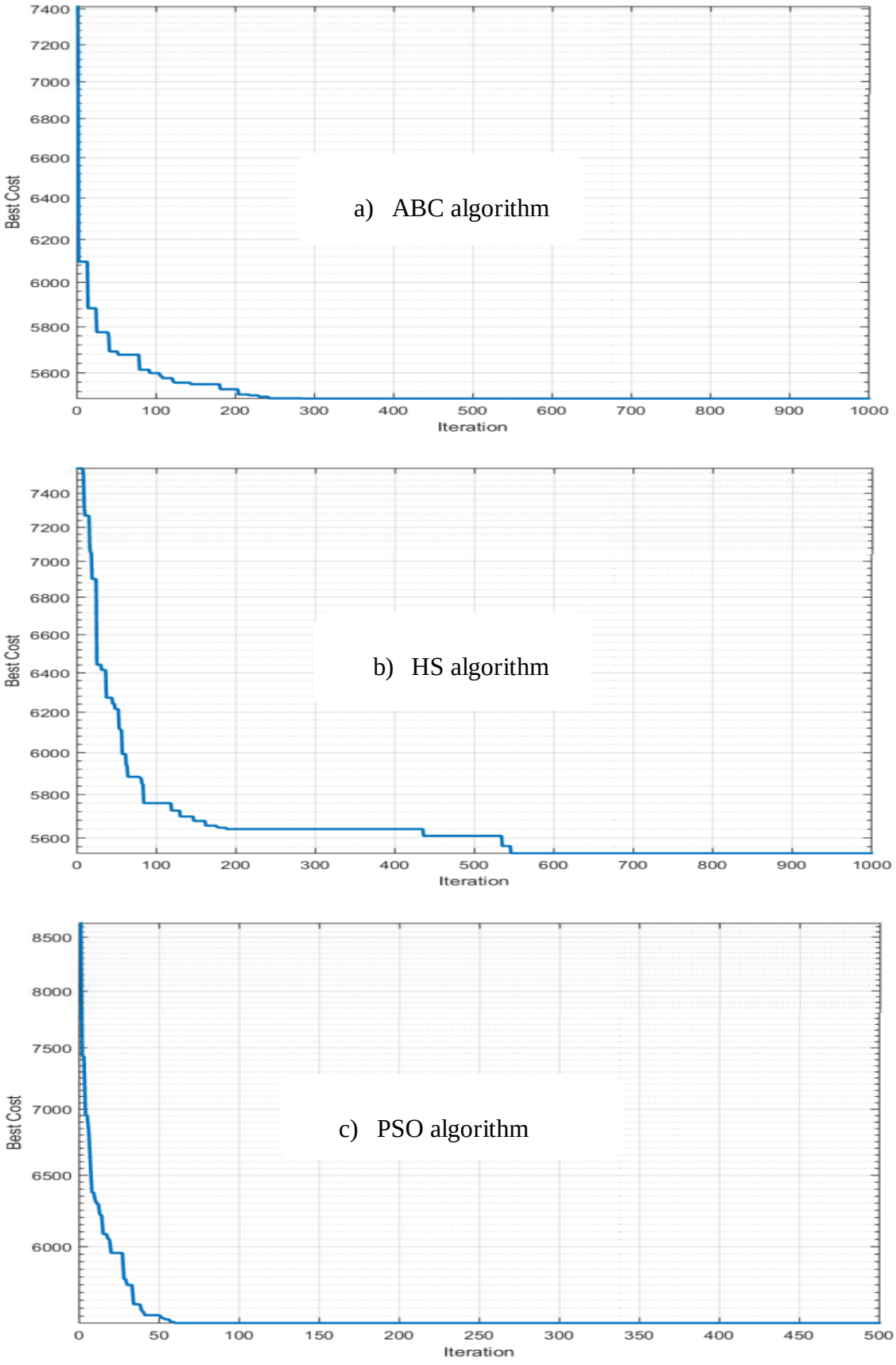


Figure 3.3. Convergence histories for 10-bar planar truss structure (case 1)

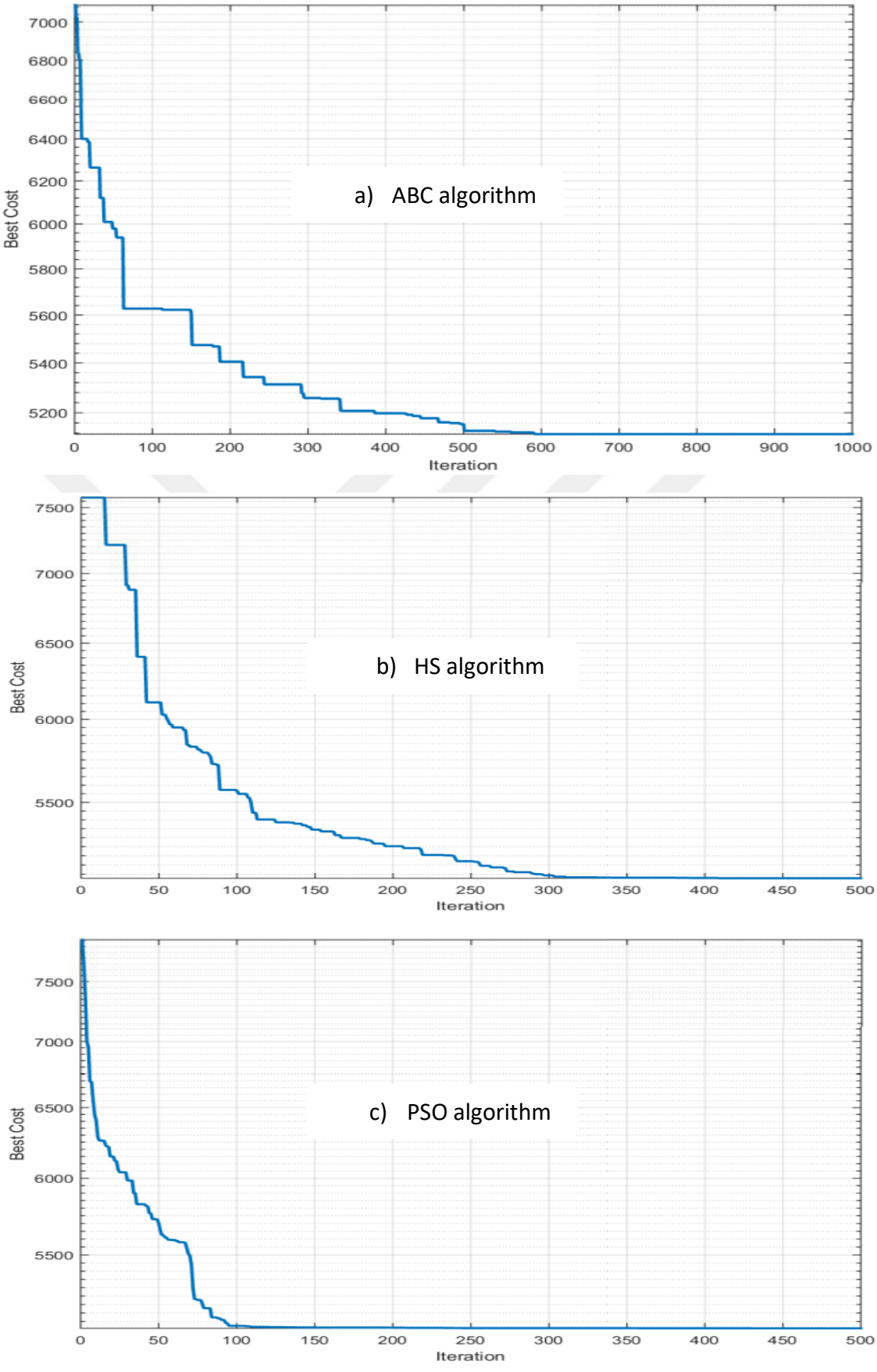


Figure 3.4. Convergence histories for 10-bar planar truss structure (case 2)

3.4.2. Seventeen-bar (17) Truss Structure

The 17-bar truss structure, with the detailed geometry in Figure 3.5., is the second weight optimization example to be performed. Only a single vertical load of 444,82 kN (100 kips) is applied at the extreme free node. The material density is 7418,214 kg/m^3 (0,268 lb/in^3) and the elasticity modulus is 20 6842,8 MPa (30 000 ksi). Stress limitations of $\pm$ 344,738 MPa (50 ksi) are subjected to all members and displacements of the free nodes in both horizontal and vertical directions should not exceed to $\pm$ 5.08 cm (2.0 in). This problem is a continuous optimization that 17 variables, generated randomly by the algorithm which the minimum allowable value of 0.6452 cm^2 (0.1 in^2). Previous studies have been done by Khot and Berke (1984) using Optimality Criteria (OC), Li et al. (2007) with Heuristic Particle Swarm Optimizer (HPSO), Koohestani and Kazemzadeh (2009) Adaptive Real-Coded Genetic Algorithm (ARCGA) and finally Modified Artificial Bee Colony (MABC) applied by Hadidi et al. (2010).

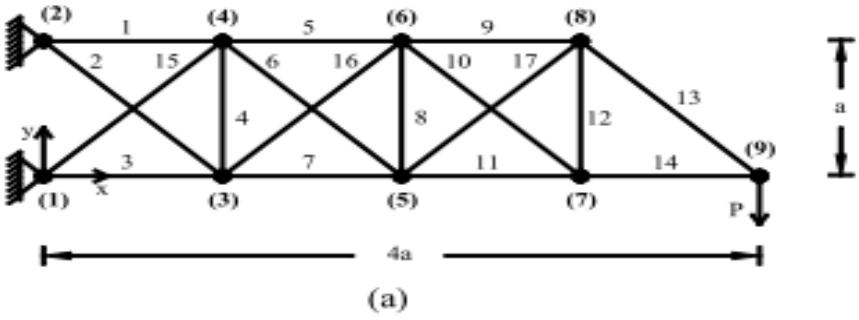


Figure 3.5. 17-bar planar truss structure, a = 254cm (100 in)

Table 3.3. shows the optimal results of 7 algorithms, while Figure 3.6. a) to c) are the representation of the convergence history of ABC, HM and PSO.

The optimization results obtained by both previous and current studies presented in Table 3.3. confirm that, all previous results are nearly the same with a rounded weight of 1171,18 kg (2582 lb), while in present study PSO yields lighter structural weight than other methods with the same value of 1171,18 kg (2582 lb). Here again the parameters employed for ABC are MaxIt=1000 and nPop=300, while for HS it is MaxIt=500, HMS=200 and nNew=200 and finally, for PSO MaxIt=500 and nPop=100.

Table 3.3. Optimal design for 17-bar planar truss structure

Variables		Previous studies				This study		
(in^2)		OC	MABC	HPSO	ARCGA	ABC	HS	PSO
A_1		15.93	15.6762	15.896	15.891	15.00	15.30	16.00
A_2		0.10	0.10	0.103	0.105	0.66	1.56	0.10
A_3		12.07	12.0491	12.092	12.101	12.90	13.20	12.10
A_4		0.10	0.10	0.10	0.10	0.10	0.10	0.10
A_5		8.067	8.1312	8.075	8.075	8.78	8.91	8.08
A_6		5.562	5.6202	5.541	5.541	4.58	4.11	5.58
A_7		11.933	11.8822	11.97	11.97	12.00	11.50	12.00
A_8		0.10	0.10	0.10	0.10	0.17	0.32	0.10
A_9		7.945	8.0517	7.955	7.955	7.60	7.63	7.92
A_{10}		0.10	0.10	0.10	0.10	0.57	0.93	0.10
A_{11}		4.055	4.0912	4.07	4.07	4.28	4.85	4.11
A_{12}		0.10	0.10	0.10	0.1	0.83	0.59	0.10
A_{13}		5.657	5.6746	5.67	5.705	5.83	5.46	5.60
A_{14}		4.00	3.9864	3.998	3.975	3.88	4.10	4.02
A_{15}		5.558	5.6729	5.548	5.516	5.06	4.06	5.51
A_{16}		0.10	0.10	0.103	0.10	0.56	1.43	0.10
A_{17}		5.579	5.4907	5.537	5.563	5.26	4.76	5.53
Weight	(lb)	2581,89	2582,27	2581,94	2581,95	2611,90	2630,00	2582,00
	(kg)	1171,1	1171,3	1171,1	1171,15	1184,74	1192,95	1171,18

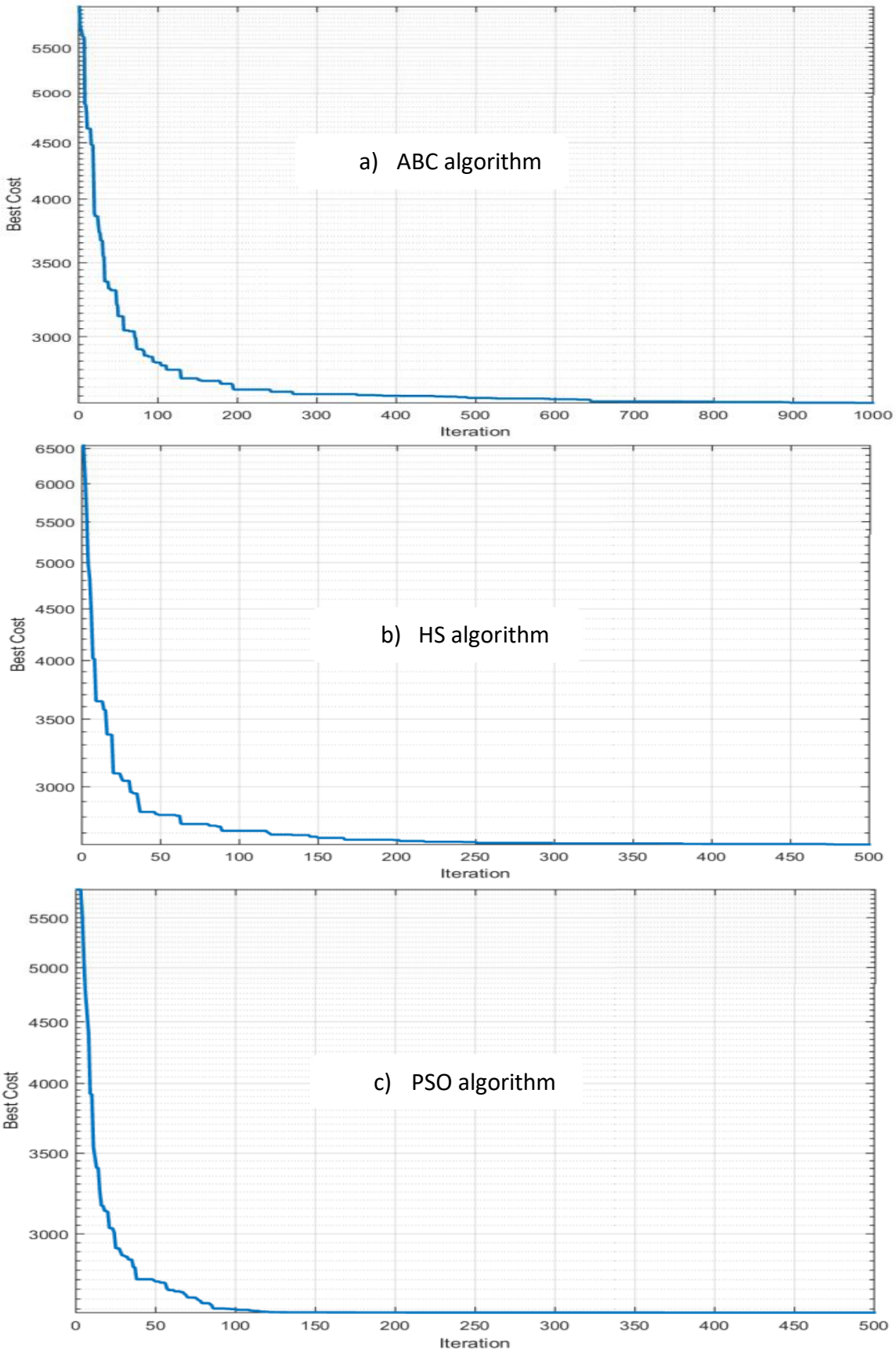


Figure 3.6. Convergence histories for 17-bar planar truss structure

3.4.3. Forty-five-bar (45) Truss Structure

Figure 3.7. shows the geometry and the loading condition of a 45-bar that has been studied by Hadidi et al. (2010) for further investigation of the performance of MABC algorithm. Nine vertical loads of 44,48 kN (10 kips) are applied at the unrestricted bottom nodes. The displacement of all nodes in both horizontal and vertical directions must not exceed $\pm$ 5,08 cm (2,0 in) and the maximum absolute value of each element's stress is limited to 206,843 MPa (30 ksi). The material density is 7833,41 kg/m^3(0,283 lb/in^3) and the modulus of elasticity is 20 6842,8 MPa (30 000 ksi). This example deals with 23 design variables that have been grouped according to the symmetry of the structure. The lower bound of all variables considered in this optimization is 0,6452 cm^2 (0,1 in^2).

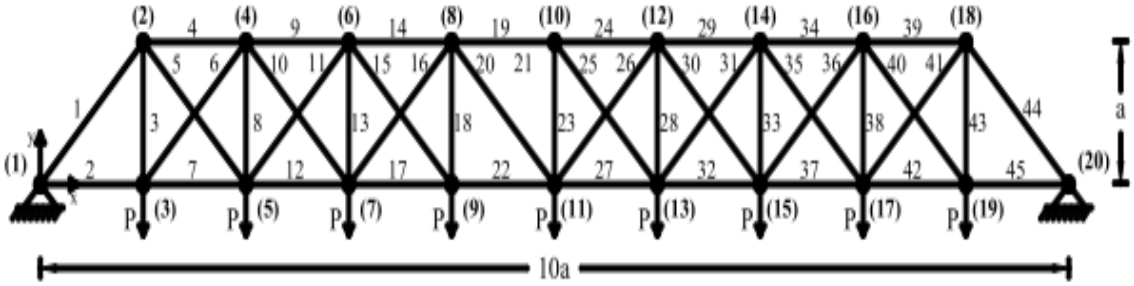


Figure 3.7. 45-bar planar truss structure, a = 508cm (200 in)

Table 3.4. shows the optimal results of employed algorithms, conformably to their convergence histories materialized in Figure 3.8. a) to c).

The results listed in Table 3.4. indicate that the lighter structure obtained in this study has come from PSO iterations and its weight is very closed the best design found by MABC. The weight difference between these two is less than 1 kg (2 lb). The parameters used to perform the study are MaxIt=500 and nPop=100 for ABC, MaxIt=500, HMS=100 and nNew=100 for HS and finally, MaxIt=500 and nPop=100 for PSO.

Table 3.4. Optimal design for 45-bar planar truss structure

Variables		Previous study		This study		
(in^2)		ABC	MABC	ABC	HS	PSO
A_1		5.4746	4.5996	5.0200	4.4600	4.5500
A_2		4.5989	3.7966	4.0100	4.2000	3.8100
A_3		4.1703	3.0497	3.1600	2.0700	2.0900
A_4		3.7872	3.2841	3.8300	4.6100	4.3400
A_5		0.10	0.1069	0.6830	1.8000	1.5900
A_6		4.1735	3.9279	3.6800	2.4200	2.4400
A_7		0.9497	0.9649	1.1700	2.4500	2.0500
A_8		1.5902	1.2133	0.9650	0.3030	0.1930
A_9		6.2656	7.6553	6.7600	7.3000	7.5900
A_{10}		2.2039	2.1993	1.6100	1.8900	2.1500
A_{11}		1.3925	1.1929	1.8300	1.4400	1.2500
A_{12}		0.10	0.1001	0.1380	0.10	0.1010
A_{13}		0.10	0.1008	0.2850	0.5770	0.10
A_{14}		9.0689	9.5360	10.1000	9.2300	9.3300
A_{15}		1.5310	1.2173	1.9800	1.4700	1.2700
A_{16}		1.6245	1.4190	0.9190	1.3800	1.3300
A_{17}		2.9146	2.5513	2.3000	2.3200	2.5600
A_{18}		0.10	0.10	0.2370	0.10	0.10
A_{19}		9.0685	11.5439	11.6000	11.4000	11.8000
A_{20}		1.6352	1.2807	1.0800	1.3300	1.2700
A_{21}		0.10	0.101	0.3080	0.10	0.10
A_{22}		4.4798	3.7598	3.82000	3.3700	3.7100
A_{23}		0.10	0.10017	0.7780	0.2000	0.10
	(lb)	8267,21	7968,95	8256,10	8097,20	7970,30
Weight	(kg)	3749,94	3614,65	3744,90	3672,83	3615,27

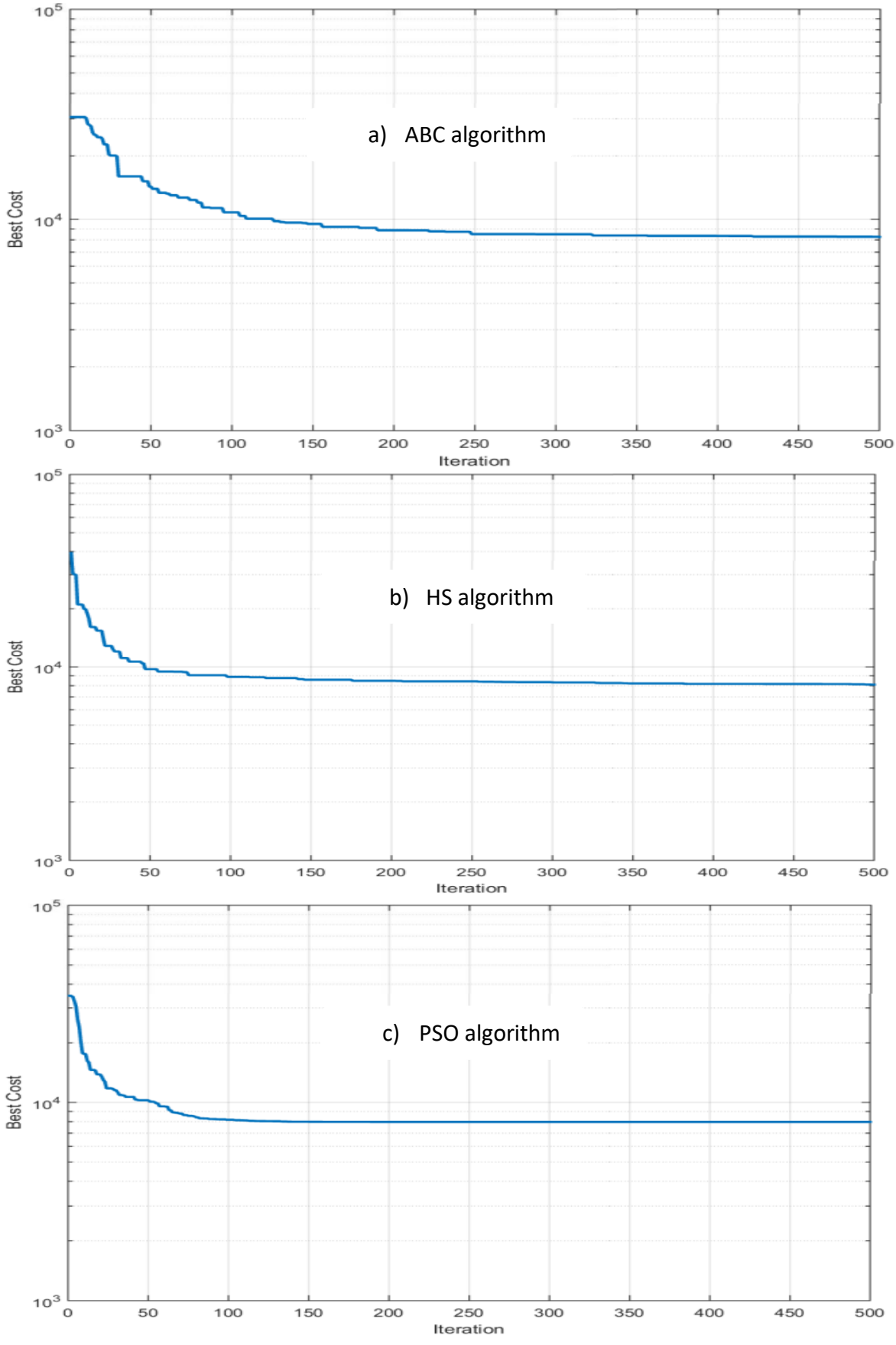


Figure 3.8. Convergence histories for 45-bar planar truss structure

3.4.4. Fifty-two-bar (52) Truss Structure

Figure 3.9. is the geometrical representation of a 52-planar bar structure. This problem has been studied previously by Wu and Chow (1995), Lee at al. (2005), and Li et al. (2009). Vertical and horizontal loads, respectively 100 kN and 200 kN are applied at the four top nodes of structure. While no displacement constraint is considered, stress constraint is applied with the maximum absolute value of each element's stress limited to 180 MPa. The material density is 7860.00 kg/m^3 and the elasticity modulus is 2.07×10^5 MPa. The structural elements have been classified into 12 design variables groups as A 1 (1 –4), A 2 (5 – 10), A 3 (11 – 13), A 4 (14 – 17), A 5 (18 – 23), A 6 (24 – 26), A 7 (27 – 30), A 8 (31– 36), A 9 (37 – 39), A 10 (40 – 43), A 11 (44 – 49), A 12 (50 – 52). The discrete variables are selected from the set of D matrix.

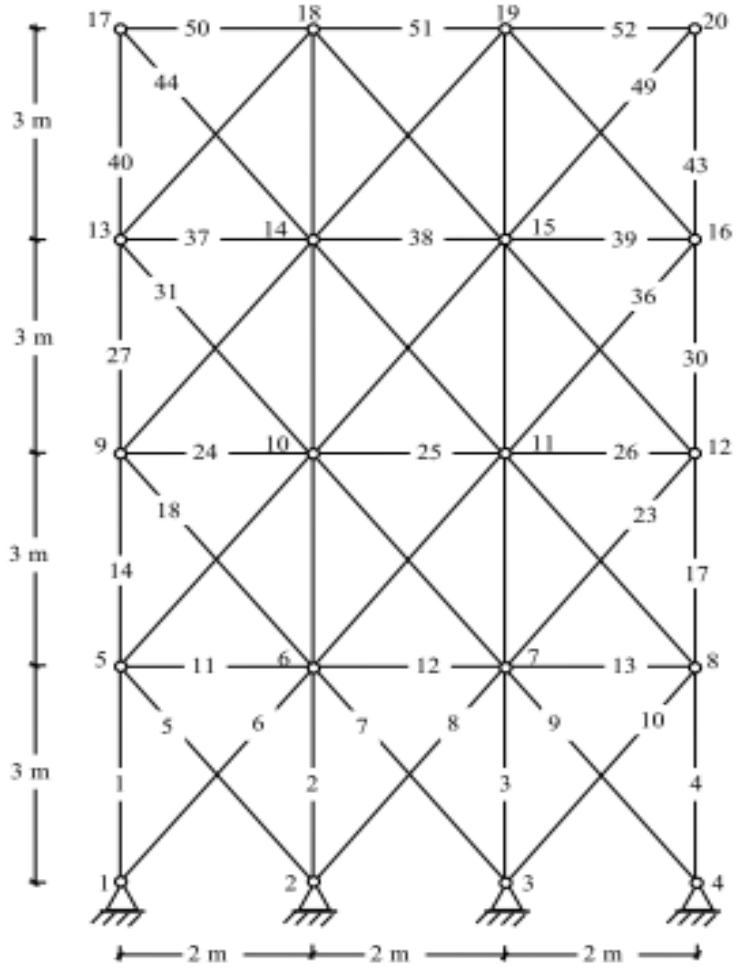


Figure 3. 9. 52-bar planar truss structure

D = [0.111, 0.141, 0.196, 0.250, 0.307, 0.391, 0.442, 0.563, 0.602, 0.766, 0.785, 0.994, 1.000, 1.228, 1.266, 1.457, 1.563, 1.620, 1.800, 1.990, 2.130, 2.380, 2.620, 2.630, 2.880, 2.930, 3.090, 3.130, 3.380, 3.470, 3.550, 3.630, 3.840, 3.870, 3.880, 4.180, 4.220, 4.490, 4.590, 4.800, 4.970, 5.120, 5.740, 7.220, 7.970, 8.530, 9.300, 10.850, 11.500, 13.500, 13.900, 14.200, 15.500, 16.000, 16.900, 18.800, 19.900, 22.000, 22.900, 24.500, 26.500, 28.000, 30.000, 33.500](in^2).

In Table 3.5. the optimal results are concise, and their convergence histories displayed in Figure 3-10 a) to c). When results of previous studies are compared to these present ones, it shows that the lightest structure is delivered by PSO and ABC with a corresponding weight that is 1902.88 lb. The corresponding parameters are respectively, MaxIt=500 and nPop=100 for ABC, MaxIt=500, HMS=50 and nNew=50 for HS and finally, MaxIt=200 and nPop=100 for PSO.

Table 3.5. Optimal design for 52-bar planar truss structure

Variables	Previous studies				This study		
(in^2)	DHPSACO	PSOPC	HPSO	GA	ABC	HS	PSO
A_1	7.220	9.300	7.220	7.220	7.220	7.220	7.220
A_2	1.800	1.563	1.800	1.800	1.800	1.800	1.800
A_3	0.766	4.180	0.563	1.00	0.766	0.563	0.766
A_4	5.120	4.970	5.120	5.120	5.120	5.120	5.120
A_5	1.563	1.800	1.457	1.620	1.460	1.460	1.460
A_6	0.442	1.228	0.766	0.766	0.766	0.766	0.766
A_7	3.550	3.470	3.470	3.840	3.470	3.470	3.470
A_8	1.563	1.563	1.563	1.620	1.560	1.560	1.560
A_9	0.602	0.766	0.602	0.442	0.766	0.766	0.766
A_{10}	1.990	1.990	1.990	2.630	1.990	1.990	1.990
A_{11}	1.800	1.800	1.800	1.620	1.800	1.800	1.800
A_{12}	0.785	0.766	1.228	0.994	0.766	0.994	0.766
Weight (kg)	1904.830	2146.630	1905.495	1970.142	1902.88	1903.64	1902.88

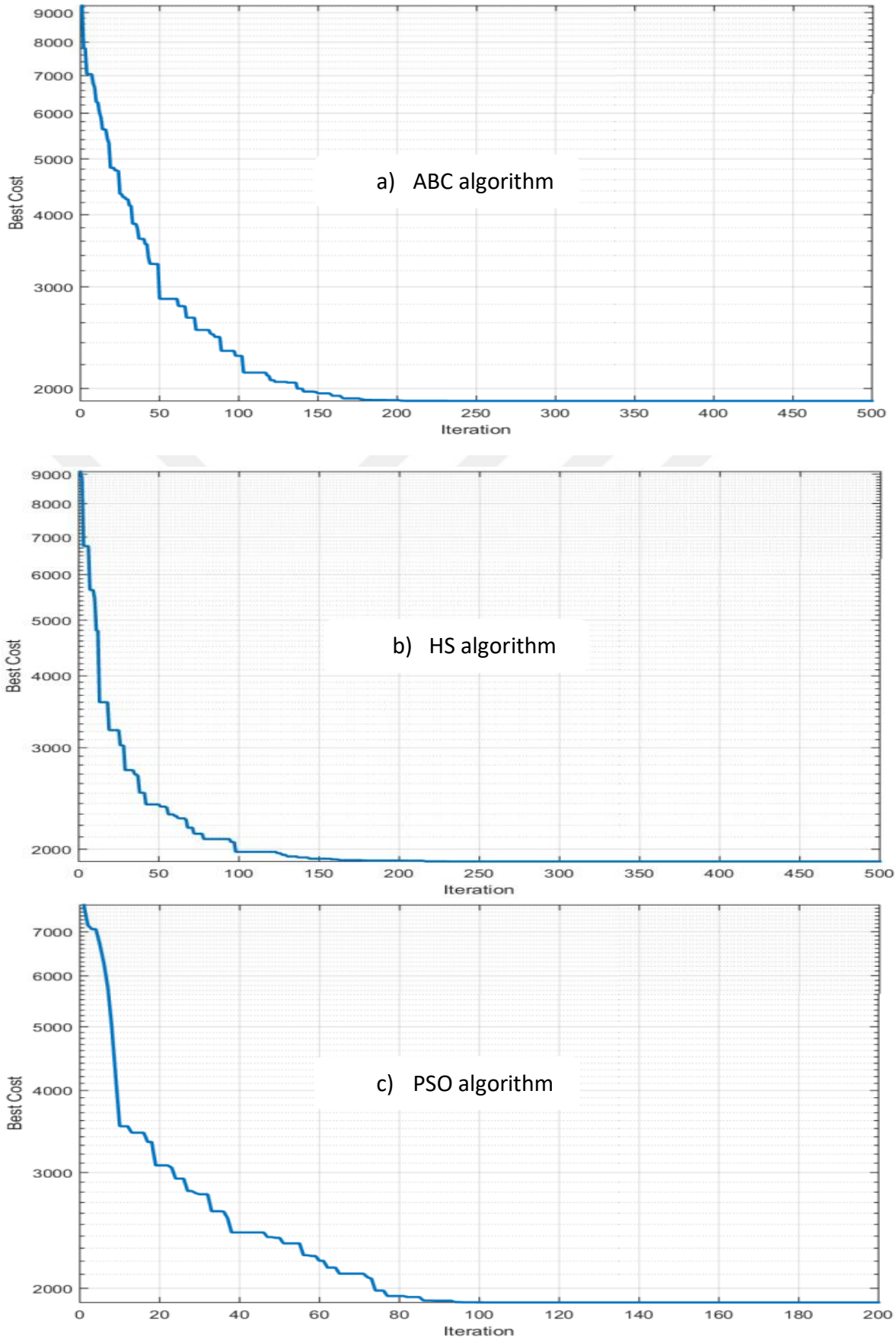


Figure 3.10. Convergence histories for 52-bar planar truss structure

4. OAPI BASED 3D TRUSS OPTIMIZATION

The combination of SAP2000 structural analysis program and MATLAB is the prior task of this chapter. The Open Application Program Interface (OAPI) of SAP200 that allows its biding with MATLAB is clearly explained and the implementation process reviewed. This study will finally compare then the results generated by the employed algorithms.

4.1. SAP2000

SAP2000 is the first version of SAP series programs having a complete integration with Microsoft Windows. Since its release date over 30 years ago, it has proven to be one of the most sophisticated and user-friendly computer programs. Since the first programs published, SAP, SOLIDSAP, SAPIV and so on, the name SAP was defined to be state-of-the-art analytical solutions. Over the years SAP2000 program has been strengthened with some futures such as a powerful graphical user interface powered by an ease-of-use engine, powerful design capabilities and finally easy and efficiency solution productivity for all structural analysis and design purposes, to the benefit of thousands of engineering companies around the world.

SAP2000 is a structural program that refers to the Finite Element Method for analysing civil engineering structures with integrated advanced analytical techniques. Its intuitive user interface and multitude of tools are very helpful for fast and accurate structural model construction without long learning detour delays. Not only the model creation but also its probable modifications, analysis run, design checking, and results output generation are made possible thanks to this single interface. The worldwide design codes integrated, allow the user to automatically generate snow, wind, or seismic loads with the possibility to check the design performance applying for instance US steel standards or Eurocodes for a given steel structure. SAP2000 is object-based program, therefore analysing an element in SAP2000 is a huge task divided into small parts possesses as follows:

- Geometry: The model representation.
- Material: Constitutive law.

- Essential boundary conditions considered as supports.
- Natural boundary conditions standing for applied Loads.

After the Structural analysis has been done and the design performed, all SAP2000 data, including model prescriptions, analysis and design results, are available through tabular data structure. These data can either be displayed in the interface by the engineer to determine the reliability of the results, or exported to a Microsoft Excel spreadsheet file, a Microsoft Access database file, or a simple text file. A model can also be opened into SAP2000 for further modifications or analysis by importing data file into the program. SAP2000 has many Import and export capabilities, especially for CSI programs but also for other well-known design programs as well as programming tools such as MATLAB.

4.2. Open Application Programming Interface (OAPI)

The fast evolution of computer technology and the large availability of increasing computational systems have motivated researchers to conduct some investigations on new structural optimization techniques that will reduce the long and tedious work of engineers, therefore avoid writing thousands of lines in a single script. This recently led to the introduction of the SAP2000 Open Application Programming Interface (OAPI), which allows a developer to create a rich and tight two-way links with SAP2000 and an external programming language. This tool has the responsibility to transfer model information from external application to SAP2000, analyse and design model, and finally extract the results back to the third-party application. Clearly defined, OAPI is a software library constituting the functions responsible of controlling SAP2000 from a different software.

The SAP2000 OAPI is a powerful tool which main aim is to provide efficient access to the analysis and design technology of the SAP2000 structural analysis software. SAP2000 can then be associated with most of the languages. The functioning mode of this process is represented by Figure 4.1. (Sextos and Balafas, 2011). An interesting part of this innovation in computational engineering is that some plug-ins can be developed to extend the program's usability. The OAPI features are enumerated as follows (Sextos and Balafas, 2011):

- A robust imbedding of SAP2000 in third-party application is direct and very fast.
- Both pre and post-processing routines are facilitated by the two-way data flow.

- Data exchange time is considerably reduced on large model implementation using most major programming languages, because no intermediate files are needed.
- Many third-party applications can ensure the data transfer and analysis control.
- An easy development and guaranty of third-party applications compatibility with future releases of SAP2000.
- Development of a calibrated custom interfaces for SAP2000 that feats the user's needs.

Developing a new computing tool that employs the above SAP2000 OAPI features is a laborious task requiring solid programming background. However, it may be sometimes less tedious when the necessity is based on simple computing tasks. OAPI is compatible with a wide range of programming languages, including Visual Basic Applications (VBA), Visual Basic.NET, Visual C#, Visual C++, Fortran and MATLAB. As a general assistant for the SAP2000 OAPI, a concise and detailed documentation is provided along with the installation (CSi_OAPI_Documentation, located in the install directory) containing the information that will help someone to get used to the programming.

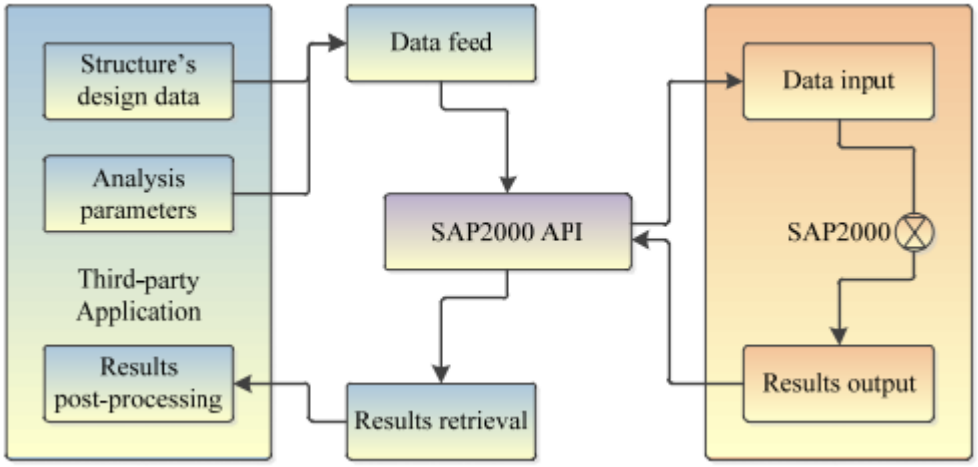


Figure 4.1. Typical data flow using the SAP2000 API

4.3. Implementation

The 3D steel truss examples studied in this chapter are implemented by binding two modern programs known as MATLAB Ra2015 and SAP2000 version V15. The structural analysis part is performed by SAP2000 that latter transfers the results to MATLAB for the

optimization process. This is repeated as many times as necessary to obtain the optimum way of each run.

There are several ways to do structural analysis in MATLAB Trough SAP2000 software. The first possibility is to write all structural information in MATLAB, concretely OAPI functions will govern the study from elements modelling to retrieving results, without any need to perform a single step by opening SAP2000 program. The second one consists in reproducing the entire model in SAP2000, after application of material proprieties, node supports and loads, the model is saved as a ". sdb file". For every iteration process MATLAB will just open the model file, run the analysis and extract results. In this thesis the second programming solution has been preferred to the first one as it is less complex, faster and makes shorter program script.

The design variables generated by the optimization algorithm will be assigned as elements cross section, in SAP2000. When the results are ready, the optimization algorithm evaluates the objective function and displays the best weight of the structure.

4.4. Examples/ Case Studies

In this present case tree (3) examples will be performed by employing tree (3) different algorithms ABC, HM and PSO. The final results of this study are compared with some previous works to demonstrate the efficiency of each approach. The parameters set for the optimization algorithms are similar to what has been defined in the previous chapter.

4.4.1. Twenty-five-bar (25) Spatial Truss Structure

Figure 4.2. represents a 25-bar transmission tower that has been studied in this example. In available literature, WU and Chow (1995), Rajeev and Krishnamoorthy (1992), Ringertz (1988), has considered this structure for research purpose employing some innovative algorithms. The material used in the model construction has a mass density of 2767,990 kg/m^3 (0,1 lb/in^3) and a modulus of elasticity E = 68 950 MPa (10 000 ksi). Stresses constrains are applied for each member with an allowable value of ± 275,8 MPa (40 000 psi) and displacements for each node are constrained in x, y, and z directions with limit value of ± 0,889 cm (0,35 in). The 25 elements of the structure are organized into eight (8) groups, considering symmetries with respect to both the y-z and the x-z

planes. These groups are as follows: A 1 (1), A 2 (2 – 5), A 3 (6 – 9), A 4 (10 – 11), A 5 (12 – 13), A 6 (14 – 17), A 7 (18 – 21), and A 8 (22 – 25). The discrete cross-sectional areas are selected from the set D while the load condition is showed in Table 4.1.

D = [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2.0, 2.1, 2.2, 2.3, 2.4, 2.5, 2.6, 2.8, 3.0, 3.2, 3.4] (in^2).

Table 4.1. Load case for the 25-bar spatial truss

Nodes	Load case (kN)		
	Px	Py	Pz
1	4,448	(-)44,48	(-)44,48
2	0.0	(-)44,48	(-)44,48
3	2,22	0.0	0.0
6	2,69	0.0	0.0

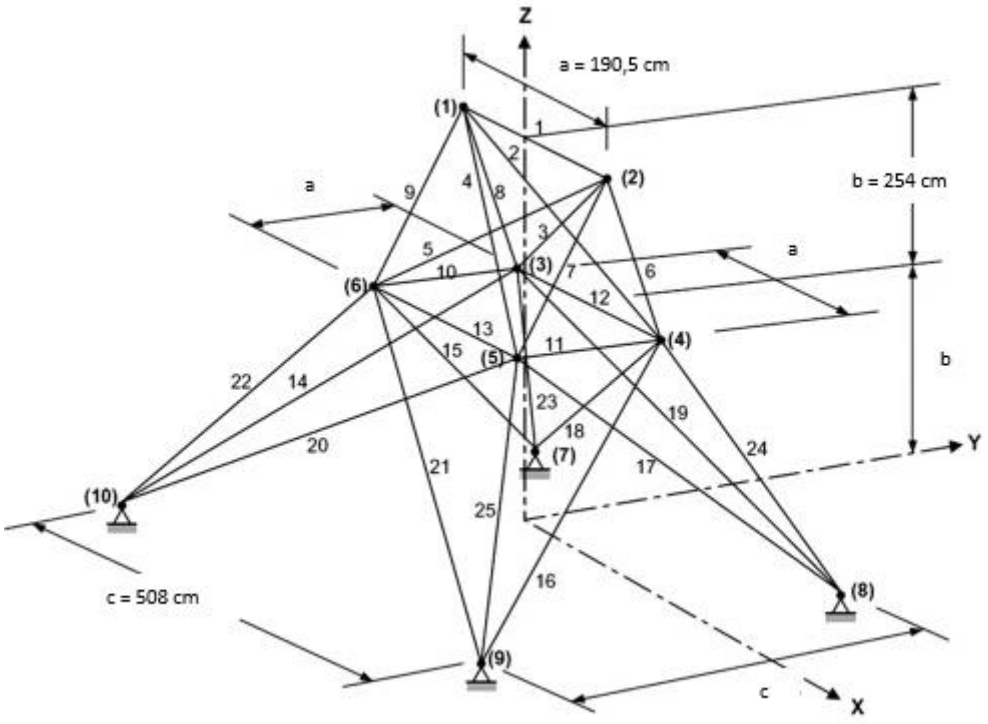


Figure 4.2. 25-bar spatial truss structure

Table 4.2. summarizes the optimal results of employed algorithms, and their convergence histories are represented in Figure 4.3. a) to c).

Table 4.2. shows that the HPSO algorithm has reported the best result 219,92 kg (484,85 lb) the category of previous studies while from algorithms referred in this present

work, PSO has brought out the best design 220,73 kg (486,63 lb). The parameters of the different algorithms are respectively, MaxIt=100 and nPop=25 for ABC, MaxIt=100, HMS=25 and nNew=20 for HS and finally, MaxIt=100 and nPop=25 for PSO.

Table 4.2. Optimal design for 25-bar spatial truss structure

Variables		Previous studies				This study		
(in^2)		PSOPC	HPSO	SGA	GA	ABC	HS	PSO
A_1		0.1	0.1	0.1	0.1	0.1	0.1	0.1
A_2		1.1	0.3	0.5	1.8	0.6	1.6	0.2
A_3		3.1	3.4	3.4	2.3	3.4	3.2	3.4
A_4		0.1	0.1	0.1	0.2	0.9	0.1	0.1
A_5		2.1	2.1	1.5	0.1	1.6	1.5	1.6
A_6		1.0	1.0	0.9	0.8	0.9	0.8	0.9
A_7		0.1	0.5	0.6	1.8	0.5	0.2	0.8
A_8		3.5	3.4	3.4	3.0	3.4	3.4	3.4
Weight	(lb)	490,16	484,85	486,29	546,01	497,77	498,94	486,63
	(kg)	222,333	219,924	220,577	247,666	225,78	226,32	220,73

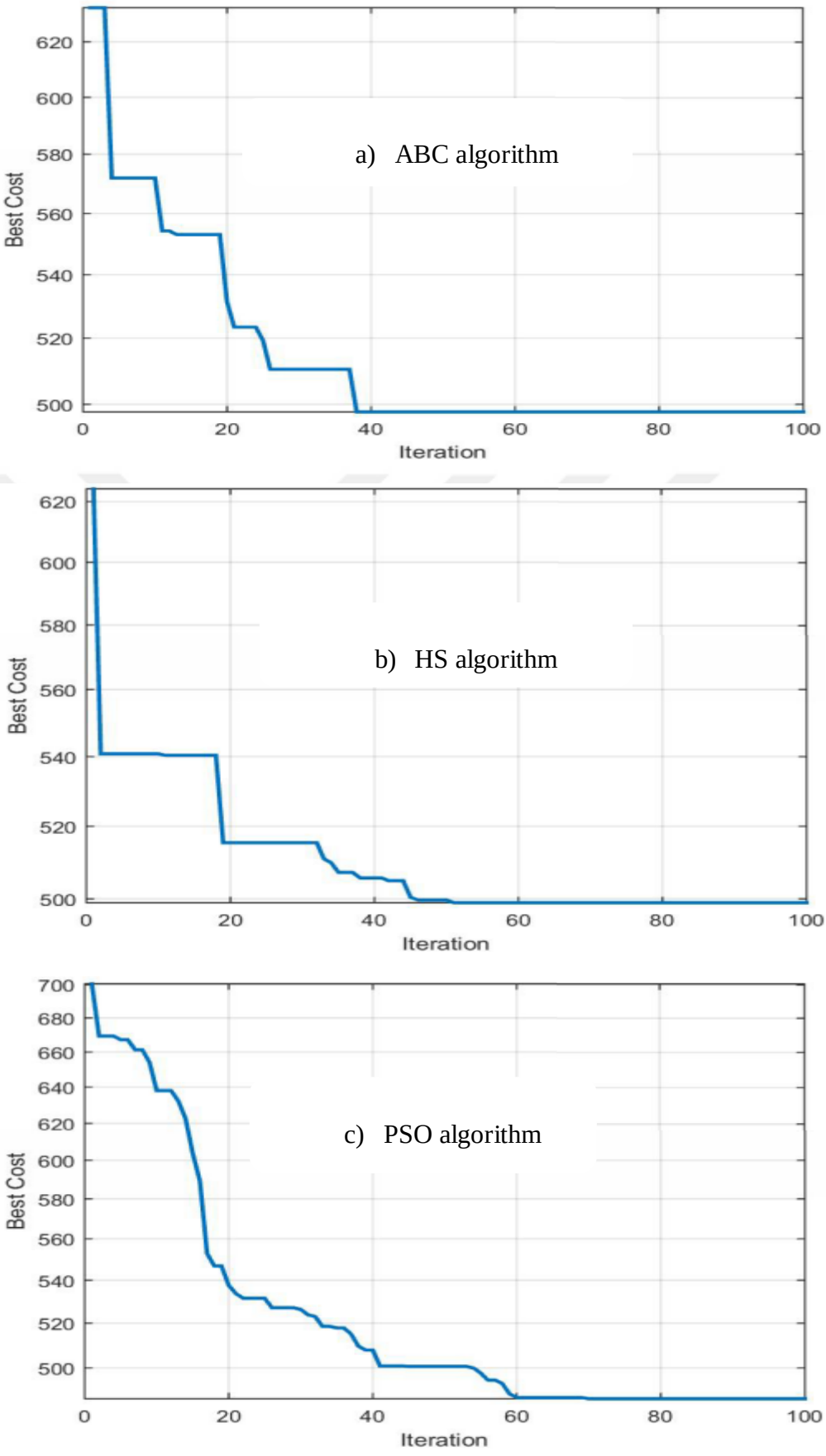


Figure 4.3. Convergence histories for 25-bar spatial truss structure

4.4.2. Seventy-two-bar (72) Spatial Truss Structure

The 72-bar four level skeletal tower has been studied by different researchers: WU and Chow (1995), Li et al. (2009) and Erbatur et al. (2000). The geometry definition of the structure is shown in Figure 4.4. For the design, the material density and the elasticity modulus are respectively equal to 2767,990 kg/m^3 (0.1 lb/in^3) and 68 950 MPa (10 000 ksi). The Constraints imposed to structure are of two kinds. Displacements in x and y directions at the uppermost joints are limited to ± 0,635 cm (0.25 in) and the elements stress should not exceed ± 172,375 MPa (25 ksi). The structural loading condition is given in Table 4.3. The 72 members of this spatial truss are symmetrically divided into 16 groups as follows: A 1 (1 – 4), A 2 (5 –12), A 3 (13 – 16), A 4 (17 – 18), A 5 (19 – 22), A 6 (20 – 30), A 7 (31 – 34), A 8 (35 – 36), A 9 (37 – 40), A 10 (41 – 48), A 11 (49 – 52), A 12 (53 – 54), A 13 (55 – 58), A 14 (59 – 62), A 15 (63 – 70), A 16 (71 – 72). The discrete variables are selected from D.

D = [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2.0, 2.1, 2.2, 2.3, 2.4, 2.5, 2.6, 2.7, 2.8, 2.9, 3.0, 3.1, 3.2] (in^2).

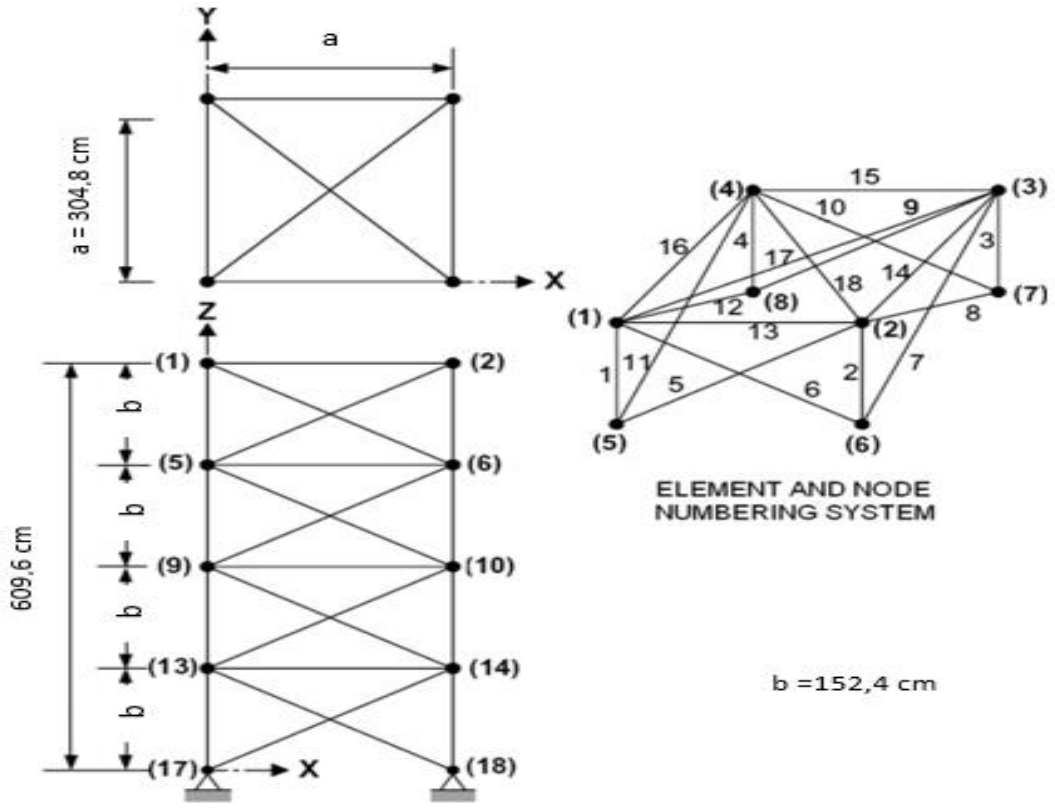


Figure 4.4. 72-bar spatial truss structure

Table 4.3. Load case for the 72-bar spatial truss

Nodes	Load case (kN)		
	Px	Py	Pz
17	22,24	22,24	(-)22,24
18	0.0	0.0	0.0
19	0.0	0.0	0.0
20	0.0	0.0	0.0

Table 4.5. compares optimization results of different algorithms, with their convergence histories represented in Figure 4.5. a) to c).

The lightest truss weight obtained in this thesis work by ABC that equals 172,64 kg (380,60 lb) which is slightly lighter than the best design value generated in referred literature that is 174,84 kg (385,54 lb). Here the following parameters were considered, MaxIt=200 and nPop=25 for ABC, MaxIt=200, HMS=25 and nNew=50 for HS and finally, MaxIt=200 and nPop=25 for PSO.

Table 4.4. Optimal design for 72-bar spatial truss structure

Variables		Previous studies				This study		
(in^2)		PSOPC	HPSO	GA	DHPSACO	ABC	HS	PSO
A_1		3.0	2.1	1.5	1.9	2.0	1.9	1.8
A_2		1.4	0.6	0.7	0.5	0.5	0.5	0.6
A_3		0.2	0.1	0.1	0.1	0.1	0.1	0.1
A_4		0.1	0.1	0.1	0.1	0.1	0.1	0.1
A_5		2.7	1.4	1.3	1.3	1.4	1.5	1.3
A_6		1.9	0.5	0.5	0.5	0.5	0.6	0.4
A_7		0.7	0.1	0.2	0.1	0.1	0.1	0.1
A_8		0.8	0.1	0.1	0.1	0.1	0.1	0.1
A_9		1.4	0.5	0.5	0.6	0.6	0.5	0.7
A_{10}		1.2	0.5	0.5	0.5	0.5	0.4	0.5
A_{11}		0.8	0.1	0.1	0.1	0.1	0.1	0.1
A_{12}		0.1	0.1	0.2	0.1	0.2	0.1	0.1
A_{13}		0.4	0.2	0.2	0.2	0.1	0.1	0.1
A_{14}		1.9	0.5	0.5	0.6	0.5	0.6	0.6
A_{15}		0.9	0.3	0.5	0.4	0.4	0.4	0.4
A_{16}		1.3	0.7	0.7	0.6	0.6	0.6	0.6
Weight	(lb)	1069,79	388,94	400,66	385,54	380,60	385,54	383,14
	(kg)	485,25	176,4	181,7	174,88	172,64	174,88	173,79

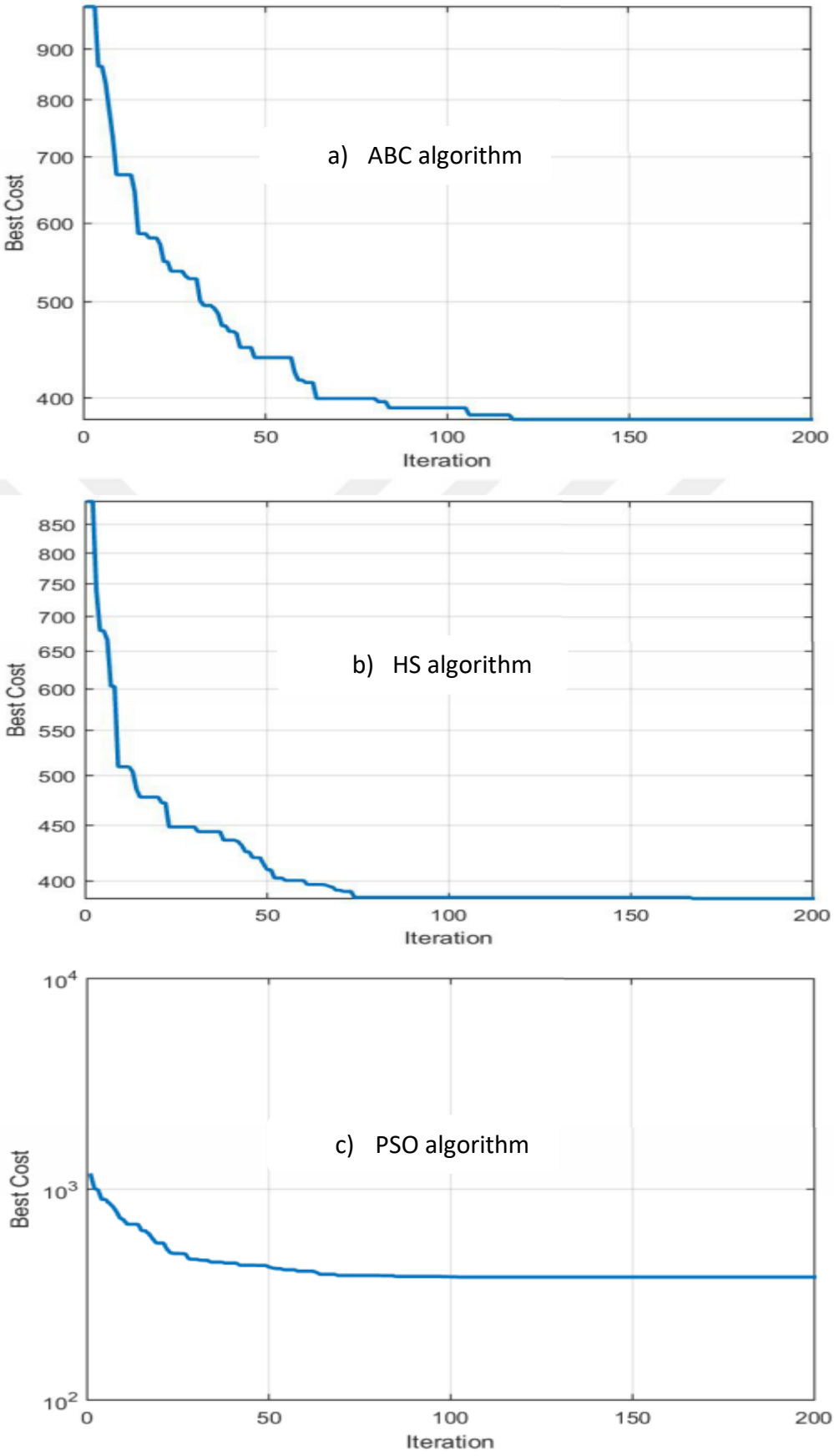


Figure 4.5. Convergence histories for 72-bar spatial truss structure

4.4.3. One Hundred Twenty-bar (120) Spatial Truss Structure

The last optimization example is conducted on a 120-bar dome truss shown in Figure 4.6., previously studied by Kaveh and Talatahari (2009). Based on structure's symmetry considerations, elements have been split into seven (7) main groups as it is noticeable on the figure. The material has an elasticity modulus of 209 952,75 MPa (30 450 ksi), a density of 7971,81 kg/m^3 (0,288 lb/in^3), and a yield stress of 399,91 MPa (58,0 ksi). All unsupported nodes are subjected to vertical forces that are equal to – 60 kN (13.49 kips) at node 1, − 29,98 (6.744 kips) at nodes 2 to 14, and −2.248 kips at the rest of the nodes. The optimization carried out is continuous and the minimum allowable cross section of members is limited to 0.775 in with a maximum of 20.0 in 2. Two cases are implemented with displacement constraints and stress constraints used according to the ASD code (AISC, 2011), as follows:

$$\begin{cases} \sigma_i^+ = 0.6F_y & \text{For } \sigma_i \geq 0 \\ \sigma_i^- & \text{For } \sigma_i \leq 0 \end{cases}$$

σ_i^- is evaluated according the formula below:

$$\begin{cases} \left[\left(1 - \frac{\lambda_i^2}{2C_c^2}\right)F_y\right] \Big/ \frac{5}{3} + \frac{3\lambda_i}{8C_c} + \frac{\lambda_i^3}{8C_c^3} & \text{For } \lambda_i < C_c \\ \frac{12\pi^2 E}{23\lambda_i^2} & \text{For } \lambda_i \geq C_c \end{cases}$$

where E represents the elasticity modulus, Fy the yield stress, λi the slenderness ratio (λi = kLi / ri), k the effective length factor, Li the length of the member, ri the radius of gyration (ri = 0.4993A$^{0.6777}$). The value of C_c is determined by $C_c = \sqrt{(2\pi^2 E/F_y)}$.

The two study cases invoked earlier above are described as follows:

Case 1: stress constraints imposed, without displacement constraints,

Case 2: stress constraints imposed, with displacement constraints at all nodes in x, y and z directions, limited to $\mp$ 0,50 cm (0.1969 in).

Table 4.6. compares optimization results of different algorithms for the first case with their convergence histories represented in Figure 4.7. a) to c), while Table 4.7. is affected to case 2, with their convergence histories represented in Figure 4.8. a) to c).

In this example the parameters considered are, MaxIt=200 and nPop=50 for ABC, MaxIt=200, HMS=50 and nNew=50 for HS and finally, MaxIt=200 and nPop=50 for PSO.

In case 1 the best design obtained from the previous studies is 8824,41 kg (19454.49 lb) which is a bit bigger than the best design value generated in this thesis work by PSO that equals 8820,54 kg (19445.97 lb).

In case 2 the best design obtained from the previous studies is 15083,74 kg (33253.95 lb) which is a bit bigger than the best design value generated in this thesis work by PSO that equals 14942,72 kg (32943.06 lb).

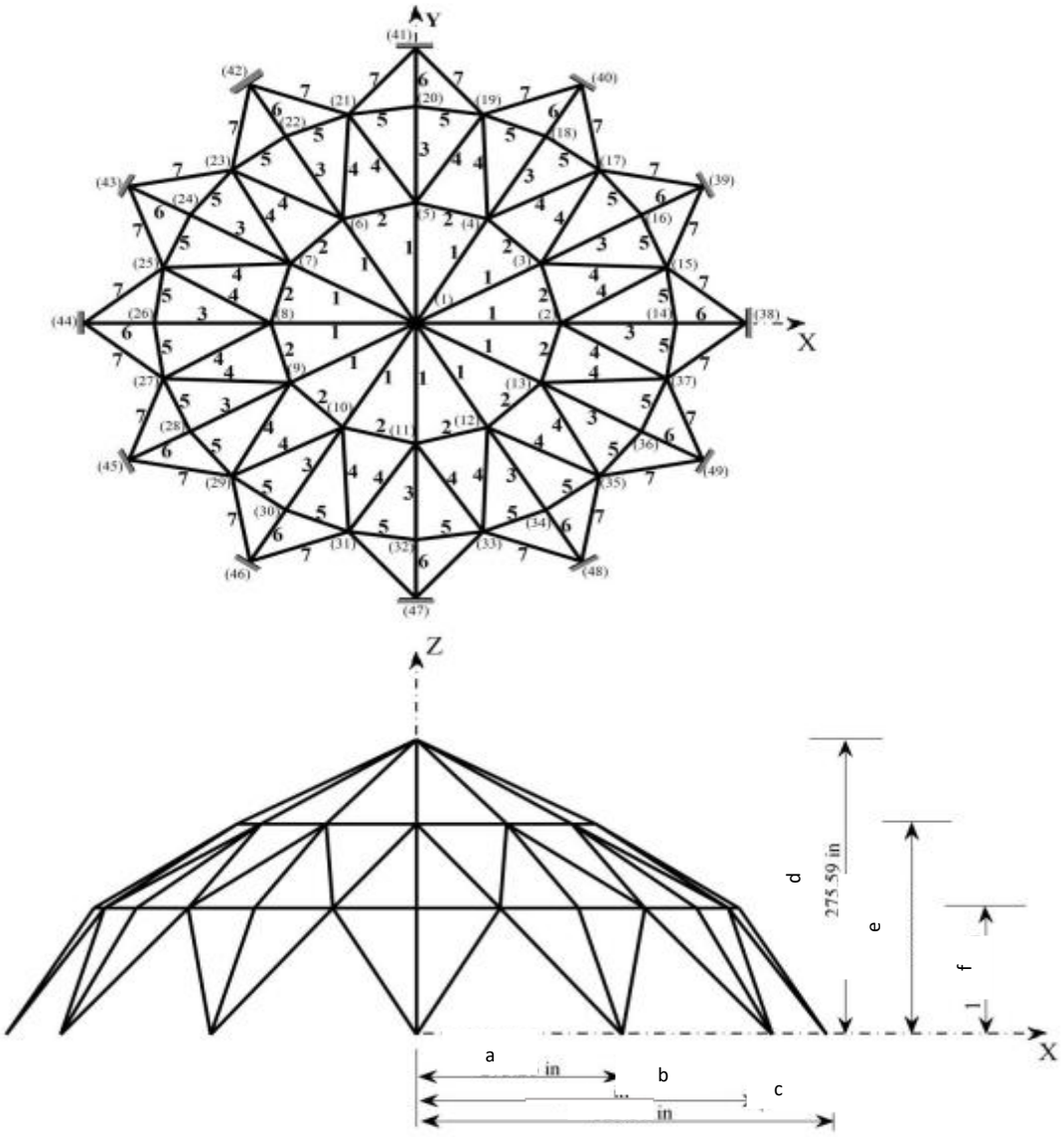


Figure 4.6. 120-bar spatial truss structure

a = 694,08 cm; b = 1249,98 cm; c = 1589cm; d = 700 cm; e = 500 cm; f = 300 cm.

Table 4.5. Optimal design for 120-bar spatial truss structure (case 1)

Variables		Previous studies				This study		
(in^2)		CA	PSOPC	RO	MABC	ABC	HS	PSO
A_1		3.122897	3.235	3.128	3.2976	3.15	3.15	3.15
A_2		3.353849	3.37	3.357	2.3964	3.36	3.35	3.34
A_3		4.111981	4.116	4.114	3.8736	4.13	4.11	4.11
A_4		2.782138	2.784	2.783	2.5710	2.78	2.77	2.77
A_5		0.775000	0.777	0.775	1.1513	0.795	0.775	0.775
A_6		3.300503	3.343	3.302	3.3323	3.31	3.31	3.30
A_7		2.445793	2.454	2.453	2.7848	2.45	2.45	2.44
	(lb)	19454,49	19618,70	19476,19	19706,62	19534,61	19458,21	19445,97
Weight	(kg)	8824,41	8898,89	8834,25	8938,77	8860,75	8826,10	8820,54

Table 4.6. Optimal design for 120-bar spatial truss structure (case 2)

Variables		Previous studies				This study		
(in^2)		CA	PSOPC	RO	ICA	ABC	HS	PSO
A_1		3.02591	3.040	3.030	3.02750	3.07	3.04	3.04
A_2		14.7652	13.149	14.806	14.45960	14.20	15.00	14.10
A_3		5.08463	5.646	5.440	5.24460	5.05	4.56	5.44
A_4		3.13569	3.143	3.124	3.14130	3.20	2.78	2.72
A_5		8.43852	8.759	8.021	8.45410	8.67	8.84	8.79
A_6		3.35678	3.758	3.614	3.35670	3.45	3.93	3.57
A_7		2.49627	2.502	2.487	2.49447	2.53	2.53	2.50
	(lb)	33253,95	33481,20	33317,80	33256,20	33407,51	33101,71	32943,06
Weight	(kg)	15083,74	15186,82	15112,70	15084,76	15153,39	15014,68	14942,72

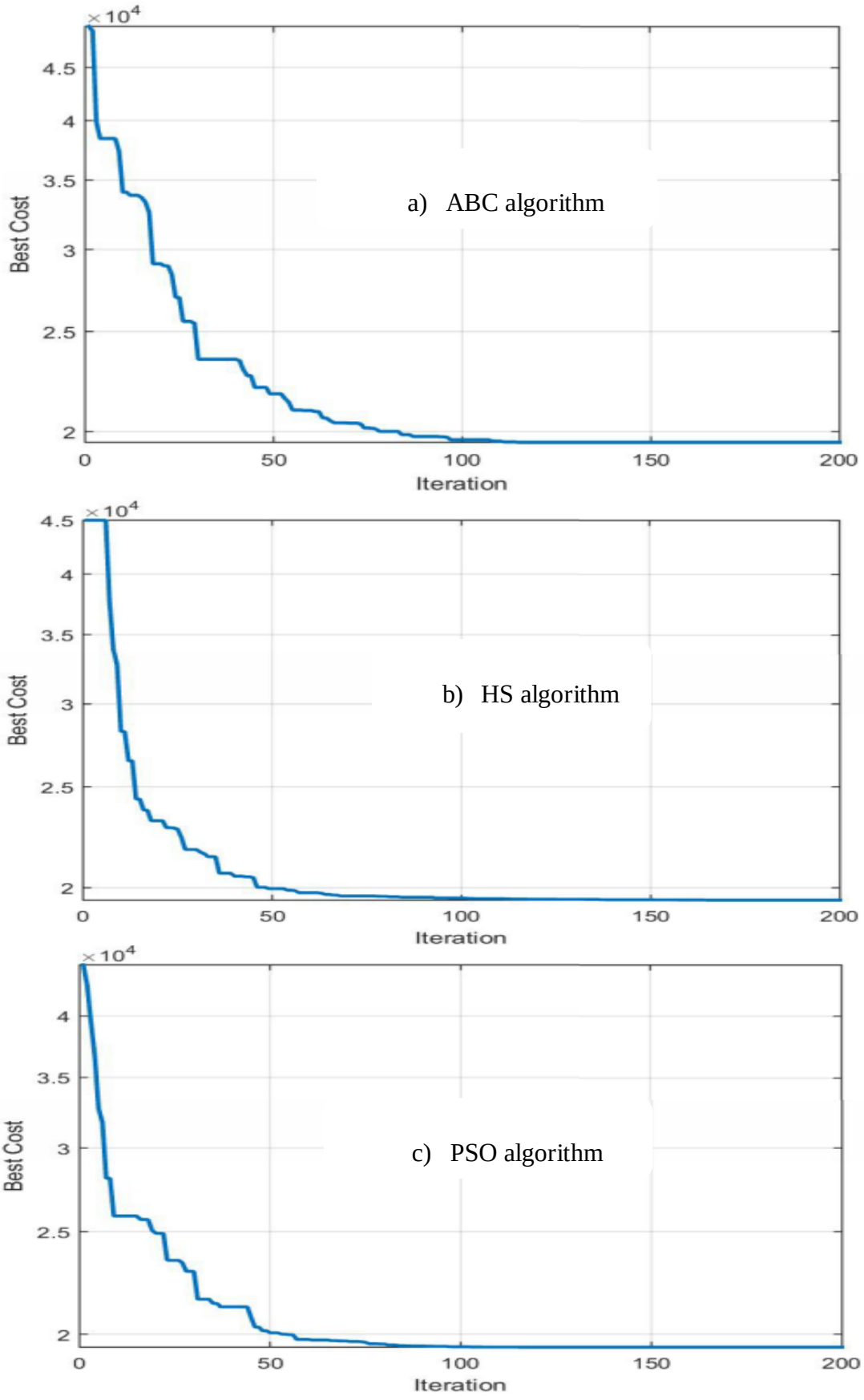


Figure 4.7. Convergence histories for 120-bar spatial truss structure (case 1)

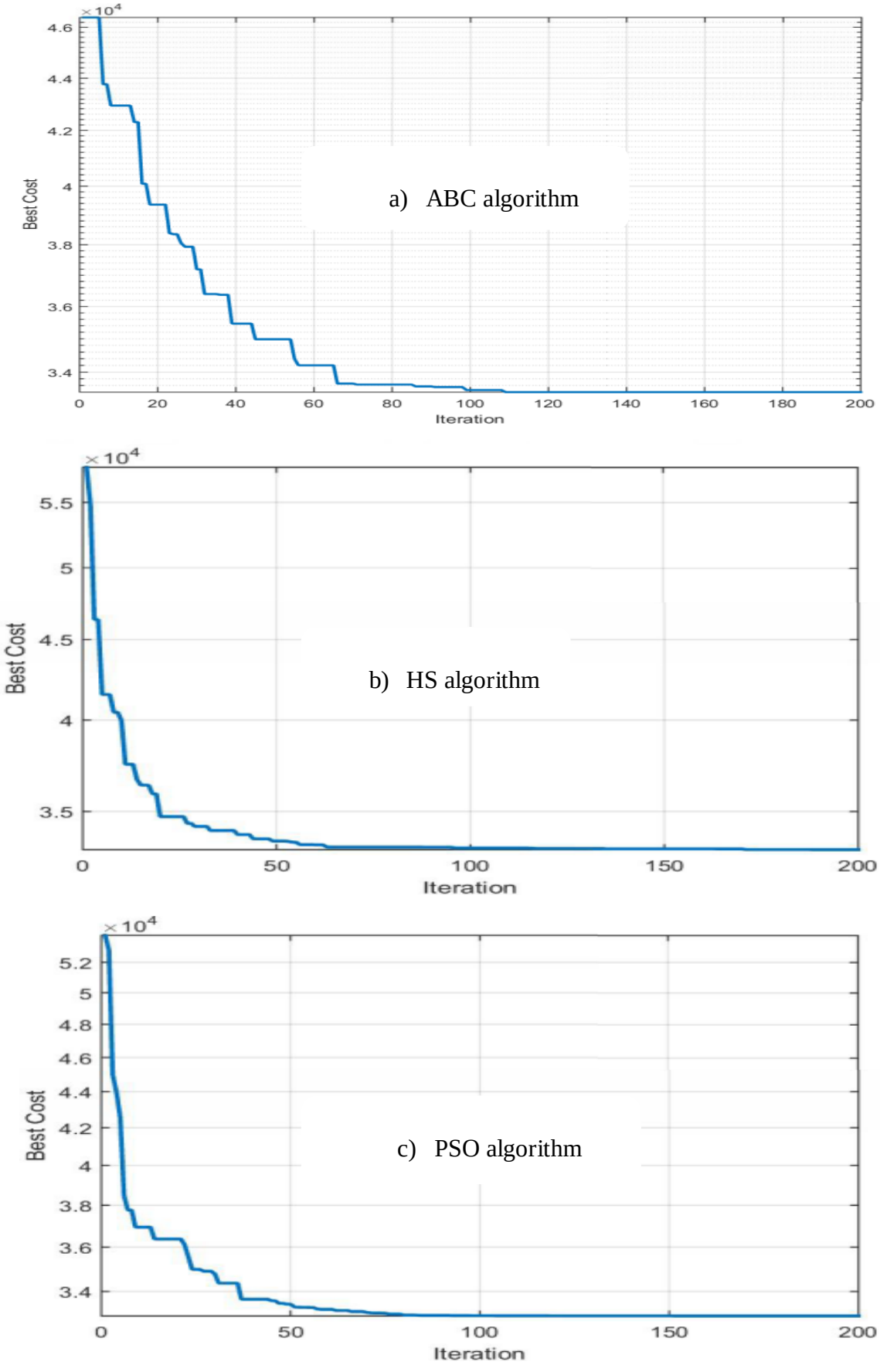


Figure 4.8. Convergence histories for 120-bar spatial truss structure (case 2)

5. CONCLUSION

This last chapter presents and makes a comparison of obtained results to draw some Conclusions. Moreover, recommendations will be suggested for future studies.

5.1. Conclusion

In this thesis study, three (3) optimization algorithms have been employed to evaluate the optimal weight of steel trusses. These algorithms Artificial Bee Colony (ABC), Harmony Search (HS) and Particle Swarm Optimization (PSO), were tested on discrete as well as continuous problems. In total, nine (9) design problems have been used to evaluate the performance of the selected algorithms. The implementation of structural weight optimization, with imposed constraints on member stresses and/or nodal displacements was principally relying on determining the minimal cross-sectional areas of truss bars that are considered as design variables. Problems were solved using two different programming ways. Planar truss structures were solved entirely in MATLAB by combining structural analysis scripts based on FEM and the optimization algorithms. On the other hand, due to the fact that generally structural configuration may be very complex and they implementation trough MATLAB codes require a considerable work time, the integration of SAP2000 structural analysis program has been preferred for solving spatial truss problems.

As indicated above, the examples were studied using either continuous or discrete design variables. Considering the first study case where both continuous and discrete optimizations were performed for the same structure with the same load case and constraints limitations, the most effective solutions have been produced by continuous variables. Despite its easy implementation, mathematically feasible solutions, and the non-negligible gap with discrete optimization results, continuous optimization is not preferred as the cross sections generated do not correspond to available commercial member sizes. For research purposes, optimization with continuous variables can be performed, but for real problems, discrete design variable although harder to set, should be preferred because of its practical and reliable solutions. Moreover, the optimization scheme is considered to be more robust.

Based on the three (3) optimization algorithms used in this present work, some important conclusions can be retrieved. The comparisons of this study's numerical results with those of previous studies are made not only to demonstrate and verify the efficiency of the work done, but also to show the robustness of each algorithm. The algorithm's parameters such as population size and the maximum iteration number were set after many trial studies, observing the convergence history. In all the considered examples PSO generally offers the best performance compared to the two others, in both convergence and optimal result. PSO revealed to bring out good results with the least population number and reaches the optimal result in less structural analyses. After PSO, ABC was the second-best algorithm, generating in some examples the same result with PSO or a better result than PSO with small difference. However, a bigger population size must be set to produces favourable results. Regarding now the HS, although the proposed method has yielded to optimal or got closer to optimal result, its performance has shown to be less efficient than ABC and PSO in this study. It must be underlined that there is not a metaheuristic algorithm elected as the best one in all cases of numerical design problems, because they offer no guaranty for the quality of the final result. A given algorithm may work well for an example and give a bad result in another example depending on the parameters assigned to the study.

5.2. Recommendations

The study carried out here, has revealed its part of success but many other considerations could be done in further research, by integrating promising interest areas in the field of design optimisation. They are summarized in form of recommendations as follows:

- Structural optimisation using linear analysis may result in a bad design because of the non-linear behaviour of some structures. Non-linear analysis should be considered regarding the geometry of the structure and assumptions made on geometric imperfections.
- In the present work only size optimization was developed, but it could also be extended to shape and topology optimization. The problem's fitness function will then have the character of multi-objective function.
- At last, more complex structural problems such as plane and space frames could be optimized to deal with bending moments, story drifts and dynamic loads.

6. REFERENCES

Alatas, B., 2010. Chaotic Harmon Search Algorithms, Applied Mathematics and Computation, 216, 2687-2699.

American Institute of Steel Construction, 2011, AISC, Steel Construction Manual, 14th ed., AISC, United States of America.

Argyris, J.H., 1965. Matrix Displacement Analysis of Anisotropic Shells by Triangular Elements, Journal of the Royal Aeronautical Society, 69, 801–805.

Bathe, K.J., 1982, Finite Element Procedures in Engineering Analysis, Prentice-Hall, Englewood Cliffs, NJ.

Berke, L. and Khot, N.S., 1974. Use of Optimality Criteria Approach to Structural Optimization for Large Scale Systems, Structural Optimization, AGARD LS, 70, 1-29.

Berke, L. and Khot, N.S., 1987. Structural Optimization Using Optimality Criteria, NATO, AS1 Series, Springer-Verlag Berlin, Heidelberg, vol. F27, 271-311.

Boussaid, I., 2013, Perfectionnement de Métaheuristiques pour l'Optimisation Continue, Thèse de Doctorat, Université Paris-Est Créteil, Paris.

Bruyneel, M., Craveur, J.-C. and Gourmelen, P., 2014, Optimisation des Structures Mécaniques – Méthodes Numériques et Eléments Finis, Dunod, Paris.

Camp, C., Pezeshk, S. and Cao, G., 1998. Optimized Design of Two-Dimensional Structures Using a Genetic Algorithm, Journal of Structural Engineering,124, 551–559.

Camp, C.V. and Farshchin, M., 2014. Design of Space Trusses Using Modified Teaching–Learning Based Optimization, Engineering Structures, 62–63, 87–97.

Carlson, S.E., 1995. A General Method for Handling Constraints in Genetic Algorithms. In Proceedings of the Second Annual Joint Conference on Information Science, 663-667.

Carroll, C.W., 1961. The Created Response Surface Technique for Optimizing Nonlinear Restrained Systems, Operations Research, 9, 169-184.

Cheng, Y.M., Li, L. and Chi, S.C., 2007. Performance Studies on Six Heuristic Global Optimization Methods in the Location of Critical Slip Surface, Computers and Geotechnics, 34, 462-484.

Clough, R.W., 1960. The finite Element Method in Plane Stress Analysis. Proceedings of the 2nd A.S.C.E. Conference in Electronic Computation, Pittsburgh, PA.

Coello Coello, C.A., 1999. A Survey of Constraint Handling Techniques Used with Evolutionary Algorithms, Laboratorio Nacional de Informática Avanzada.

Coello Coello, C.A, 2000. Use of a Self-Adaptive Penalty Approach for Engineering Optimization Problems, Computers in Industry, 41, 113-127.

Connor, J.J., 2003, Introduction to Structural Motion Control, Prentice Hall Pearson Education, Inc., Michigan University.

Courant, R., 1943. Variational Methods for the Solution of Problems of Equilibrium and Vibrations, Bulletin of the American Mathematical Society, 49, 1-23.

Erbatur, F. and Al-Hussainy, M.M., 1992. Optimum Design of Frames, Computers and Structures, 45, 887–891.

Erbatur F., Hasancebi O., Tutuncu I., and Kilic H., 2000. Optimal Design of Planar and Space Structures with Genetic Algorithms, Computers and Structures, 75, 209–232.

Eskkandar H., Sadollah A., Bahreininejad A. and Hamdi M. 2012. Water Cycle Algorithm-A Novel Metaheuristic Optimization Method for Solving Constrained Engineering Optimization Problems, Computers and Structures, 110-111, 151-166.

Geem, Z.W., 2007, Harmony Search Algorithm for Solving Sudoku, in Proceedings of the 11th International Conference, KES 2007 and XVII Italian Workshop on Neural Networks Conference on Knowledge-Based Intelligent Information and Engineering Systems: Part I. Springer-Verlag, Vietrisul Mare, Italy.

Glover, F., 1986. Future Paths for Integer Programming and Links to Artificial Intelligence, Computers and Operation Research, 13, 533–549.

Hackwood, S. and Wang, J., 1988. The Engineering of Cellular Robotic Systems, Proceedings of Institute of Electrical and Electronics Engineers International Symposium, Arlington, Virginia, USA.

Hadidi, A. and Kazemzadeh, S., 2010. Structural Optimization Using Artificial Bee Colony Algorithm, 2nd international conference on engineering optimization.

Hadidi, A., Kazemzadeh, A.S. and Kazemzadeh, A.S., 2010. Structural Optimization Using Artificial Bee Colony Algorithm, 2nd International Conference on Engineering Optimization, Lisbon, Portugal.

Holland, J.H., 1975. Adaptation in Natural and Artificial System, University of Michigan Press.

Joines, J.A. and Houck, C.R., 1994. On the Use of Non-Stationary Penalty Functions to Solve Nonlinear Constrained Optimization Problems with GA's. In Z. Michalewicz, J. D. Schaffer, H.-P. Schwefel, D. B. Fogel & H. Kitano (Eds.), Proceedings of the First IEEE International Conference on Evolutionary Computation (ICEC'94), Orlando, FL, USA, 579-584.

Karaboga, D., 2005. An Idea Based on Honey Bee Swarm for Numerical Optimization, Technical Report TR06, Erciyes University, Engineering Faculty, Computer Engineering Department.

Karaboga, D. and Basturk, B., 2007. A Powerful and Efficient Algorithm for Numerical Function Optimization: Artificial Bee Colony (ABC) Algorithm, Journal of Global Optimization, 39, 459-471.

Karaboga, D. and Basturk, B., 2008. Artificial Bee Colony (ABC) Optimization Algorithm for Solving Constrained Optimization Problems, Adv Soft Comput 4529, 687–697.

Karaboga, D. and Akay, B., 2009. A comparative Study of Artificial Bee Colony Algorithm, Applied Mathematics and Computation, 214, 108-132.

Kaveh A. and Talatahari S., 2009. Size Optimization of Space Trusses Using Big Bang-Big Crunch Algorithm, Computers and Structures, 87, 1129-1140.

Kennedy, J. and Eberhart, R.C., 1997. A Discrete Binary Version of the Particle Swarm Algorithm, Proceedings of the World Multi-Conference on Systemic, Cybernetics and Informatics, NJ, USA.

Khot, N.S. and Berke, L., 1984. Structural Optimization Using Optimality Criteria Methods, New Directions in Optimum Structural Design, E. Atrek, R.H. Gallagher, K.M. Ragsdell, O.C. Zienkiewicz, (Eds), John Wiley, New York.

Kicinger, R., Arciszewki, T. and Jong, K.D., 2005. Evolutionary computation and Structural Design: A Survey of The State-of-the-art, Computers and Structures, 83,1943–1978.

Kirkpatrick, S., Gelatt, C., and Vecchi, M., 1983. Optimization by Simulated Annealing, Science, Vol. 220, 4598, 671-680.

Kirsch, U., 1982. Optimal Design Based on Approximate Scaling, J. Struct. Div., ASCE 108, 888-910.

Koohestani, K. and Kazemzadeh, A.S., 2009. An Adaptive Real-Coded Genetic Algorithm for Size and Shape Optimization of Truss Structures, The First International Conference on Soft Computing Technology in Civil, Structural and Environmental Engineering, Civil-Comp Press, Stirlingshire, UK.

Lee, K.S. and Geem, Z.W., 2004. A New Structural Optimization Method Based on The Harmony Search Algorithm, Computers and Structures, 82, 781-798.

Lee, K.S. and Geem, Z., 2005. A New Meta-Heuristic Algorithm for Continuous Engineering Optimization: Harmony Search Theory and Practice, Computer Methods in Applied Mechanics and Engineering, 194, 36-38, 3902-3933.

Lee, K.S., Geem, Z.W., Lee, S.-H. and Bae, K.W., 2005. The Harmony Search Heuristic Algorithm for Discrete Structural Optimization, Engineering Optimization, 37, 663-684.

Lemmens, N., Jong, S., Tuyls, K. and Nowe, A., 2007. A Bee Algorithm for Multi-Agent Systems: Recruitment and Navigation Combined, Proceeding of ALAG 2007, an AAMAS'07 workshop, Honolulu, Hawaii.

Leu, L. J. and Huang, C.W., 2000. Optimized Design of Two Dimensional Structures Using a Genetic Algorithm, J. Struct. Engrg., 5, 551-559.

Li, L.J., Huang, Z.B. and Liu, F., 2009. A Heuristic Particle Swarm Optimization Method for Truss Structures with Discrete Variables, Computers and Structures, 87, 435-443.

Li, L.J., Huang, Z.B., Liu, F. and Wu, Q.H., 2007. A Heuristic Particle Swarm Optimizer for Optimization of Pin Connected Structures, Computers and Structures, 85 (7-8), 340-349.

Medina, A.J.R., Pulido, G.T. and Torres, J.G.R., 2009. A Comparative Study of Neighbourhood Topologies for Particle Swarm Optimizers, Proceedings of the International Joint Conference on Computational Intelligence, Funchal, Madeira, Portugal.

Michalewicz, Z., 1994, 1994. Genetic Algorithms + Data Structures = Evolution Programs, 2nd, Extended ed., Springer-Verlag New York, Inc., New York.

Michalewicz, Z. and Attia, N.F., 1994. Evolutionary Optimization of Constrained Problems. In A. V. Sebald & L. J. Fogel (Eds.), Proceedings of the Third Annual Conference on Evolutionary Programming, San Diego, CA, USA, 98-108.

Nanakorn, P. and Meesomklin, K., 2001. An Adaptive Penalty Function in Genetic Algorithms for Structural Design Optimization, Computers and Structures, 79, 2527-2539.

Patnaik, S.N., Guptill, J.D. and Berke, L., 1993. Merits and Limitations of Optimality Criteria Method for Structural Optimization, NASA Technical Paper 3373.

Pedersen, M.E., 2010. Good Parameters for Particle Swarm Optimization, Luxembourg: Hvass Laboratories.

Pham, D.T., Ghanbarzadeh, A., Koc, E., Otri, S., Rahim, S. and Zaidi, M. 2006. The Bees Algorithm, a Novel Tool for Complex Optimisation Problems. Proceedings of the 2nd International Virtual Conference on Intelligent Production Machines and Systems (IPROMS 2006), Oxford, Elsevier, 454-459.

Rajeev, S. and Krishnamoorthy, C., 1992.Discrete Optimization of Structures Using Genetic Algorithms, Journal of Structural Engineering, 118, 1233-1250.

Ringertz, U.T., 1988. On Methods for Discrete Structural Constraints, Engineering Optimization, 13, 47-64.

Saka, M.P., 2007. Optimum Design of Steel Frames using Stochastic Search Techniques Based on Natural Phenomena, A Review Civil Engineering Computations: Tools and Techniques Saxe-Coburg Publications.

Schmit, L.A., 1960. Structural Design by Systematic Synthesis, Proceedings of the Second Conference on Electronic Computation, ASCE, New York.

Schwefel, H.-P., 1981, Numerical Optimization of Computer Models, John Wiley & Sons, Chichester, UK.

Sextos, A.G. and Balafas, G.K., 2011. Using the New Sap2000 Open Application Programming Interface to Develop an Interactive Front-End for the Modal Pushover Analysis of Bridges, 3rd ECCOMAS Thematic Conference on Computational Methods in Structural Dynamics and Earthquake Engineering, Corfu, Greece.

Shi, Y. and Eberhart, R., 1998. A Modified Particle Swarm Optimizer. Proceedings of the International Conference on Evolutionary Computation, 69–73.

Storn, R. and Price, K., 1997. Differential Evolution – A Simple and Efficient Heuristic for Global Optimization Over Continuous Spaces, Journal of Global Optimization, 11, 341-359.

Tirro, F., 1977, Jazz: A History, W. W. Norton and Company.

Tunchan, C., 2009. Particle Swarm Optimization Approach to Portfolio Optimization, Nonlinear Analysis Real World, 10, 2396-2406.

Venkayya, V. B., Khot, N.S., and Berke, L., 1973. Application of Optimality Criteria Approaches to Automated Design of Large Practical Structures, Proceedings of the 2nd Symposium on Structural Optimization, AGARD-CP-123, 3-1–3-19.

Venkaya, V.B., Knot, N.S. and Reddy, V.S., 1968. Optimization of Structures Based on the Study of Strain Energy Distribution, Proceedings of the Second Conference on Matrix Methods in Structural Mechanics, WPAFB, AFFDL-TR-68-150.

Wang, X., Gao, X.Z. and Ovaska, S.J., 2009. Fusion of Clonal Selection Algorithm and Harmony Search Method in Optimization of Fuzzy Classification Systems, International Journal of Bio-Inspired Computation, 1, 1–2, 80–88.

Wu C. Y. and Chow P.-T., 1995. Steady State Genetic Algorithms for Discrete Optimization of Trusses, Computers and Structures, 56, 979-991.

Yang, X.-S., 2009. Firefly Algorithms for Multimodal Optimization. Proceedings of the 5th International Conference on Stochastic Algorithms: Foundations and Applications, Springer-Verlag, Sapporo, Japan.

Yang X.-S. and Deb S., 2009, Cuckoo Search via Levy Flights. Nature and Biologically Inspired Computing, 2009. NaBIC 2009, World Congress on, 210-214.

Yeniay, O., 2005. Penalty Function Methods for Constrained Optimization with Genetic Algorithms, Math. Comput. Applic., 10, 45-56.

Honey Bee Biology, http://honeybee.tamu.edu. Texas A&M University. Accessed, 2 June 2010.

Ministry of Agriculture and Lands of British Colombia, Apiary Fact Sheets. http://www.al.gov.bc.ca., Accessed 2 June 2010.

Mathworks MATLAB, http://www.Mathworks.com.

Yarpiz Metaheuristics. http://www.yarpiz.com/category/metaheuristics.

Csi Computers and Structures Inc. http://www.csiamerica.com/products/spa2000.

Wikipedia research tool. http://www.wikipedia.fr.

CURRICULUM VITAE

My name is Patrick Jean de Dieu Ouedraogo, I was born in Port-Bouët in1991 and I am from Burkina Faso. I have completed my previous degrees in Ouagadougou my home town, in Ecole Supérieure Polytechnique de la Jeunesse, where I got my bachelor in 2014, then I came to Karadeniz Technical University in Trabzon to pursue my master degree as a Turkish government scholarship holder. I studied Turkish language during one year before I start civil engineering courses. I also spend one semester in Ruhr University Bochum, in Germany with the Erasmus exchange program. Beside English and Turkish, I also speak French and Moore, my mother tongue. Now I am willing to publish some articles in collaboration with my thesis supervisor.