A. Kaveh

Advances in Metaheuristic Algorithms for Optimal Design of Structures

Springer

Advances in Metaheuristic Algorithms for Optimal Design of Structures

A. Kaveh

Advances in Metaheuristic Algorithms for Optimal Design of Structures

Springer

A. Kaveh
School of Civil Engineering, Centre of Excellence
for Fundamental Studies in Structural Engineering
Iran University of Science and Technology
Tehran, Iran

ISBN 978-3-319-05548-0 ISBN 978-3-319-05549-7 (eBook)
DOI 10.1007/978-3-319-05549-7
Springer Cham Heidelberg New York Dordrecht London

Library of Congress Control Number: 2014937527

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

Recent advances in structural technology require greater accuracy, efficiency, and speed in design of structural systems. It is therefore not surprising that new methods have been developed for optimal design of real-life structures and models with complex configurations and a large number of elements.

This book can be considered as an application of metaheuristic algorithms to optimal design of skeletal structures. The present book is addressed to those scientists and engineers, and their students, who wish to explore the potential of newly developed metaheuristics. The concepts presented in this book are not only applicable to skeletal structures and finite element models but can equally be used for design of other systems such as hydraulic and electrical networks.

The author and his graduate students have been involved in various developments and applications of different metaheuristic algorithms to structural optimization in the last two decades. This book contains part of this research suitable for various aspects of optimization for skeletal structures.

This book is likely to be of interest to civil, mechanical, and electrical engineers who use optimization methods for design, as well as to those students and researchers in structural optimization who will find it to be necessary professional reading.

In Chap. 1, a short introduction is provided for the development of optimization and different metaheuristic algorithms. Chapter 2 contains one of the most popular metaheuristic known as the Particle Swarm Optimization (PSO). Chapter 3 provides an efficient metaheuristic algorithm known as Charged System Search (CSS). This algorithm has found many applications in different fields of civil engineering. In Chap. 4, Magnetic Charged System Search (MCSS) is presented. This algorithm can be considered as an improvement to CSS, where the physical scenario of electrical and magnetic forces is completed. Chapter 5 contains a generalized metaheuristic so-called Field of Forces Optimization (FFO) approach and its applications. Chapter 6 presents the recently developed algorithm known as Dolphin Echolocation Optimization (DEO) mimicking the behavior of dolphins. Chapter 7 contains a powerful parameter independent algorithm, called Colliding Bodies Optimization (CBO). This algorithm is based on one-dimensional collisions

between bodies, with each agent solution being considered as the massed object or body. After a collision of two moving bodies having specified masses and velocities, these bodies are separated with new velocities. This collision causes the agents to move toward better positions in the search space. In Chap. 8, Ray Optimization Algorithm (ROA) is presented in which agents of the optimization are considered as rays of light. Based on the Snell's light refraction law when light travels from a lighter medium to a darker medium, it refracts and its direction changes. This behavior helps the agents to explore the search space in early stages of the optimization process and to make them converge in the final stages. In Chap. 9, the well-known Big Bang-Big Crunch (BB-BC) algorithm is improved (MBB-BC) and applied to structural optimization. Chapter 10 contains application of Cuckoo Search Optimization (CSO) in optimal design of skeletal structures. In Chap. 11, Imperialist Competitive Algorithm (ICA) and its application are discussed. Chaos theory has found many applications in engineering and optimal design. Chapter 12 presents Chaos Embedded Metaheuristic (CEM) Algorithms. Finally, Chap. 13 can be considered as a brief introduction to multi-objective optimization. In this chapter a multi-objective optimization algorithm is presented and applied to optimal design of large-scale skeletal structures.

I would like to take this opportunity to acknowledge a deep sense of gratitude to a number of colleagues and friends who in different ways have helped in the preparation of this book. Professor F. Ziegler encouraged and supported me to write this book. My special thanks are due to Mrs. Silvia Schilgerius, the senior editor of the Applied Sciences of Springer, for her constructive comments, editing, and unfailing kindness in the course of the preparation of this book. My sincere appreciation is extended to our Springer colleagues Ms. Beate Siek and Ms. Sashivadhana Shivakumar.

I would like to thank my former and present Ph.D. and M.Sc. students, Dr. S. Talatahari, Dr. K. Laknejadi, Mr. V.R. Mahdavi, Mr. A. Zolghadr, Mrs. N. Farhoudi, Mr. S. Massoudi, Mr. M. Khayatazad, Mr. M. Ilchi, Mr. R. Sheikholeslami, Mr. T. Bakhshpouri, and Mr. M. Kalate Ahani, for using our joint papers and for their help in various stages of writing this book. I would like to thank the publishers who permitted some of our papers to be utilized in the preparation of this book, consisting of Springer Verlag, Elsevier and Wiley.

My warmest gratitude is due to my family and in particular my wife, Mrs. L. Kaveh, for her continued support in the course of preparing this book.

Every effort has been made to render the book error free. However, the author would appreciate any remaining errors being brought to his attention through his email address: alikaveh@iust.ac.ir.

Tehran, Iran A. Kaveh
February 2014

Contents

Chapter 1
Introduction

1.1 Metaheuristic Algorithms for Optimization

In today's extremely competitive world, human beings attempt to exploit the maximum output or profit from a limited amount of available resources. In engineering design, for example, choosing design variables that fulfill all design requirements and have the lowest possible cost is concerned, i.e. the main objective is to comply with basic standards but also to achieve good economic results. Optimization offers a technique for solving this type of problems.

The term "optimization" refers to the study of problems in which one seeks to minimize or maximize a function by systematically choosing the values of variables from/within a permissible set. In one hand, a vast amount of research has been conducted in this area of knowledge, hoping to develop effective and efficient optimization algorithms. On the other hand, the application of the existing algorithms to real projects has been the focus of many studies.

In the past, the most commonly used optimization techniques were gradient-based algorithms which utilized gradient information to search the solution space near an initial starting point [1, 2]. In general, gradient-based methods converge faster and can obtain solutions with higher accuracy compared to stochastic approaches. However, the acquisition of gradient information can be either costly or even impossible to obtain the minima. Moreover, this kind of algorithms is only guaranteed to converge to local optima. Furthermore, a good starting point is quite vital for a successful execution of these methods. In many optimization problems, prohibited zones, side limits and non-smooth or non-convex functions should be taken into consideration. As a result, these non-convex optimization problems cannot easily be solved by these methods.

On the other hand other types of optimization methods, known as metaheuristic algorithms, are not restricted in the aforementioned manner. These methods are suitable for global search due to their capability of exploring and finding promising regions in the search space at an affordable computational time. Metaheuristic algorithms tend to perform well for most of the optimization problems [3, 4].

A. Kaveh, *Advances in Metaheuristic Algorithms for Optimal Design of Structures*,
DOI 10.1007/978-3-319-05549-7_1, © Springer International Publishing Switzerland 2014

This is because these methods refrain from simplifying or making assumptions about the original problem. Evidence of this is their successful applications to a vast variety of fields, such as engineering, physics, chemistry, art, economics, marketing, genetics, operations research, robotics, social sciences, and politics.

The word *heuristic* has its origin in the old Greek work *heuriskein*, which means the art of discovering new strategies (rules) to solve problems. The suffix *meta*, also is a Greek word, means "upper level methodology". The term *metaheuristic* was introduced by F. Glover in the paper [5].

A heuristic method can be considered as a procedure that is likely to discover a very good feasible solution, but not necessarily an optimal solution, for a considered specific problem. No guarantee can be provided about the quality of the solution obtained, but a well-designed heuristic method usually can provide a solution that is at least nearly optimal. The procedure also should be sufficiently efficient to deal with very large problems. The heuristic methods are often considered as *iterative algorithm*, where each iteration involves conducting a search for a new solution that might be better than the best solution found previously. After a reasonable time when the algorithm is terminated, the solution it provides is the best one that was found during any iteration. A metaheuristic is formally defined as an iterative generation process which guides a subordinate heuristic by combining intelligently different concepts for exploring (global search) and exploiting (local search) the search space, learning strategies are used to structure information in order to find efficiently near-optimal solutions [5–7].

Metaheuristic algorithm has found many applications in different areas of applied mathematics, engineering, medicine, economics and other sciences. These methods are extensively utilized in the design of different systems in civil, mechanical, electrical, and industrial engineering. At the same time, one of the most important trends in optimization is the constantly increasing emphasis on the interdisciplinary nature of the field.

1.2 Optimal Design of Structures and Goals of the Present Book

In the area of structural engineering that is the main concern of this book, one tries to achieve certain objectives in order to optimize weight, construction cost, geometry, layout, topology and time satisfying certain constraints. Since resources, fund and time are always limited, one has to find solutions to optimal usage of these resources.

The main goal of this book is to introduce some well established and the most recently developed metaheuristics for optimal design of structures. Schematic of the chapters of the present book in one glance is shown in Fig. 1.1.

Most of these methods are either nature-based or physics-based algorithms, Fig. 1.2. Though many design examples are included, however, the results may

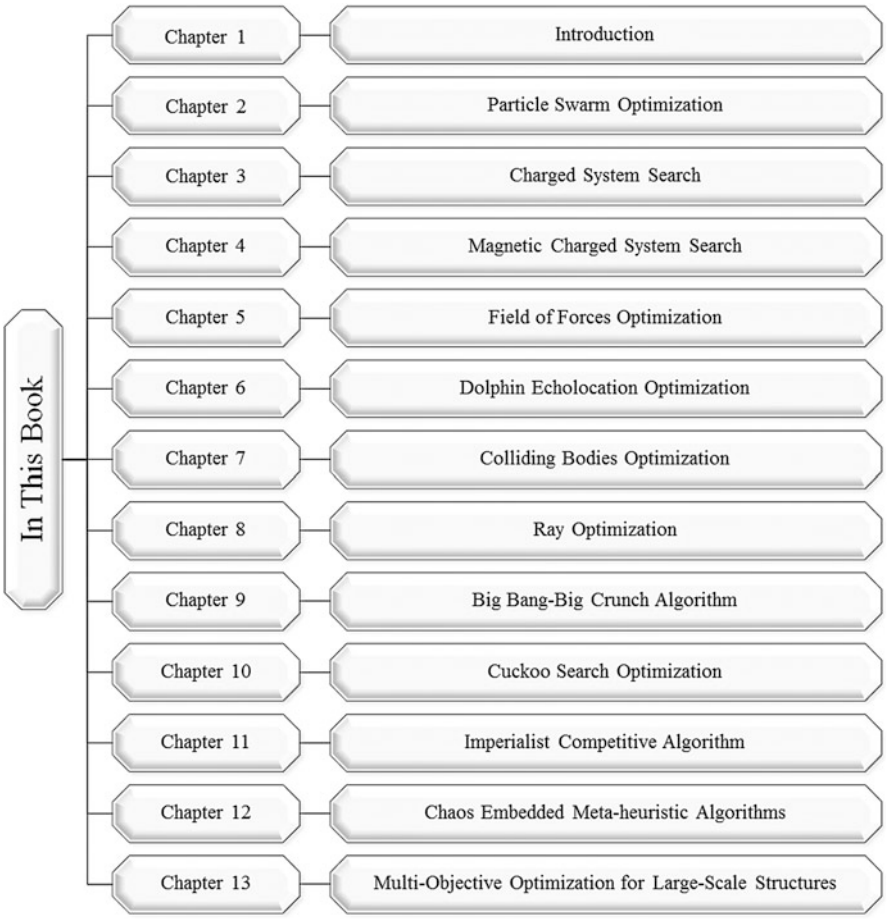

Fig. 1.1 Schematic of the chapters of the present book in one glance

or may not have small constraint violations, and do not constitute the main objective of the book.

1.3 Organization of the Present Book

After this introductory chapter, the remaining chapters of this book are organized in the following manner:

Chapter 2 introduces the well-known Particle Swarm Optimization (PSO) algorithms. These algorithms are nature-inspired population-based metaheuristic algorithms originally accredited to Eberhart, Kennedy and She. The algorithms mimic the social behavior of birds flocking and fishes schooling. Starting with a

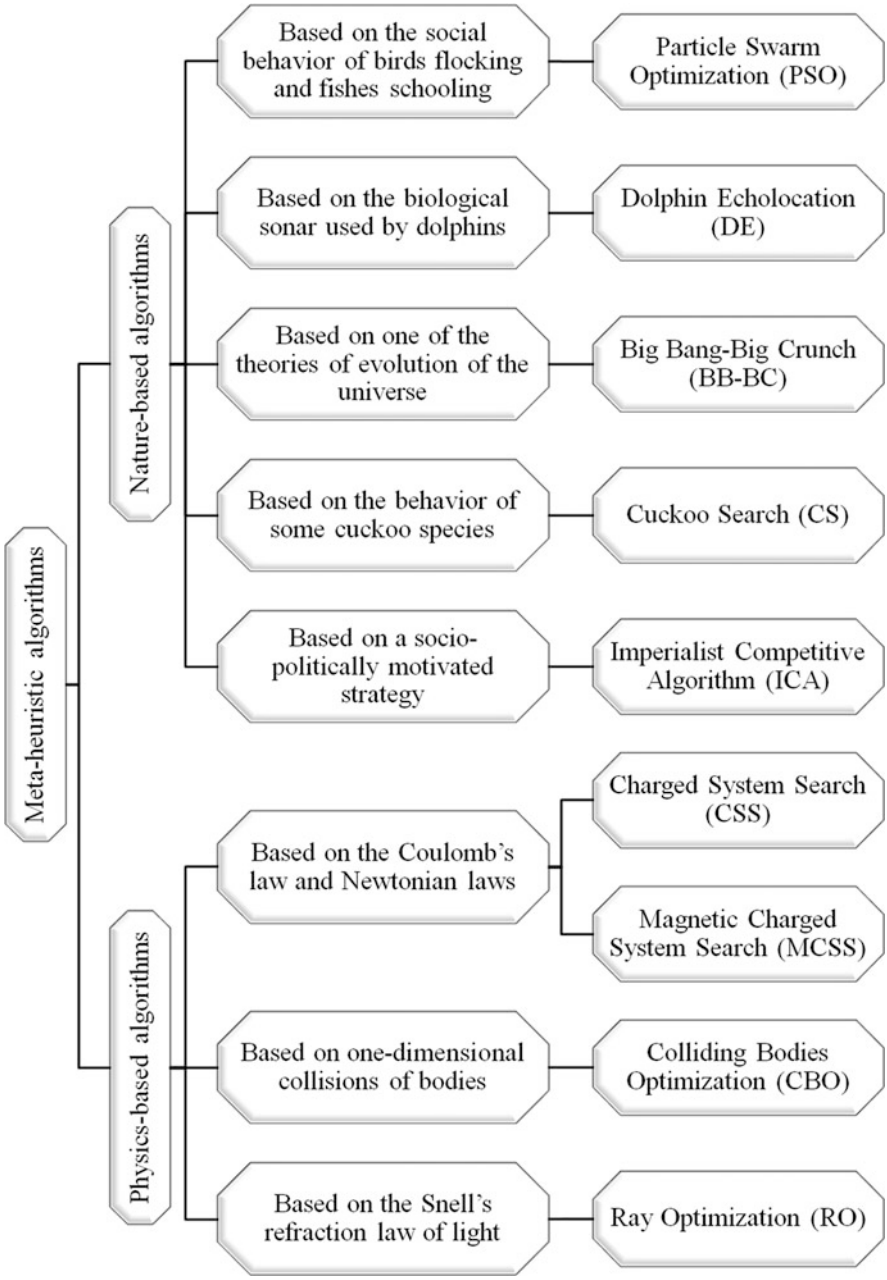

Fig. 1.2 Classification of the metaheuristics presented in this book

randomly distributed set of particles (potential solutions), the algorithms try to improve the solutions according to a quality measure (fitness function). The improvisation is preformed through moving the particles around the search space by means of a set of simple mathematical expressions which model some inter-particle communications. These mathematical expressions, in their simplest and most basic form, suggest the movement of each particle towards its own best experienced position and the swarm's best position so far, along with some random perturbations.

Chapter 3 presents the well established Charged System Search Algorithm (CSS), developed by Kaveh and Talatahari. This chapter consists of two parts. In the first part an optimization algorithm based on some principles from physics and mechanics is introduced. In this algorithm the governing Coulomb law from electrostatics and the Newtonian laws of mechanics are utilized. CSS is a multi-agent approach in which each agent is a Charged Particle (CP). CPs can affect each other based on their fitness values and their separation distances. The quantity of the resultant force is determined by using the electrostatics laws and the quality of the movement is determined using Newtonian mechanics laws. CSS can be utilized in all optimization fields; especially it is suitable for non-smooth or non-convex domains. CSS needs neither the gradient information nor the continuity of the search space. In the second part, CSS is applied to optimal design of skeletal structures and high performance of CSS is illustrated.

Chapter 4 extends the algorithm of the previous chapter and presents the Magnetic Charged System Search, developed by Kaveh, Motie Share and Moslehi. This chapter consists of two parts. In first part, the standard Magnetic Charged System Search (MCSS) is presented and applied to different numerical examples to examine the efficiency of this algorithm. The results are compared to those of the original charged system search method. In the second part, an improved form of the MCSS algorithm, denoted by IMCSS, is presented and also its discrete version is described. The IMCSS algorithm is applied to optimization of truss structures with continuous and discrete variables to demonstrate the performance of this algorithm in the field of structural optimization.

Chapter 5 presents a generalized CSS algorithm known as the Field of Forces Optimization. Although different metaheuristic algorithms have some differences in approaches to determine the optimum solution, however their general performance is approximately the same. They start the optimization with random solutions; and the subsequent solutions are based on randomization and some other rules. With the progress of the optimization process, the power of rules increases, and the power of randomization decreases. It seems that these rules can be modelled by a familiar concept of physics known as the *fields of forces* (FOF). FOF is a concept which is utilized in physics to explain the reason of the operation of the universe. The virtual FOF model is approximately simulated by using the concepts of real world fields such as gravitational, magnetic or electric fields.

Chapter 6 presents the recently developed algorithm known as Dolphin Echolocation Optimization, proposed by Kaveh and Farhoudi. Nature has provided inspiration for most of the man-made technologies. Scientists believe that dolphins are the second to human beings in smartness and intelligence. Echolocation is the

biological sonar used by dolphins and several kinds of other animals for navigation and hunting in various environments. This ability of dolphins is mimicked in this chapter to develop a new optimization method. There are different metaheuristic optimization methods, but in most of these algorithms parameter tuning takes a considerable time of the user, persuading the scientists to develop ideas to improve these methods. Studies have shown that metaheuristic algorithms have certain governing rules and knowing these rules helps to get better results. Dolphin Echolocation takes advantages of these rules and outperforms some of the existing optimization methods, while it has few parameters to be set. The new approach leads to excellent results with low computational efforts.

Chapter 7 contains the most recently developed algorithm so-called Colliding Bodies Optimization proposed by Kaveh and Mahdavi. This chapter presents a novel efficient metaheuristic optimization algorithm called Colliding Bodies Optimization (CBO), for optimization. This algorithm is based on one-dimensional collisions between bodies, with each agent solution being considered as the massed object or body. After a collision of two moving bodies having specified masses and velocities, these bodies are separated with new velocities. This collision causes the agents to move toward better positions in the search space. CBO utilizes simple formulation to find minimum or maximum of functions; also it is internally parameter independent.

Chapter 8 presents the Ray Optimization (RO) Algorithm originally developed by Kaveh and Khayat Azad. Similar to other multi-agent methods, Ray Optimization has a number of particles consisting of the variables of the problem. These agents are considered as rays of light. Based on the Snell's light refraction law when light travels from a lighter medium to a darker medium, it refracts and its direction changes. This behaviour helps the agents to explore the search space in early stages of the optimization process and to make them converge in the final stages. This law is the main tool of the Ray Optimization algorithm. This chapter consists of three parts. In first part, the standard Ray optimization is presented and applied to different mathematical functions and engineering problems. In the second part, RO is employed for size and shape optimization of truss structures. Finally in the third part, an improved ray optimization (IRO) algorithm is introduced and applied to some benchmark mathematical optimization problems and truss structure examples.

Chapter 9 presents a modified Big Bang-Big Crunch (BB–BC) Algorithm. The standard BB–BC method is developed by Erol and Eksin, and consists of two phases: a Big Bang phase, and a Big Crunch phase. In the Big Bang phase, candidate solutions are randomly distributed over the search space. Similar to other evolutionary algorithms, initial solutions are spread all over the search space in a uniform manner in the first Big Bang. Erol and Eksin associated the random nature of the Big Bang to energy dissipation or the transformation from an ordered state (a convergent solution) to a disorder or chaos state (new set of solution candidates).

Chapter 10 presents the Cuckoo Search (CS) Optimization developed by Yang and colleagues. In this chapter CS is utilized to determine optimum design of

structures for both discrete and continuous variables. This algorithm is recently developed by Yang and colleagues, and it is based on the obligate brood parasitic behaviour of some cuckoo species together with the Lévy flight behaviour of some birds and fruit flies. The CS is a population based optimization algorithm and similar to many others metaheuristic algorithms starts with a random initial population which is taken as host nests or eggs. The CS algorithm essentially works with three components: selection of the best by keeping the best nests or solutions; replacement of the host eggs with respect to the quality of the new solutions or Cuckoo eggs produced based randomization via Lévy flights globally (exploration); and discovering of some cuckoo eggs by the host birds and replacing according to the quality of the local random walks (exploitation).

Chapter 11 presents the Imperialist Competitive Algorithm (ICA) proposed by Atashpaz et al. ICA is a multi-agent algorithm with each agent being a country, which is either a colony or an imperialist. These countries form some empires in the search space. Movement of the colonies toward their related imperialist, and imperialistic competition among the empires, form the basis of the ICA. During these movements, the powerful Imperialists are reinforced and the weak ones are weakened and gradually collapsed, directing the algorithm towards optimum points.

Chapter 12 is an introduction is provided to Chaos Embedded Metaheuristic Algorithms. In nature complex biological phenomena such as the collective behaviour of birds, foraging activity of bees or cooperative behaviour of ants may result from relatively simple rules which however present nonlinear behaviour being sensitive to initial conditions. Such systems are generally known as "deterministic nonlinear systems" and the corresponding theory as "chaos theory". Thus real world systems that may seem to be stochastic or random, may present a nonlinear deterministic and chaotic behaviour. Although chaos and random signals share the property of long term unpredictable irregular behaviour and many of random generators in programming softwares as well as the chaotic maps are deterministic; however chaos can help order to arise from disorder. Similarly, many metaheuristics optimization algorithms are inspired from biological systems where order arises from disorder. In these cases disorder often indicates both non-organized patterns and irregular behaviour, whereas order is the result of self-organization and evolution and often arises from a disorder condition or from the presence of dissymmetries. Self-organization and evolution are two key factors of many metaheuristic optimization techniques. Due to these common properties between chaos and optimization algorithms, simultaneous use of these concepts can improve the performance of the optimization algorithms. Seemingly the benefits of such combination is a generic for other stochastic optimization and experimental studies confirmed this; although, this has not mathematically been proven yet.

Chapter 13 consists of a multi-objective optimization method to solve large-scale structural problems in continuous search space. This method is based on the Charged System Search, which has been used for single objective optimization in chapter 3. In this study the aim is to develop a multi-objective optimization algorithm with higher convergence rate compared to the other well-known methods

to enable to deal with multi-modal optimization problems having many design variables. In this method, the CSS algorithm is utilized as a search engine in combination with clustering and particle regeneration procedures. The proposed method is examined for four mathematical functions and two structural problems, and the results are compared to those of some other state-of-art approaches.

Finally it should be mentioned that most of the metaheuristic algorithms are attractive, because each one has its own striking features. However, the one which is simple, less parameter dependent, easy to implement, and has a good balance between exploration and exploitation, higher capability to avoid being trapped in local optima, higher accuracy and applicable to wider types of problems and can deal with higher number of variables, can be considered as the most attractive for engineering usage.

In order to have the above features partially or collectively, sometimes it is necessary to design hybrid algorithms. There are many such algorithms and a successful example of this is that of Kaveh and Talatahari [8].

Finally, the author strongly believes that optimal analysis, introduced by Kaveh [9,10], can provide an important means for optimal design of future large-scale structures. Metaheuristic algorithms often require a large number of analyses and optimal analysis can play an important role in reducing the computational cost of the design.

References

1. Majid KI (1974) Optimum design of structures. Newness-Butterworth, UK
2. Kirsch U (1993) Structural optimization: fundamentals and applications. Springer, Berlin
3. Gonzalez TF (ed) (2007) Handbook of approximation algorithms and metaheuristics, Computer and information science series. Chapman & Hall/CRC, Boca Raton, FL
4. Yang X-S (2010) Nature-inspired metaheuristic algorithms, 2nd edn. Luniver Press, UK
5. Glover F, Kochenberger GA (eds) (2003) Handbook of metaheuristics. Kluwer Academic Publishers, Dordrecht
6. Voß S, Martello S, Osman IH, Roucairol C (eds) (1999) Metaheuristics: advances and trends in local search paradigms for optimization. Kluwer Academic Publishers, Dordrecht
7. Osman IH, Laporte G (1996) Metaheuristics: a bibliography. Ann Oper Res 63:513–623
8. Kaveh A, Talatahari S (2009) Particle swarm optimizer, ant colony strategy and harmony search scheme hybridized for optimization of truss structures. Comput Struct 87:267–293
9. Kaveh A (2013) Optimal analysis of structures by concepts of symmetry and regularity. Springer, Wien
10. Kaveh A (2014) Computational structural analysis and finite element methods. Springer, Wien

Chapter 2
Particle Swarm Optimization

2.1 Introduction

Particle Swarm Optimization (PSO) algorithms are nature-inspired population-based metaheuristic algorithms originally accredited to Eberhart, Kennedy and Shi [1, 2]. The algorithms mimic the social behavior of birds flocking and fishes schooling. Starting form a randomly distributed set of particles (potential solutions), the algorithms try to improve the solutions according to a quality measure (fitness function). The improvisation is preformed through moving the particles around the search space by means of a set of simple mathematical expressions which model some inter-particle communications. These mathematical expressions, in their simplest and most basic form, suggest the movement of each particle towards its own best experienced position and the swarm's best position so far, along with some random perturbations. There is an abundance of different variants using different updating rules, however.

Though being generally known and utilized as an optimization technique, PSO has its roots in image rendering and computer animation technology where Reeves [3] defined and implemented a particle system as a set of autonomous individuals working together to form the appearance of a fuzzy object like a cloud or an explosion. The idea was to initially generate a set of points and to assign an initial velocity vector to each of them. Using these velocity vectors each particle changes its position iteratively while the velocity vectors are being adjusted by some random factors.

Reynolds [4] added the notion of inter-object communication to Reeves' particle system to introduce a flocking algorithm in which the individuals were able to follow some basic flocking rules such as trying to match each other's velocities. Such a system allowed for modeling more complex group behaviors in an easier and more natural way.

Kennedy and Eberhart [1] while trying to "graphically simulate the graceful but unpredictable choreography of a bird flock" came across the potential optimization capabilities of a flock of birds. In the course of refinement and simplification of their

A. Kaveh, *Advances in Metaheuristic Algorithms for Optimal Design of Structures*,
DOI 10.1007/978-3-319-05549-7_2, © Springer International Publishing Switzerland 2014

paradigm they discussed that the behavior of the population of agents that they were suggesting follows the five principles of *swarm* intelligence articulated by Millonas [5]. First, the proximity principle: the population should be able to carry out simple space and time computations. Second, the quality principle: the population should be able to respond to quality factors in the environment. Third, the principle of diverse response: the population should not commit its activities along excessively narrow channels. Fourth, the principle of stability: the population should not change its mode of behavior every time the environment changes. Fifth, the principle of adaptability: the population must be able to change behavior mode when it's worth the computational price. They also mentioned that they compromisingly call their mass-less volume-less population members *particles* in order to make the use of concepts like velocity and acceleration more sensible. Thus, the term particle swarm optimization was coined.

2.2 PSO Algorithm

2.2.1 Development

As Kennedy and Eberhart [1] indicated appropriately particle swarm optimization is probably best presented and understood by explaining its conceptual development. Hence, the algorithms transformation process from its earliest stages to its current canonical form is briefly reviewed in this section. Future discussion on the main aspects and issues would be more easily done in this way.

The earliest attempt to use the concept for social behavior simulation carried out by Kennedy and Eberhart [1] resulted in a set of agents randomly spread over a torus pixel grid which used two main strategies: nearest neighbor velocity matching and craziness. At each iteration a loop in the program determined for each agent which other agent was its nearest neighbor, then assigned that agent's X and Y velocities to the agent in focus. As it is predictable, it has been viewed that sole use of such a strategy will quickly settle down the swarm on a unanimous, unchanging direction. To avoid this, a stochastic variable called craziness was introduced. At each iteration, some change was added to randomly chosen X and Y velocities. This introduced enough variation into the system to give the simulation a "life-like" appearance. The above observation points out one of the most necessary features of PSO which indicates its seemingly unalterable non-deterministic nature: incorporation of randomness.

Kennedy and Eberhart took the next step by replacing the notion of "roost" (a place that the birds know previously) in Heppner and Grenander [6] by "food" (for which the birds must search) and therefore converted the social simulation algorithm into an optimization paradigm. The idea was to let the agents (birds) find

an unknown favorable place in the search space (food source) through capitalizing on one another's knowledge. Each agent was able of remembering its best position and knowing the best position of the whole swarm. The extremum of the mathematical function to be optimized can be thought of as the food source. After a series of minor alterations and elimination of the ancillary variables, the updating rules for calculating the next position of a particle was introduced as:

$$v_{i,j}^{k+1} = v_{i,j}^{k} + c_1 r_1 \left(xbest_{i,j}^{k} - x_{i,j}^{k} \right) + c_2 r_2 \left(xgbest_{j}^{k} - x_{i,j}^{k} \right) \tag{2.1}$$

$$x_{i,j}^{k+1} = x_{i,j}^{k} + v_{i,j}^{k+1} \tag{2.2}$$

where ${x_{i,j}}^{k}$ and ${v_{i,j}}^{k}$ are the jth component of the ith particle's position and velocity vector respectively in the kth iteration, r_1 and r_2 are two random numbers uniformly distributed in the range (1,0), xbest_i and xgbest indicate the best positions experienced so far by the ith particle and the whole swarm, respectively, c_1 and c_2 are two parameters representing the particle's confidence in itself (cognition) and in the swarm (social behavior), respectively. These two parameters were set equal to 2 in the initial version of the PSO presented by Kennedy and Eberhart [1] so that the particles would overfly the target about half the time. These two parameters are among the most important parameters of the algorithm in that they control the balance between exploration and exploration tendencies. A relatively high value of c_1 will encourage the particles to move towards their local best experiences while higher values of c_2 will result in faster convergence to the global best position.

Although the above formulation embodies the main concept of PSO that has survived over time and forms the skeleton of quite all subsequent variants, it has still been subject to amendment. Eberhart et al. [7] introduced a maximum velocity parameter, V_{max}, in order to prevent particles from leaving the search space. Shi and Eberhart [8] discussed the role of the three terms of (2.1) and concluded that the first term, previous velocity of the particle, has an important effect of global and local search balance. By eliminating this term the particles can note leave their initially encircled portion of the search space and the search space will shrink gradually over time. This will be equivalent to a local search procedure. Alternatively, by giving the previous velocity term relatively higher effects the particles will be reluctant to converge to the known good positions. They will instead tend to explore unseen regions of the search space. This could be conceived as global search tendency. Both the local search and global search will benefit solving some kinds of problems. Therefore, an inertia weight w is thus introduced into (2.1) in order to maintain balance between these to effects:

$$v_{i,j}^{k+1} = w v_{i,j}^{k} + c_1 r_1 \left(xlbest_{i,j}^{k} - x_{i,j}^{k} \right) + c_2 r_2 \left(xgbest_{j}^{k} - x_{i,j}^{k} \right) \tag{2.3}$$

Shi and Eberhart [8] indicated that the inertia weight can be a positive constant or even a positive linear or nonlinear function of time. They examined the use of constant values in the range [0, 1.4] for the benchmark problem of Schaffer's f6 function and concluded the range [0.9, 1.2] to be appropriate. Later, Eberhart and Shi [9] indicated that the use of the inertia weight w, which decreases linearly from about 0.9 to 0.4 during a run, provides improved performance in a number of applications. Many different research works has focused on inertia weight parameter and different strategies have been proposed ever since. A brief discussion on these methods and strategies will be presented in the next section.

Later, Clerc [10] indicated that the use of a constriction factor may be necessary to insure convergence of the particle swarm algorithm and proposed an alternative formulation for the velocity vector:

$$v_{i,j}^{k+1} = \chi\left[v_{i,j}^{k} + c_1 r_1\left(xlbest_{i,j}^{k} - x_{i,j}^{k}\right) + c_2 r_2\left(xgbest_{j}^{k} - x_{i,j}^{k}\right)\right] \tag{2.4}$$

$$\chi = \frac{2}{\left|2 - \phi - \sqrt{\phi^2 - 4\phi}\right|} \quad where \quad \phi = c_1 + c_2, \quad \phi > 4 \tag{2.5}$$

Schematic movement of a particle is illustrated in Fig. 2.1.

Such a formulation was intended to impose some restriction on velocity vectors and thus to prevent divergence. Eberhart and Shi [9] compared the use of inertia weights and constriction factors in particle swarm optimization and concluded that the two approaches are equivalent and could be interchangeably used by proper parameter setting. They also indicated that the use of constriction factor does not eliminate the need for V_{max} parameter unlike what might be assumed at the first glance. Though the two approaches are shown to be equivalent they both survived and have been continually used by researchers. Simultaneous utilization of inertia weight and constriction factor can also be found in the literature (e.g. see [11] among others).

2.2.2 PSO Algorithm

The general structure of a canonical PSO algorithm is as follows [12]:

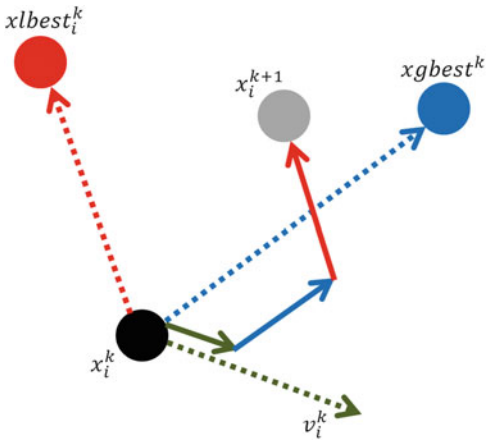

Fig. 2.1 Scematic movement of a particle based on (2.4)

```
procedure Particle Swarm Optimization
begin
    Initialize xi, vi and xbesti for each particle i;
    while (not termination condition) do
    begin
        for each particle i
            Evaluate objective function;
            Update xbesti
        end
        for each i
            Set g equal to index of neighbor with best xbesti;
            Use g to calculate vi;
            Update xi = xi + vi;
            Evaluate objective function;
            Update xbesti
        end
    end
end
```

2.2.3 Parameters

Like any other metaheuristic algorithm PSO's performance is dependent on the values of its parameters. The optimal values for the parameters depend mainly on the problem at hand and even the instance to deal with and on the search time that

the user wants to spend for solving the problem [13]. In fact the main issue is to provide balance between exploration and exploitation tendencies of the algorithm.

Total number of particles, total number of iterations, inertia weight and/or constriction factor, and cognition and social behavior coefficients (c_1 and c_2) are the main parameters that should be considered in a canonical PSO. The total number of iterations could be replaced with a desired precision or any other termination criterion. In general, the search space of an n-dimensional optimization problem can be conceived as an n-dimensional hyper-surface. The suitable values for a metaheuristic's parameters depend on relative ruggedness and smoothness of this hyper-space. For example, it is imaginable that in a smoother hyper-space fewer number of particles and iteration numbers will be required. Moreover, in a smoother search space there will be fewer local optimal positions and less exploration effort will be needed while in a rugged search space a more through exploration of the search space will be advisable.

Generally speaking, there are two different strategies for parameter value selection, namely offline parameter initialization and online parameter tuning [13]. In off-line parameter initialization, the values of different parameters are fixed before the execution of the metaheuristic. These values are usually decided through empirical study. It should be noted that deciding about a parameter of a metaheuristic algorithm while keeping the others fixed (i.e. one-by-one parameter selection) may result in misleading observations since the interactions of the parameters are not taken into account. However, it is the common practice in the literature since examining combinations of the algorithm parameters might be very time-consuming. To perform such an examination, when desired, a meta-optimization approach may be performed i.e. the algorithm parameters can be considered as design variables and be optimized in an overlying level.

The main drawback of the off-line approaches is their high computational cost since the process should be repeated for different problems and even for different instances of the same problem. Moreover, appropriate values for a parameter might change during the optimization process. Hence, online approaches that change the parameter values during the search procedure must be designed. Online approaches may be classified in two main groups [13]; dynamic approaches and adaptive approaches. In a dynamic parameter updating approach, the change of the parameter value is performed without taking into account the search progress. The adaptive approach changes the values according to the search progress.

Attempts have been made to introduce guidelines and strategies for selection of PSO parameter. Shi and Eberhart [14] analyzed the impact of inertia weight and maximum allowable velocity on the performance of PSO and provided some guidelines for selecting these two parameters. For this purpose they utilized different combinations of w and V_{max} parameters to solve the Schaffer's f6 test function while keeping other parameters unchanged. They concluded that when V_{max} is small ($<= 2$ for the f6 function), an inertia weight of approximately 1 is a good choice, while when V_{max} is not small ($> = 3$), an inertia weight $w = 0.8$ is a good choice. They declared that in absence of proper knowledge regarding the selection of V_{max}, it is also a good choice to set V_{max} equal to X_{max} and an inertia weight

$w = 0.8$ is a good starting point. Furthermore if a time varying inertia weight is employed, even better performance can be expected. As the authors indicated, such an empirical approach using a small benchmark problem cannot be easily generalized.

Carlisle and Dozier [15] proposed another set of guidelines based on evidence from six experiments. They recommended to start with an asynchronous constricted algorithm setting $r_1 = 2.8$ and $r_2 = 1.3$. However, no directives are provided in order to progress from this initial setting.

Trelea [16] used dynamic system theory for a theoretical analysis of the algorithm producing some graphical guidelines for parameter selection. A simplified deterministic one-dimensional PSO was used for this study. Trelea indicates that the results are predictably dependent on the form of the objective function. The discussion is supported by experiments on five benchmark functions.

Zhang et al. [17] suggested some optimal ranges for constriction factor and V_{max} to X_{max} ratio parameters based on experimental study on nine mathematical functions. The optimal range for constriction factor is claimed to be [4.05, 4.3] while for V_{max} to X_{max} ratio the range [0.01, 1] is recommended.

More recently Pedersen [18] carried out Meta-Optimization to tune the PSO parameters. A table is presented to help the practitioner choose appropriate PSO parameters based on the dimension of the problem at hand and the total number of function evaluation that is intended to be performed. Performance evaluation of PSO is performed using some mathematical functions. As mentioned before, the results of the above-mentioned off-line parameter tuning studies are all problem-dependent and can not be claimed as universally optimal.

Many different online tuning strategies are also proposed for different PSO parameters. For inertia weight, methods such as Random Inertia Weight, Adaptive Inertia Weight, Sigmoid Increasing/Decreasing Inertia Weight, Linear Decreasing Inertia Weight, Chaotic Inertia Weight and Chaotic Random Inertia Weight, Oscillating Inertia Weight, Global-Local Best Inertia Weight, Simulated Annealing Inertia Weight, Natural Exponent Inertia Weight Strategy, Logarithm Decreasing Inertia Weight, Exponent Decreasing Inertia Weight are reported in the literature. All of these methods replace the inertia weight parameter with a mathematical expression which is either dependent on the state of the search process (e.g. global best solution, current position of the particle etc.) or not. Bansal et al. [19] examined the above mentioned inertia weight strategies for a set of five mathematical problems and concluded that Chaotic Inertia Weight is the best strategy for better accuracy while Random Inertia Weight strategy is best for better efficiency. This shows that the choice of a suitable inertia weight strategy depends not only on the problem under consideration, but also on the practitioner's priorities.

Other adaptive particle swarm optimization algorithms could be found in the literature [20].

2.2.4 Premature Convergence

One of the main advantages of PSO is its ability to attain reasonably good solutions relatively fast. At the same time, this is probably the algorithm's most recognized drawback. In fact, Angeline [21] while studying PSO versus Evolutionary optimization techniques showed that although PSO discovers good quality solutions much faster than evolutionary algorithms, it does not improve the quality of the solutions as the number of generations is increased. This is because of the particles getting clustered in a small region of the search space and thus the loss of diversity [22]. Improving the exploration ability of PSO has been an active research topic in recent years [20].

Attempts have been made in order to improve the algorithm's exploration capabilities and thus to avoid premature convergence. van den Bergh and Engelbrecht [23] introduced a Guaranteed Convergence PSO (GCPSO) in which particles perform a random search around *xgbest* within a radius defined by a scaling factor. The algorithm is reported to perform better than original PSO in unimodal problems while producing similar results in multimodal ones. The scaling factor however is another parameter for which prior knowledge may be required to be optimally set.

Krink et al. [24] proposed a collision free PSO where particles attempting to gather about a sub-optimal solution bounce away. A random direction changer, a realistic bounce and a random velocity changer where used as three bouncing strategies. The latter two are reported to significantly improve the exploration capabilities of the algorithm and obtain better results especially in multimodal problems.

Implementing diversity measures is another way to control swarm stagnation. Riget and Vesterstrøm [25] utilized such a measure along with alternative attraction and repulsion of the particles to and from the swarm best position. Repulsion could be induced by inverting the velocity update rule. The approach improves the performance in comparison to canonical PSO, especially when problems under consideration are multimodal.

Silva et al. [26] introduced a predator-prey optimization system in which a predator particle enforces other particles to leave the best position of the search space and explore other regions. Improved performance is reported based on experiments carried out on four high-dimensional test functions.

Jie et al. [27] introduced an adaptive PSO with feedback control of diversity in order to tune the inertia weight parameter and alleviate the premature convergence. The improvements increase the robustness and improve the performance of the standard PSO in multimodal functions.

Wang et al. [20] proposed a self-adaptive learning based particle swarm optimization which used four PSO based search strategies on probabilistic basis according to the algorithms performance in previous iterations. The use of different search strategies in a learning based manner helps the algorithm to handle problems with different characteristics at different stages of optimization process. 26 test functions

with different characteristics such as uni-modality, multi-modality, rotation, ill-condition, mis-scale and noise are considered and the results are compared with eight other PSO variants.

Kaveh and Zolghadr [28] introduced a democratic particle swarm optimization (DPSO) which uses the updating information of a particle form a more diverse set of sources instead of using local and global best solutions merely. An eligibility parameter is introduced which determines which particles to incorporate when updating a specific particle. The improved algorithm is compared to the standard one for some mathematical and structural problems. The performance is improved for the problems under consideration.

2.2.5 Topology

While *xgbest* of (2.1) is considered to be the whole swarm's global best position in canonical PSO, this is not necessarily always the case. Different topologies have been defined and used for inter-particle communications in PSO [29, 30]. In fact in updating a particle's position, *xgbest* could mean the best particle position of a limited neighborhood to which the particle is connected instead of the whole swarm. It has been shown that the swarm topologies in PSO can remarkably influence the performance of the algorithm. Figure 2.2 shows some of the basic PSO neighborhood topologies introduced by Mendes et al. [29]. Many other topologies can be defined and used.

These different topologies affect the way that information circulates between the swarm's particles and thus can control exploration-exploitation behavior and convergence rate of the algorithm. Canonical PSO uses the fully-connected topology in which all of the particles are neighbors. Such a topology exhibits a fast (and probably immature) convergence since all of the particles are directly linked to the global best particle and simultaneously affected by it. Thus, the swarm does not explore other areas of the search space and would most probably get trapped in local optima.

Ring topology which is a usual alternative to fully-connected topology represents a regular graph with the minimum node degrees. This could be considered the slowest way of information circulation between the particles and is supposed to result in the slowest rate of convergence since it takes a relatively long time for information of the best particle to reach the other end of the ring.

Other neighborhood topologies are somewhere in between. Predictably, the effect of different neighborhood topologies on effectiveness and efficiency of the algorithm is problem dependent and is more or less empirically studied.

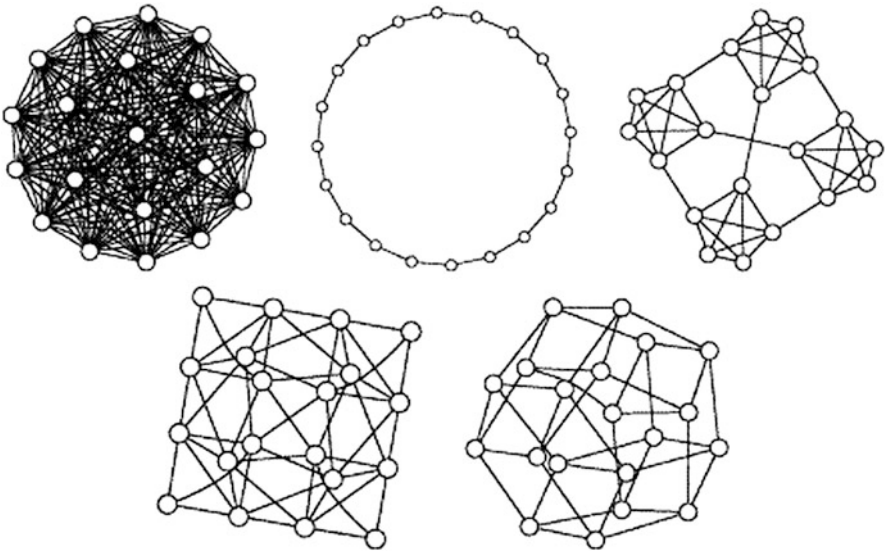

Fig. 2.2 Some topologies for PSO neighborhoods [29]. *Fully-connected*: where all vertexes are connected to every other; *Ring*: where every vertex is connected to two others; *Four clusters*: with four cliques connected among themselves by gateways; *Pyramid*: a triangular wire-frame pyramid, and *Square*: which is a mesh where every vertex has four neighbors that wraps around on the edges as a torus

2.2.6 Biases

It is shown that many metaheuristic optimization algorithms, including PSO, are biased toward some specific regions of the search space. For example, they perform best when the optimum is located at or near the center of the initialization region, which is often the origin [31]. This is particularly true when the information from different members of the population is combined using some sort of averaging operator [32]. Since many of the benchmark optimization problems have their global optimal solutions at or near the origin, such a biased behavior can make the performance evaluation of the algorithms problematic. Different approaches have been taken in order to expose and probably alleviate this bias while testing PSO. Angeline [32] popularized a method called Region Scaling initially proposed by Gehlhaar and Fogel [33]. The method, tries to let the origin outside the initial region covered by the particles by generating the initial solutions in a portion of the search space that does not include origin. Monson and Seppi [31] showed that origin-seeking biases depend on the way that the positions of the particles are updated and Region Scaling method could not be sufficient for all motion rules. They introduced a Center Offset method in which the center of the benchmark function under consideration was moved to a different location of the search space. Suganthan et al. [34] also recommended the use of non-biased shifted or rotated benchmark problems.

Clerc [35] showed that this biased behavior can be attributed to the confinement method used i.e. the method by which the particles are prevented from leaving the search space. A hybrid confinement method is introduced and claimed to be useful in terms of reducing the bias.

Attempts have also been made to propose improved non-biased variants (e.g. Wilke et al. [36]). This is however of less generality in case of unbiased performance comparison because it does not have any effect on the other existing algorithms.

2.3 Hybrid Algorithms

A popular way of producing new improved algorithms is to hybridize two or more existing ones in an attempt to combine favorable features while omitting undesirable aspects. Some of the best results for the real-life and classical optimization problems are obtained using hybrid methods [37]. Numerous different hybrid algorithms using PSO as the main or the supplementary ingredient have been proposed usually in the context of some specific application domain for which that hybrid is particularly well suited [38]. A selection of these methods and approaches is briefly mentioned here along with some examples.

Hybridizing PSO with other metaheuristic algorithms seems to be one of the most popular strategies. This is mainly because the resulting algorithm maintains positive characteristics of metaheuristic algorithms such as global search capability, little dependence on starting point, no need to gradient information, and applicability to non-smooth or non-convex domains. The other metaheuristic algorithm (s) to be hybridized with PSO can be either single-agent or population-based.

Simulated Annealing (SA) [39] is a single-agent metaheuristic algorithm that has been successfully hybridized with PSO. It has been shown in the literature [40] that SA algorithms, when subject to very low variations of temperature parameters, and when the solution search for each temperature can reach an equilibrium condition have very high chances of finding the global optimal solution. Moreover, the metropolis process in SA provides an ability of jumping away from a local. However, SA algorithms require very slow temperature variations and thus increase the required computational effort. On the other hand, although PSO exhibits relatively fast convergence rate, is easy to implement, and is able to find local optimal solutions in a reasonable amount of time, it is notorious of premature convergence i.e. getting trapped in local optima. Therefore, combining these two algorithms in a judicious way will probably result in a hybridized algorithm with improved performance [41]. Execution of PSO and SA algorithms can be either alternative or sequential. In an alternative execution every member of the PSO swarm can be considered as a SA single-agent at the end of each iteration. Instead, in a sequential execution, the final local solution found by PSO could be considered as a starting point for SA.

As another single-agent metaheuristic algorithm, Tabu Search algorithm (TS) [42, 43] can have the same effect as SA in hybridization with PSO. The global search could be left to PSO while TS attempts to improve the suboptimal solutions found by PSO in a local search process. In these hybridized algorithms TS alleviates premature convergence of PSO while PSO alleviates excessive required computational effort of TS [44].

Hybridization of PSO with other population-based metaheuristic algorithms is more popular. In this case hybridization might signify different meanings. In some hybridized schemes some techniques are simply borrowed from other algorithms. For example Løvebjerg et al. [45] borrowed the breeding technique from GAs i.e. along with standard PSO updating rules pairs of particles could be chosen to breed with each other and produce off-springs. Moreover, to keep away from suboptimal solutions subpopulations were introduced.

Another approach to be mentioned is to use different metaheuristics simultaneously. Krink and Løvebjerg [46] introduced a lifecycle model that allowed for use of PSO, GA or hill climber by each particle depending on the particle's own preference based on its memory of the best recent improvements. Kaveh and Talatahari [47] introduced a hybridized HPSACO algorithm in which particle swarm optimizer with passive congregation (PSOPC) was used to perform global search task while Ant Colony Optimization (ACO) [48] was utilized for updating positions of particles to attain the feasible solution space and Harmony Search (HS) [49] algorithm was employed for dealing with variable constraints.

In the above-mentioned approaches the position updating rules of the original algorithms need not to be changed. The algorithms are merely operating in combination to each other. Another hybridization approach, however, could be based on combining the updating rules. Higashi and Iba [50] combined GA's Gaussian mutation with velocity and position update rules of PSO. Juang [51] incorporated mutation, crossover and elitism. As another example, Kaveh and Talatahari [52] introduced some of the positive aspects of PSO like directing the agents toward the global best and the local best positions into Charged System Search (CSS) [53] algorithm to improve its performance.

PSO could also be hybridized with techniques and tools other than metaheuristic algorithms. Liu and Abraham [54] hybridized a turbulent PSO with a fuzzy logic controller to produce a Fuzzy Adaptive TPSO (FATPSO). The fuzzy logic controller was used for adaptively tune the velocity parameters during an optimization in order to balance exploration and exploitation tendencies. Zahara et al. [55] hybridized Nelder-Mead simplex search and particle swarm optimization for constrained engineering design problems. A hybrid PSO-simplex method was also used for damage identification of delaminated beams by Qian et al. [56].

2.4 Discrete PSO

Though PSO has been introduced and more commonly utilized for continuous optimization problems, it can be equally applied to discrete search spaces. A simple and frequently used method to use PSO in discrete problems is to transform the real-valued vectors found by a continuous PSO algorithm into discrete ones. To do this the nearest permitted discrete values could be replaced with any value selected by agents i.e. a rounding function could be used [57]. However, many discrete and binary PSO variants have been developed that work in discrete search space directly.

The first discrete binary version of PSO is developed by Kennedy and Eberhart [58]. They kept the particle position updating rule unchanged and replaced the velocity in each vector by the probability of a bit in position vector taking the value 1. In other words, if for example $v_{i,j} = 0.20$, then there is a twenty percent chance that $x_{i,j}$ will be a one, and an eighty percent chance it will be a zero. In order to keep $v_{i,j}$ in interval [0,1] a sigmoid transformation function was used.

More recently, Chen et al. [59] have proposed a set-based PSO for discrete optimization problems. They have replaced the candidate solutions and velocity vectors as crisp sets and sets with possibilities, respectively. The arithmetic operators in position updating rules are replaced by the operators and procedures defined on such sets.

2.5 Democratic PSO for Structural Optimization

2.5.1 Description of the Democratic PSO

As discussed earlier different updating strategies have been proposed for PSO resulting in many different variants. Mendes et al. [29] have proposed a fully informed PSO for example, in which each particle uses the information from all of the other particles in its neighborhood instead of just the best one. It has been shown that the fully informed PSO outperforms the canonical version in all of the mathematical functions under consideration. In a conceptually similar work Kaveh and Zolghadr [28] have proposed a Democratic PSO for structural optimization problems with frequency constraints. Here a brief description of the Democratic algorithm is presented as an improved PSO version in the field of structural optimization. The structural optimization under consideration is then introduced in the following section and the results are then compared to those of the canonical PSO on the same problems reported by Gomes [60].

As indicated before, canonical PSO is notorious for premature convergence and this can be interpreted as a lack of proper exploration capability. In fact in the standard PSO all of the particles are just being eagerly attracted toward better solutions. And by each particle, moving toward the best position experienced by

itself and by the whole swarm so far is thought of as the only possible way of improvement. Naturally, such an enthusiasm for choosing the shortest possible ways to accomplishment results in some of the better regions of the search space being disregarded.

In a sense, it can be said that the particles of the canonical PSO are only motivated by selfishness (their own preference) and tyranny (the best particle's dictation). Except for their own knowledge and that of the best particle so far, they do not take the achievements of the other members of the swarm into account i.e. the information is not appropriately shared between the members of the swarm.

In order to address this problem, the velocity vector of the democratic PSO is defined as:

$$v_{i,j}^{k+1} = \chi\left[\omega v_{i,j}^{k} + c_1 r_1\left(xlbest_{i,j}^{k} - x_{i,j}^{k}\right) + c_2 r_2\left(xgbest_{j}^{k} - x_{i,j}^{k}\right) + c_3 r_3 d^{k}{}_{i,j}\right] \quad (2.6)$$

in which $d^{k}{}_{i,j}$ is the jth variable of the vector D for the ith particle. The vector D represents the democratic effect of the other particles of the swarm on the movement of the ith particle. r_3 is a random number uniformly distributed in the range (1,0). Parameter c_3 is introduced to control the weight of the democratic vector. Here, the vector D is taken as:

$$D_i = \sum_{k=1}^{n} Q_{ik}\left(X_k - X_i\right) \quad (2.7)$$

where Q_{ik} is the weight of the kth particle in the democratic movement vector of the ith particle and can be defined as:

$$Q_{ik} = \frac{E_{ik}\frac{obj_{best}}{obj(k)}}{\sum_{j=1}^{n} E_{ij}\frac{obj_{best}}{obj(j)}} \quad (2.8)$$

in which obj stands for objective function value; obj_{best} is the value of the objective function for the best particle in the current iteration; X is the particle's position vector; E is the eligibility parameter and is analogous to parameter P in CSS [53]. In a minimization problem E can be defined as:

$$E_{ik} = \begin{cases} 1 & \dfrac{obj(k) - obj(i)}{obj_{worst} - obj_{best}} > rand \vee obj(k) < obj(i) \\ 0 & else \end{cases} \quad (2.9)$$

where obj_{worst} and obj_{best} are the values of the objective function for the worst and the best particles in the current iteration, respectively. The symbol $\vee$ stands for union.

Schematic movement of a particle is illustrated in Fig. 2.3.

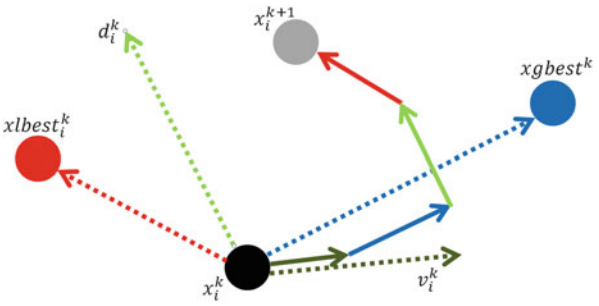

Fig. 2.3 Schmatic movement of a particle based on (2.6)

Since a term is added to the velocity vector of PSO, the parameter χ should be decreased in order to avoid divergence. Here, this parameter is determined using a trial and error process. It seems that a value in the range (0.4, 0.5) is suitable for the problems under consideration.

As it can be seen, the democratic PSO makes use of the information produced by all of the eligible members of the swarm in order to determine the new position of each particle. In fact, according to (2.9), all of the better particles and some of the worse particles affect the new position of the particle under consideration. This modification enhances the performance of the algorithm in two ways: (1) helping the agents to receive information about good regions of the search space other than those experienced by themselves and the best particle of the swarm and (2) letting some bad particles take part in the movement of the swarm and thus improving the exploration capabilities of the algorithm. Both of the above effects help to alleviate the premature convergence of the algorithm.

Numerical results show that this simple modification which does not call for any extra computational effort meaningfully enhances the performance of the PSO.

2.5.2 Truss Layout and Size Optimization with Frequency Constraints

In a frequency constraint truss layout and size optimization problem the aim is to minimize the weight of the structure while satisfying some constraints on natural frequencies. The design variables are considered to be the cross-sectional areas of the members and/or the coordinates of some nodes. The topology of the structure is not supposed to be changed and thus the connectivity information is predefined and kept unaffected during the optimization process. Each of the design variables should be chosen within a permissible range. The optimization problem can be stated mathematically as follows:

$$\begin{aligned}
&\textit{Find } X = [x_1, x_2, x_3, \ldots., x_n] \\
&\textit{to minimize } P(X) = f(X) \times f_{penalty}(X) \\
&\textit{subjected to} \\
&\omega_j \leq \omega_j^* \quad \text{for some natural frequencies } j \\
&\omega_k \geq \omega_k^* \quad \text{for some natural frequencies } k \\
&x_{imin} \leq x_i \leq x_{imax}
\end{aligned} \tag{2.10}$$

where X is the vector of the design variables, including both nodal coordinates and cross-sectional areas; n is the number of variables which is naturally affected by the element grouping scheme which in turn is chosen with respect to the symmetry and practice requirements; $P(X)$ is the penalized cost function or the objective function to be minimized; $f(X)$ is the cost function, which is taken as the weight of the structure in a weight optimization problem; $f_{penalty}(X)$ is the penalty function which is used to make the problem unconstrained. When some constraints corresponding to the response of the structure are violated in a particular solution, the penalty function magnifies the weight of the solution by taking values bigger than one; ω_j is the jth natural frequency of the structure and ω_j^* is its upper bound. ω_k is the kth natural frequency of the structure and ω_k^* is its lower bound. x_{imin} and x_{imax} are the lower and upper bounds of the design variable x_i, respectively.

The cost function is expressed as:

$$f(X) = \sum_{i=1}^{nm} \rho_i L_i A_i \tag{2.11}$$

where ρ_i is the material density of member i; L_i is the length of member i; and A_i is the cross-sectional area of member i.

The penalty function is defined as:

$$f_{penalty}(X) = (1 + \varepsilon_1 . v)^{\varepsilon_2}, \quad v = \sum_{i=1}^{q} v_i \tag{2.12}$$

where q is the number of frequency constraints.

$$v_i = \begin{cases} 0 & \textit{if the ith constraint is satisfied} \\ \left| 1 - \dfrac{\omega_i}{\omega^*_i} \right| & \textit{else} \end{cases} \tag{2.13}$$

The parameters ε_1 and ε_2 are selected considering the exploration and the exploitation rate of the search space. In this study ε_1 is taken as unity, and ε_2 starts from 1.5 linearly increases to 6 in all test examples. These values penalize the unfeasible solutions more severely as the optimization process proceeds. As a result, in the early stages, the agents are free to explore the search space, but at the end they tend to choose solutions without violation.

2.5.3 Numerical Examples

Four numerical examples from the field of truss layout and size optimization are provided in this section in order to examine the viability of the democratic PSO and to compare it with the canonical PSO to clarify the effect of the modifications. The results are compared with those of the canonical version and some other methods reported in the literature.

Parameter χ is set to 0.5 in all numerical examples while parameter c_3 is set to 4. A Total population of 30 particles is utilized for all of the examples. Each example has been solved 30 times independently. In all the examples, the termination criterion is taken as the number of iterations. Total number of 200 iterations is considered for all of the examples. The side constraints are handled using an HS-based constraint handling technique, as introduced by Kaveh and Talatahari [47], is used. Any other appropriate side constraint handling technique might be used.

2.5.3.1 A 10-bar Truss

For the first example, size optimization of a 10-bar planar is considered. The configuration of the structure is depicted in Fig. 2.4.

This is a well-known benchmark problem in the field of frequency constraint structural optimization. Each of the members' cross-sectional area is assumed to be an independent variable. A non-structural mass of 454.0 kg is attached to all free nodes. Table 2.1 summarizes the material properties, variable bounds, and frequency constraints for this example.

This problem has been investigated by different researchers: Grandhi and Venkayya [61] employing an optimality algorithm, Sedaghati et al. [62] using a sequential quadratic programming and finite element force method, Wang et al. [63] using an evolutionary node shift method, Lingyun et al. [64] utilizing a niche hybrid genetic algorithm, Gomes employing the standard particle swarm optimization algorithm [60] and Kaveh and Zolghadr [65, 66] utilizing the standard and an enhanced CSS, and a hybridized CSS-BBBC with a trap recognition capability.

The design vectors and the corresponding masses of the optimal structures found by different methods are summarized in Table 2.2.

It should be noted that a modulus of elasticity of $E = 6.98 \times 10^{10}$ Pa is used in Gomes [60] and Kaveh and Zolghadr [65]. This will generally result in relatively lighter structures. Considering this, it appears that the proposed algorithm has obtained the best solution so far. Particularly, the optimal structure found by the algorithm is more than 5.59 kg lighter than that of the standard PSO in spite of using smaller value for modulus of elasticity. Using $E = 6.98 \times 10^{10}$ Pa DPSO finds a structure weighted 524.70 kg which is about 13 kg lighter than that of standard PSO. The mean weight and the standard deviation of the results gained by DPSO are 537.80 kg and 4.02 kg respectively, while PSO has obtained a mean weight of

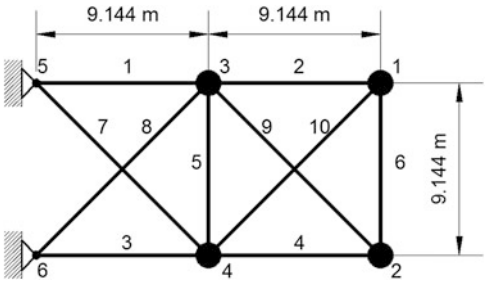

Fig. 2.4 Schematic of the planar 10-bar truss structure

Table 2.1 Material properties, variable bounds and frequency constraints for the 10-bar truss structure

Property [unit]	Value
E (Modulus of elasticity) [N/m^2]	6.89×10^{10}
ρ (Material density) [kg/m^3]	2,770.0
Added mass [kg]	454.0
Design variable lower bound [m^2]	0.645×10^{-4}
Design variable upper bound [m^2]	50×10^{-4}
L (main bar's dimension) [m]	9.144
Constraints on first three frequencies [Hz]	$\omega_1 \geq 7$, $\omega_2 \geq 15$, $\omega_3 \geq 20$

Table 2.2 Optimized designs (cm^2) obtained for the planar 10-bar truss problem (the optimized weight does not include the added masses)

Element number	Grandhi and Venkayya [61]	Sedaghati et al. [62]	Wang et al. [63]	Lingyun et al. [64]	Gomes [60]	Kaveh and Zolghadr [65] Standard CSS	Democratic PSO [28]
1	36.584	38.245	32.456	42.23	37.712	38.811	35.944
2	24.658	9.916	16.577	18.555	9.959	9.0307	15.530
3	36.584	38.619	32.456	38.851	40.265	37.099	35.285
4	24.658	18.232	16.577	11.222	16.788	18.479	15.385
5	4.167	4.419	2.115	4.783	11.576	4.479	0.648
6	2.070	4.419	4.467	4.451	3.955	4.205	4.583
7	27.032	20.097	22.810	21.049	25.308	20.842	23.610
8	27.032	24.097	22.810	20.949	21.613	23.023	23.599
9	10.346	13.890	17.490	10.257	11.576	13.763	13.135
10	10.346	11.452	17.490	14.342	11.186	11.414	12.357
Weight (kg)	594.0	537.01	553.8	542.75	537.98	531.95	532.39

540.89 kg and a standard deviation of 6.84 kg. This means that DPSO performs better than the standard PSO in terms of best weight, average weight, and standard deviation.

Table 2.3 Natural frequencies (Hz) evaluated at the optimized designs for the planar 10-bar truss

Frequency number	Grandhi and Venkayya [61]	Sedaghati et al. [62]	Wang et al. [63]	Lingyun et al. [64]	Gomes [60]	Kaveh and Zolghadr [65]	
						Standard CSS	Democratic PSO [28]
1	7.059	6.992	7.011	7.008	7.000	7.000	7.000
2	15.895	17.599	17.302	18.148	17.786	17.442	16.187
3	20.425	19.973	20.001	20.000	20.000	20.031	20.000
4	21.528	19.977	20.100	20.508	20.063	20.208	20.021
5	28.978	28. 173	30.869	27.797	27.776	28.261	28.470
6	30.189	31.029	32.666	31.281	30.939	31.139	29.243
7	54.286	47.628	48.282	48.304	47.297	47.704	48.769
8	56.546	52.292	52.306	53.306	52.286	52.420	51.389

Table 2.3 represents the natural frequencies of the optimized structures obtained by different methods.

Figure 2.5 compares the convergence curves for the 10-bar planar truss obtained by the democratic PSO and the standard PSO.

The termination criterion is not clearly stated in reference [60]. It is just declared that a combination of three different criteria was simultaneously employed: (1) the differences in the global best design variables between two consecutive iterations, (2) the differences of the global best objective function, and (3) the coefficient of variation of objective function in the swarm. In any case, it seems no improvement is expected from PSO after the 2,000th analysis and hence the execution is terminated.

Comparison of the convergence curves above provides some useful points about the differences of the two algorithms. The standard and the democratic PSO utilize 50 and 30 particles for this problem, respectively. Although the standard PSO uses more particles which is supposed to maintain better coverage of the search space and higher level of exploration, its convergence curve shows that the convergence is almost attained within the first 1,000 analyses and after that the convergence curve becomes straight. On the other hand democratic PSO reaches an initial convergence after about 1,500 analyses and it still keeps exploring the search space until it reaches the final result at 3,000th analysis. This can be interpreted as the modifications being effective on the alleviation of the premature convergence problem. It should be noted that the structure found by DPSO at 2,000th analysis is much lighter than that found by PSO at the same analysis. In fact while the modifications improve the exploration capabilities of the algorithm, they do not disturb the algorithm's convergence task.

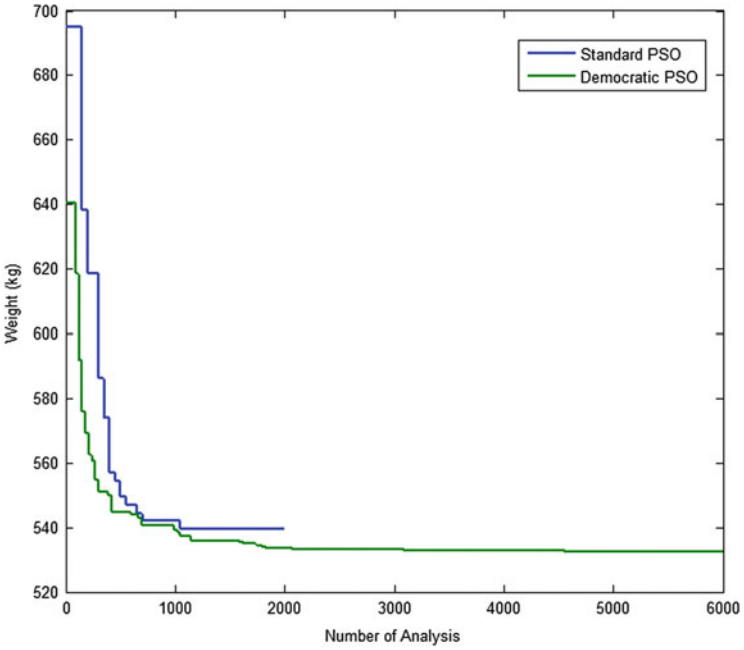

Fig. 2.5 Comparison of convergence curves of democratic and standard PSO algorithms recorded in the 10-bar truss problem

2.5.3.2 A Simply Supported 37-Bar Planar Truss

A simply supported 37-bar Pratt type truss, as depicted in Fig. 2.6, is examined as the second example.

The elements of the lower chord are modeled as bar elements with constant rectangular cross-sectional areas of 4×10^{-3} m^2. The other members are modeled as bar elements. These members which form the sizing variables of the problem are grouped with respect to symmetry. Also, the y-coordinate of all the nodes on the upper chord can vary in a symmetrical manner to form the layout variables. On the lower chord, a non-structural mass of 10 kg is attached to all free nodes. The first three natural frequencies of the structure are considered as the constraints. So this is an optimization on layout and size with nineteen design variables (14 sizing variables + five layout variables) and three frequency constraints. This example has been studied by Wang et al. [63] using an evolutionary node shift method and Lingyun et al. [64] using a niche hybrid genetic algorithm. Gomes [60] has investigated this problem using the standard particle swarm algorithm. Kaveh and Zolghadr [65] used the standard and an enhanced CSS to optimize the structure.

Material properties, frequency constrains and added masses are listed in Table 2.4.

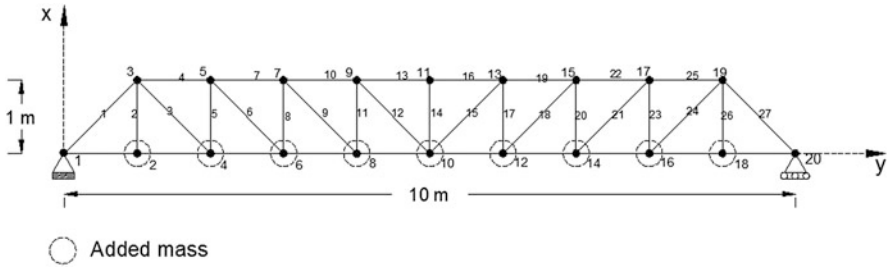

Fig. 2.6 Schematic of the simply-supported planar 37-bar truss

Table 2.4 Material properties and frequency constraints for the simply supported planar 37-bar truss

Property [unit]	Value
E (Modulus of elasticity) [N/m^2]	2.1×10^{11}
ρ (Material density) [kg/m^3]	7,800
Design variable lower bound [m^2]	1×10^{-4}
Design variable upper bound [m^2]	10×10^{-4}
Added mass [kg]	10
Constraints on first three frequencies [Hz]	$\omega 1 \geq 20$, $\omega 2 \geq 40$, $\omega 3 \geq 60$

Final cross-sectional areas and node coordinates obtained by different methods together with the corresponding weight are presented in Table 2.5. It can be seen that the proposed algorithm has found the best results so far. Specifically, in comparison to the standard PSO, the resulted structure is meaningfully lighter.

The mean weight and the standard deviation of the results obtained by DPSO are 362.21 kg and 1.68 kg respectively, while PSO has obtained a mean weight of 381.2 kg and a standard deviation of 4.26 kg. This indicates that DPSO not only finds a better best solution but also is more stable.

Table 2.6 represents the natural frequencies of the final structures obtained by various methods for the 37-bar simply supported planar truss.

Figure 2.7 shows the optimized layout of the simply-supported 37-bar truss as found by DPSO. The convergence curves for the democratic PSO and the standard PSO are shown in Fig. 2.6. The information on the convergence curve values at the few first analyses is not available in [60] (Fig. 2.8).

2.5.3.3 A 52-Bar Dome-Like Truss

Simultaneous layout and size optimization of a 52-bar dome-like truss is considered as the third example. Initial layout of the structure is depicted in Fig. 2.9. Non-structural masses of 50 kg are attached to all free nodes.

Table 2.7 summarized the material properties, frequency constraints and variable bounds for this example.

Table 2.5 Optimized designs obtained for the planar 37-bar truss problem

Variable	Wang et al. [63]	Lingyun et al. [64]	Gomes [60]	Kaveh and Zolghadr [65] Standard CSS	Democratic PSO [28]
Y3 , Y19 (m)	1.2086	1.1998	0.9637	0.8726	0.9482
Y5 , Y17 (m)	1.5788	1.6553	1.3978	1.2129	1.3439
Y7 , Y15 (m)	1.6719	1.9652	1.5929	1.3826	1.5043
Y9 , Y13 (m)	1.7703	2.0737	1.8812	1.4706	1.6350
Y11 (m)	1.8502	2.3050	2.0856	1.5683	1.7182
A1, A27 (cm^2)	3.2508	2.8932	2.6797	2.9082	2.6208
A2, A26 (cm^2)	1.2364	1.1201	1.1568	1.0212	1.0397
A3, A24 (cm^2)	1.0000	1.0000	2.3476	1.0363	1.0464
A4, A25 (cm^2)	2.5386	1.8655	1.7182	3.9147	2.7163
A5, A23 (cm^2)	1.3714	1.5962	1.2751	1.0025	1.0252
A6, A21 (cm^2)	1.3681	1.2642	1.4819	1.2167	1.5081
A7, A22 (cm^2)	2.4290	1.8254	4.6850	2.7146	2.3750
A8, A20 (cm^2)	1.6522	2.0009	1.1246	1.2663	1.4498
A9, A18 (cm^2)	1.8257	1.9526	2.1214	1.8006	1.4499
A10, A19 (cm^2)	2.3022	1.9705	3.8600	4.0274	2.5327
A11, A17 (cm^2)	1.3103	1.8294	2.9817	1.3364	1.2358
A12, A15 (cm^2)	1.4067	1.2358	1.2021	1.0548	1.3528
A13, A16 (cm^2)	2.1896	1.4049	1.2563	2.8116	2.9144
A14 (cm^2)	1.0000	1.0000	3.3276	1.1702	1.0085
Weight (kg)	366.50	368.84	377.20	362.84	360.40

Table 2.6 Natural frequencies (Hz) evaluated at the optimized designs for the planar 37-bar truss

Frequency number	Wang et al. [63]	Lingyun et al. [64]	Gomes [60]	Kaveh and Zolghadr [65] Standard CSS	Democratic PSO [28]
1	20.0850	20.0013	20.0001	20.0000	20.0194
2	42.0743	40.0305	40.0003	40.0693	40.0113
3	62.9383	60.0000	60.0001	60.6982	60.0082
4	74.4539	73.0444	73.0440	75.7339	76.9896
5	90.0576	89.8244	89.8240	97.6137	97.2222

All of the elements of the structure are categorized in 8 groups according to Table 2.8. All free nodes are permitted to move ± 2 m from their initial position in a symmetrical manner. This is a configuration optimization problem with thirteen variables (eight sizing variables + five layout variables) and two frequency constraints.

This example has been investigated by Lin et al. [67] using a mathematical programming technique and Lingyun et al. [64] using a niche hybrid genetic algorithm. Gomes [60] has analyzed this problem using the standard particle swarm algorithm. This problem has been studied using the standard and an

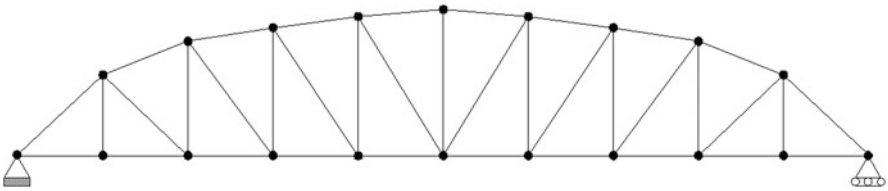

Fig. 2.7 Schematic of the optimized layout of the simply-supported planar 37-bar truss

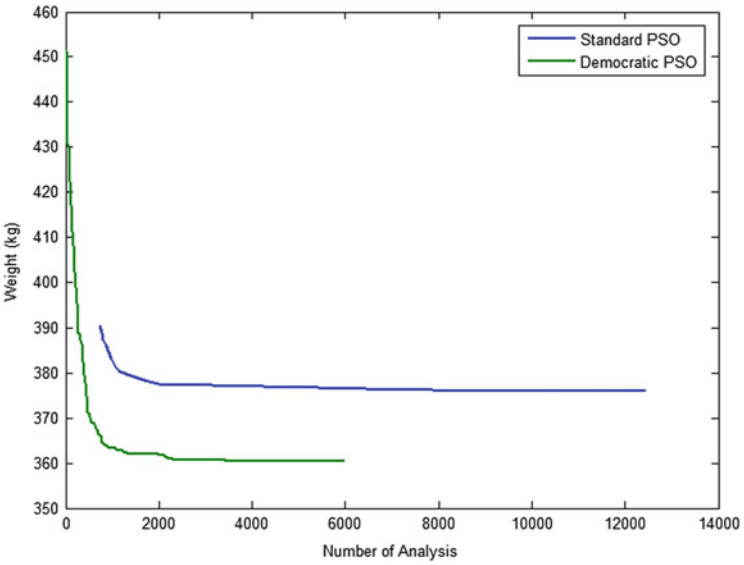

Fig. 2.8 Comparison of convergence curves of democratic and standard PSO algorithms recorded in the 37-bar Pratt type planar truss

enhanced CSS [65] and a hybridized CSS-BBBC with a trap recognition capability [66].

Table 2.9 compares the final cross-sectional areas and node coordinates found by different methods together with the corresponding weight for the 52 bar space truss.

It can be seen that the result gained by the democratic PSO is far better than the standard PSO. The standard PSO uses 70 particles and about 160 iterations (11,200 analyses) to reach its best result while the democratic PSO uses 30 particles and 200 iterations (6,000 analyses). Table 2.8 indicates that among all the methods listed above the democratic PSO has obtained the best solution. The mean weight and the standard deviation of the results gained by DPSO are 198.71 kg and 13.85 kg, respectively while PSO has obtained a mean weight of 234.3 kg and a standard deviation of 5.22 kg. DPSO performs considerably better in terms of best and mean weight.

Table 2.10 shows the natural frequencies of the final structures found by various methods for the 52-bar dome-like space truss.

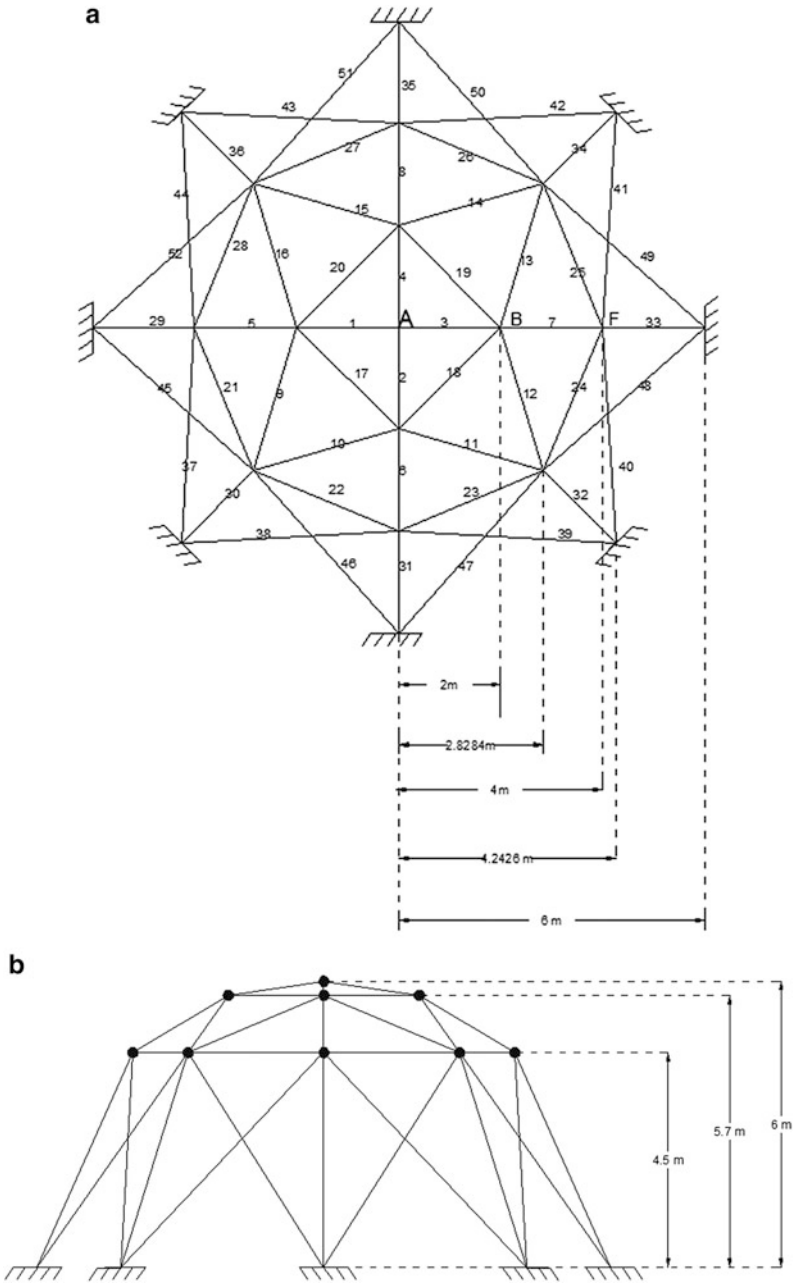

Fig. 2.9 Schematic of the initial layout of the spatial 52-bar truss. (**a**) Top view and (**b**) Side view

Table 2.7 Material properties and frequency constraints and variable bounds for the spatial 52-bar truss

Property [unit]	Value
E (Modulus of elasticity) [N/m^2]	2.1×10^{11}
ρ (Material density) [kg/m^3]	7,800
Added mass [kg]	50
Allowable range for cross-sections [m^2]	$0.0001 \leq A \leq 0.001$
Constraints on first three frequencies [Hz]	$\omega_1 \leq 15.916$, $\omega_2 \geq 28.648$

Table 2.8 Element grouping adopted in the spatial 52-bar truss problem

Group number	Elements
1	1–4
2	5–8
3	9–16
4	17–20
5	21–28
6	29–36
7	37–44
8	45–52

Table 2.9 Optimized designs obtained for the spatial 52-bar truss problem

				Kaveh and Zolghadr [65]	
Variable	Liu et al. [67]	Lingyun et al. [64]	Gomes [60]	Standard CSS	Democratic PSO [28]
Z_A (m)	4.3201	5.8851	5.5344	5.2716	6.1123
X_B (m)	1.3153	1.7623	2.0885	1.5909	2.2343
Z_B (m)	4.1740	4.4091	3.9283	3.7093	3.8321
X_F (m)	2.9169	3.4406	4.0255	3.5595	4.0316
Z_F (m)	3.2676	3.1874	2.4575	2.5757	2.5036
A1 (cm^2)	1.00	1.0000	0.3696	1.0464	1.0001
A2 (cm^2)	1.33	2.1417	4.1912	1.7295	1.1397
A3 (cm^2)	1.58	1.4858	1.5123	1.6507	1.2263
A4 (cm^2)	1.00	1.4018	1.5620	1.5059	1.3335
A5 (cm^2)	1.71	1.911	1.9154	1.7210	1.4161
A6 (cm^2)	1.54	1.0109	1.1315	1.0020	1.0001
A7 (cm^2)	2.65	1.4693	1.8233	1.7415	1.5750
A8 (cm^2)	2.87	2.1411	1.0904	1.2555	1.4357
Weight (kg)	298.0	236.046	228.381	205.237	195.351

Figure 2.10 shows the optimized layout of the spatial 52-bar truss as found by DPSO. The convergence curve of the best run of the democratic PSO for the 52-bar dome-like truss is shown in Fig. 2.11. The convergence curve for the standard PSO is not available in [60].

Table 2.10 Natural frequencies (Hz) evaluated at the optimized designs for the spatial 52-bar truss

Frequency number	Liu et al. [67]	Lingyun et al. [64]	Gomes [60]	Kaveh and Zolghadr [65]	Democratic PSO [28]
				Standard CSS	
1	15.22	12.81	12.751	9.246	11.315
2	29.28	28.65	28.649	28.648	28.648
3	29.28	28.65	28.649	28.699	28.648
4	31.68	29.54	28.803	28.735	28.650
5	33.15	30.24	29.230	29.223	28.688

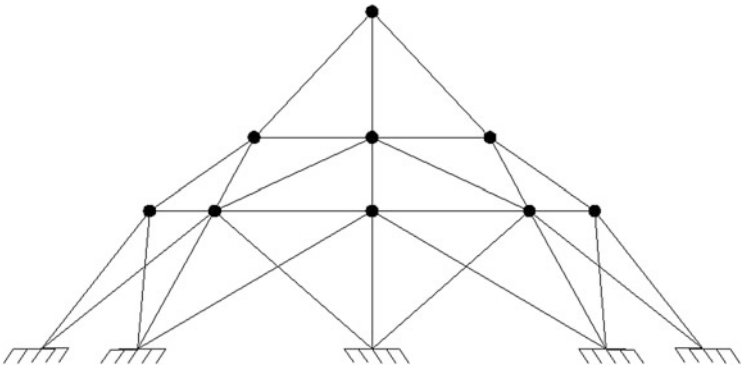

Fig. 2.10 Schematic of the optimized layout of the spatial 52-bar truss

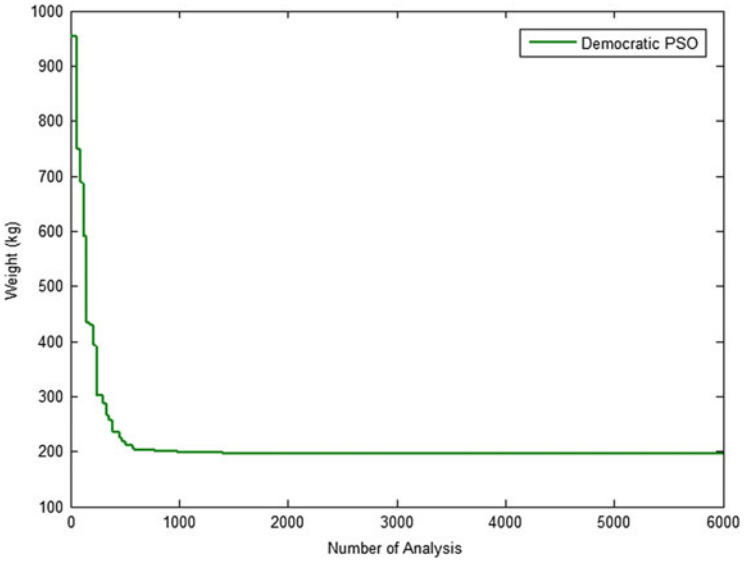

Fig. 2.11 Convergence curve of the democratic PSO for the spatial 52-bar truss

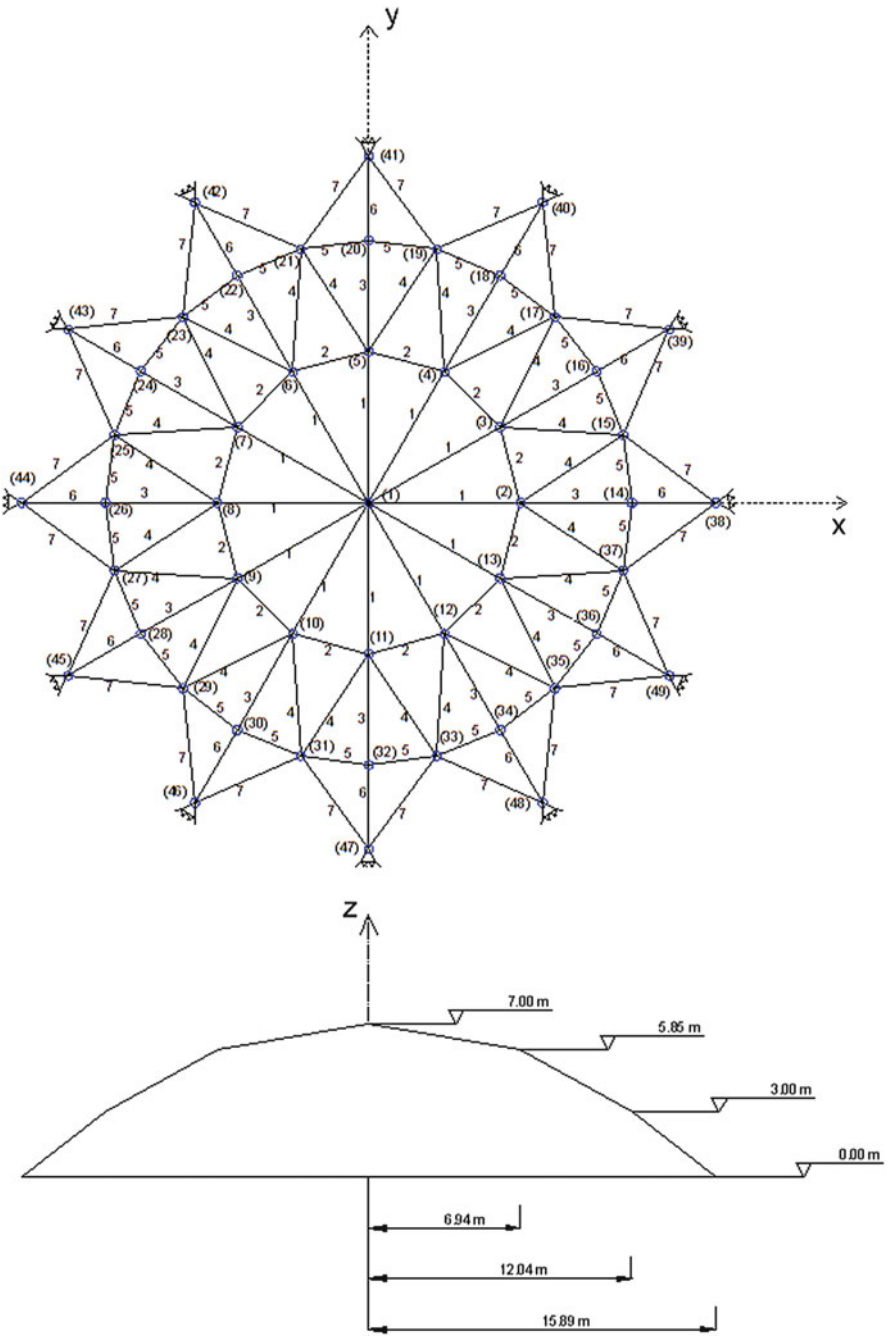

Fig. 2.12 Schematic of the 120-bar

Table 2.11 Material properties and frequency constraints and variable bounds for the 120-bar dome truss

Property [unit]	Value
E (Modulus of elasticity) [N/m^2]	2.1×10^{11}
ρ (Material density) [kg/m^3]	7971.810
Added mass [kg]	$m_1 = 3{,}000$, $m_1 = 500$, $m_2 = 100$
Allowable range for cross-sections [m^2]	$0.0001 \leq A \leq 0.01293$
Constraints on first three frequencies [Hz]	$\omega_1 \geq 9$, $\omega_2 \geq 11$

Table 2.12 Optimized designs (cm^2) obtained for the 120-bar dome truss

Element group	Standard PSO	Democratic PSO
1	23.494	19.607
2	32.976	41.290
3	11.492	11.136
4	24.839	21.025
5	9.964	10.060
6	12.039	12.758
7	14.249	15.414
Weight (kg)	9,171.93	8,890.48

Table 2.13 Natural frequencies (Hz) evaluated at the optimized designs for the 120-bar dome truss

Frequency number	Standard PSO	Democratic PSO
1	9.0000	9.0001
2	11.0000	11.0007
3	11.0052	11.0053
4	11.0134	11.0129
5	11.0428	11.0471

2.5.3.4 A 120-Bar Dome Truss

The 120-bar dome truss shown in Fig. 2.12 is considered as the last example. This problem has been previously studied as a benchmark optimization problem with static constraints.

This problem has been used as a size optimization problem with frequency constraints in [65]. Non-structural masses are attached to all free nodes as follows: 3,000 kg at node one, 500 kg at nodes 2 through 13 and 100 kg at the rest of the nodes. Material properties, frequency constraints and variable bounds for this example are summarized in Table 2.11. The layout of the structure is kept unchanged during the optimization process. Hence, this is a sizing optimization problem.

This example is solved here using both the standard and democratic PSO in order to make the comparison possible. 30 particles and 200 iterations are used for both methods. Table 2.12 represents a comparison between the final results obtained by

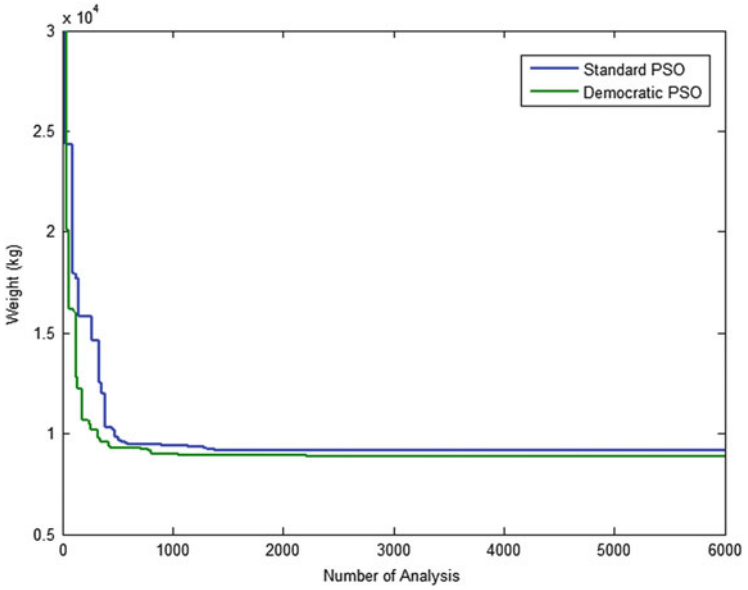

Fig. 2.13 Comparison of converge curves of democratic and standard PSO algorithms recorded in the 120-bar dome problem

the standard and the democratic PSO. Table 2.13 shows the natural frequencies of the final structures found by both methods.

According to Table 2.12, the result obtained by the democratic PSO is meaningfully lighter than that of the standard PSO. The mean weight and the standard deviation of the results gained by DPSO are 8,895.99 kg and 4.26 kg, respectively while PSO has obtained a mean weight of 9,251.84 kg and a standard deviation of 89.38 kg. This shows that the Democratic PSO outperforms the standard version in all of the above mentioned aspects. Fig. 2.13 shows the convergence curves for both methods.

References

1. Kennedy J, Eberhart R (1995) Particle swarm optimization. In: Proceedings of IEEE international conference on neural networks, vol 4, pp 1942–1948
2. Shi Y, Eberhart R (1998) A modified particle swarm optimizer. In: Proceedings of IEEE World congress on computational intelligence. In: The 1998 I.E. international conference on evolutionary computation, pp 69–73
3. Reeves WT (1983) Particle systems – a technique for modeling a class of fuzzy objects. ACM Trans Graph 2(2):91–108
4. Renolds CW (1987) Flocks, herds, and schools: a distributed behavioral model. Comput Graph 21(4):25–34 (Proc SIGGRAPH '87)

5. Millonas MM (1993) Swarms, phase transitions, and collective intelligence. In: Langton CG (ed) Proceedings of ALIFE III. Addison-Wesley, Santa Fe Institute
6. Heppner F, Grenander U (1990) A stochastic nonlinear model for coordinated bird flocks. In: Krasner S (ed) The ubiquity of chaos. AAAS Publications, Washington, DC
7. Eberhart RC, Simpson P, Dobbins R (1996) Computational intelligence PC tools, Chapter 6. AP Professional, San Diego, CA, pp 212–226
8. Shi Y, Eberhart RC (1998) A modified particle swarm optimizer. In: Proceedings of the congress on evolutionary computation, pp 73–79
9. Eberhart RC, Shi Y (2000) Comparing inertia weights and constriction factors in particle swarm optimization. In: Proceedings of IEEE congress evolutionary computation, San Diego, CA, pp 84–88
10. Clerc M (1999) The swarm and the queen: towards a deterministic and adaptive particle swarm optimization. In: Proceedings I999 ICEC, Washington, DC, 1951–1957
11. Bui LT, Soliman O, Abass HS (2007) A modified strategy for the constriction factor in particle swarm optimization. In: Randall M, Abass HS, Wiles J (eds) Lecture Notes in Artificial Intelligence 4828, pp 333–344
12. Kennedy J (2006) Swarm intelligence. In Handbook of Nature-Inspired and Innovative Computing, Springer, 187-219
13. Talbi EG (2009) Metaheuristics: from design to implementation. Wiley, UK
14. Shi Y, Eberhart RC (1998) Parameter selection in particle swarm optimization. In: Proceedings of evolutionary programming VII (EP98), pp 591–600
15. Carlisle A, Dozier G (2001) An off-the-shelf PSO. In: Proceedings of workshop on particle swarm optimization, Indianapolis, IN
16. Trelea IC (2003) The particle swarm optimization algorithm: convergence analysis and parameter selection. Inform Proc Lett 85:317–325
17. Zhang L, Yu H, Hu S (2005) Optimal choice of parameters for particle swarm optimization. J Zhejiang Univ Sci 6A(6):528–534
18. Pedersen MEH (2010) Good parameters for particle swarm optimization. Hvass Laboratories Technical Report HL1001
19. Bansal JC, Singh PK, Saraswat M, Verma A, Jadon SS, Abraham A (2011) Inertia weight strategies in particle swarm optimization. In: IEEE 3rd world congress on nature and biologically inspired computing (NaBIC 2011), Salamanca, Spain, pp 640–647
20. Wang Y, Li B, Weise T, Wang J, Yuan B, Tian Q (2011) Self-adaptive learning based particle swarm optimization. Inform Sci 181(20):4515–4538
21. Angeline PJ (1998) Evolutionary optimization versus particle swarm optimization: philosophy and performance difference. In: Proceedings of 7th annual conference on evolutionary programming, p 601
22. Zhao Y, Zub W, Zeng H (2009) A modified particle swarm optimization via particle visual modeling analysis. Comput Math Appl 57:2022–2029
23. van den Bergh F, Engelbrecht AP (2002) A new locally convergent particle swarm optimizer. In: Proceedings of IEEE conference on systems, man and cybernetics, Hammamet, Tunisia
24. Krink T, Vestertroem JS, Riget J (2002) Particle swarm optimization with spatial particle extension. Proceedings of the IEEE congress on evolutionary computation (CEC 2002), Honolulu, Hawaii
25. Riget J, Vesterstrøm JS (2002) A diversity-guided particle swarm optimizer–the ARPSO. EVALife technical report no 2002–2002
26. Silva A, Neves A, Costa E (2002) An empirical comparison of particle swarm and predator prey optimization. In: Proceedings of the 13th Irish international conference on artificial intelligence and cognitive science, vol 2464, pp 103–110
27. Jie J, Zeng J, Han CZ (2006) Adaptive particle swarm optimization with feedback control of diversity. In: Proceedings of the 2006 international conference on computational intelligence and bioinformatics (ICIC'06), vol Part III, pp 81–92

28. Kaveh A, Zolghadr A (2013) A democratic PSO for truss layout and size optimization with frequency constraints. Comput Struct 42(3):10–21
29. Mendes R, Kennedy J, Neves J (2004) The fully informed particle swarm: simpler, maybe better. IEEE Trans Evolut Comput 8(3):204–210
30. Matsushita H, Nishio Y (2009) Network-structured particle swarm optimizer with various topology and its behaviors. Advances in self-organizing maps. Lecture Notes in Computer Science 5629:163–171
31. Monson CK, Seppi KD (2005) Exposing origin-seeking bias in PSO. In: Proceedings of the conference on genetic and evolutionary computation (GECCO'05), Washington DC, USA, pp 241–248
32. Angeline PJ (1998) Using selection to improve particle swarm optimization. In: Proceedings of the IEEE congress on evolutionary computation (CEC 1998), Anchorage, Alaska, USA
33. Gehlhaar DK, Fogel DB (1996) Tuning evolutionary programming for conformationally flexible molecular docking. In: Evolutionary Programming, pp 419–429
34. Suganthan PN, Hansen N, Liang JJ, Deb K, Chen YP, Auger A, Tiwari S (2005) Problem definitions and evaluation criteria for the CEC 2005 Special Session on Real Parameter Optimization. Nanyang Technological University, Singapore
35. Clerc M (2006) Particle swarm optimization. Antony Rowe, Chippenham, Wiltshire
36. Wilke DN, Kok S, Groenwold AA (2007) Comparison of linear and classical velocity update rules in particle swarm optimization: notes on scale and frame invariance. Int J Numer Methods Eng 70:985–1008
37. Talbi E-G (2002) A taxonomy of hybrid metaheuristics. J Heuristics 8:541–564
38. Banks A, Vincent J, Anyakoha C (2008) A review of particle swarm optimization. Part II: Hybridisation, combinatorial, multicriteria and constrained optimization, and indicative applications. Nat Comput 7(1):109–124
39. Černý V (1985) Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. J Optim Theory Appl 45:41–51
40. Locatelli M (1996) Convergence properties of simulated annealing for continuous global optimization. J Appl Probability 33:1127–1140
41. Shieh HL, Kuo CC, Chiang CM (2011) Modified particle swarm optimization algorithm with simulated annealing behavior and its numerical verification. Appl Math Comput 218:4365–4383
42. Glover F (1989) Tabu Search - Part 1. ORSA J Comput 1(2):190–206
43. Glover F (1990) Tabu Search - Part 2. ORSA J Comput 2(1):4–32
44. Shen Q, Shi WM, Kong W (2008) Hybrid particle swarm optimization and tabu search approach for selecting genes for tumor classification using gene expression data. Comput Bio Chemist 32:53–60
45. Løvberg M, Rasmussen TK, Krink T (2001) Hybrid particle swarm optimiser with breeding and subpopulations. In: Proceedings of the genetic and evolutionary computation conference, pp 469–476
46. Krink T, Løvbjerg M (2002) The lifecycle model: combining particle swarm optimization, genetic algorithms and hillclimbers. In: Proceedings of parallel problem solving from nature VII (PPSN 2002). Lecture Notes in Computer Science (LNCS) 2439: 621–630
47. Kaveh A, Talatahari S (2009) Particle swarm optimizer, ant colony strategy and harmony search scheme hybridized for optimization of truss structures. Comput Struct 87(56):267–283
48. Dorigo M (1992) Optimization, learning and natural algorithms (in Italian), PhD Thesis. Dipartimento di Elettronica, Politecnico di Milano, IT
49. Geem ZW, Kim J-H, Loganathan GV (2001) A new heuristic optimization algorithm: harmony search. Simulation 76(2):60–68
50. Higashi N, Iba H (2003) Particle swarm optimization with Gaussian mutation. In: Proceedings of the IEEE swarm intelligence symposium 2003 (SIS 2003), Indianapolis, IN, USA, pp 72–79
51. Juang C-F (2004) A hybrid of genetic algorithm and particle swarm optimization for recurrent network design. IEEE Trans Syst Man Cybern – Part B: Cybern 34(2):997–1006

52. Kaveh A, Talatahari S (2011) Hybrid charged system search and particle swarm optimization for engineering design problems. Eng Comput 28(4):423–440
53. Kaveh A, Talatahari S (2010) A novel heuristic optimization method: charged system search. Acta Mech 213(3–4):267–289
54. Liu H, Abraham A (2005) Fuzzy adaptive turbulent particle swarm optimization. In: Proceedings of 5th international conference on hybrid intelligent systems (HIS'05), Rio de Janeiro, Brazil, 6–9 November
55. Zahara E, Kao YT (2009) Hybrid Nelder-Mead simplex search and particle swarm optimization for constrained engineering design problems. Expert Syst Appl 36:3880–3886
56. Qian X, Cao M, Su Z, Chen J (2012) A hybrid particle swarm optimization (PSO)-simplex algorithm for damage identification of delaminated beams. Math Prob Eng:11 (Article ID 607418)
57. Kaveh A, Talatahari S (2007) A discrete particle swarm ant colony optimization for design of steel frames. Asian J Civil Eng 9(6):563–575
58. Kennedy J, Eberhart RC (1997) A discrete binary version of the particle swarm algorithm. In: Proceedings of the conference on systems, man and cybernetics, Piscataway, New Jersey, pp 4104–4109
59. Chen WN, Zhang J, Chung HSH, Zhong WL, Wu WG, Shi Y (2010) A novel set-based particle swarm optimization method for discrete optimization problems. IEEE Trans Evol Comput 14 (2):278–300
60. Gomes MH (2011) Truss optimization with dynamic constraints using a particle swarm algorithm. Expert Syst Appl 38:957–968
61. Grandhi RV, Venkayya VB (1988) Structural optimization with frequency constraints. AIAA J 26:858–866
62. Sedaghati R, Suleman A, Tabarrok B (2002) Structural optimization with frequency constraints using finite element force method. AIAA J 40:382–388
63. Wang D, Zha WH, Jiang JS (2004) Truss optimization on shape and sizing with frequency constraints. AIAA J 42:1452–1456
64. Lingyun W, Mei Z, Guangming W, Guang M (2005) Truss optimization on shape and sizing with frequency constraints based on genetic algorithm. J Comput Mech 25:361–368
65. Kaveh A, Zolghadr A (2011) Shape and size optimization of truss structures with frequency constraints using enhanced charged system search algorithm. Asian J Civil Eng 12:487–509
66. Kaveh A, Zolghadr A (2012) Truss optimization with natural frequency constraints using a hybridized CSS-BBBC algorithm with trap recognition capability. Comput Struct 102–103:14–27
67. Lin JH, Chen WY, Yu YS (1982) Structural optimization on geometrical configuration and element sizing with static and dynamic constraints. Comput Struct 15:507–515

Chapter 3
Charged System Search Algorithm

3.1 Introduction

This chapter consists of two parts. In the first part an optimization algorithm based on some principles from physics and mechanics, which is known as the Charged System Search (CSS) [1]. In this algorithm the governing Coulomb law from electrostatics and the Newtonian laws of mechanics. CSS is a multi-agent approach in which each agent is a Charged Particle (CP). CPs can affect each other based on their fitness values and their separation distances. The quantity of the resultant force is determined by using the electrostatics laws and the quality of the movement is determined using Newtonian mechanics laws. CSS can be utilized in all optimization fields; especially it is suitable for non-smooth or non-convex domains. CSS needs neither the gradient information nor the continuity of the search space.

In the second part, CSS is applied to optimal design of skeletal structures and high performance of CSS is illustrated [2].

3.2 Charged System Search

3.2.1 Background

3.2.1.1 Electrical Laws

In physics, the space surrounding an electric charge creates an electric field, which exerts a force on other electrically charged objects. The electric field surrounding a point charge is given by Coulomb's law. Coulomb confirmed that the electric force between two small charged spheres is proportional to the inverse square of their separation distance. The electric force between charged spheres A and B in Fig. 3.1 causes the spheres to either attract or repel each other, and the resulting motion causes the suspended fiber to twist. Since the restoring torque of the twisted fiber is

A. Kaveh, *Advances in Metaheuristic Algorithms for Optimal Design of Structures*,
DOI 10.1007/978-3-319-05549-7_3, © Springer International Publishing Switzerland 2014

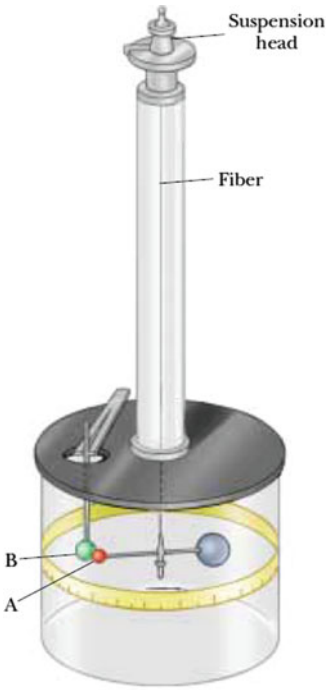

Fig. 3.1 Coulomb's torsion balance, used to establish the inverse-square law for the electric force between two charges [1]

proportional to the angle through which the fiber rotates, a measurement of this angle provides a quantitative measure of the electric force of attraction or repulsion [3]. Coulomb's experiments showed that the electric force between two stationary charged particles:

- is inversely proportional to the square of the separation distance between the particles and directed along the line joining them;
- is proportional to the product of the charges q_i and q_j on the two particles;
- is attractive if the charges are of opposite sign, and repulsive if the charges have the same sign.

From the above observations, Coulomb's law provides the magnitude of the electric force (Coulomb force) between the two point-charges [3] as

$$F_{ij} = k_e \frac{q_i q_j}{r_{ij}^2} \tag{3.1}$$

where k_e is a constant called the Coulomb constant; r_{ij} is the distance between the two charges.

Consider an insulating solid sphere of radius a, which has a uniform volume charge density and carries a total positive charge q_i. The electric field E_{ij} at a point outside the sphere is defined as

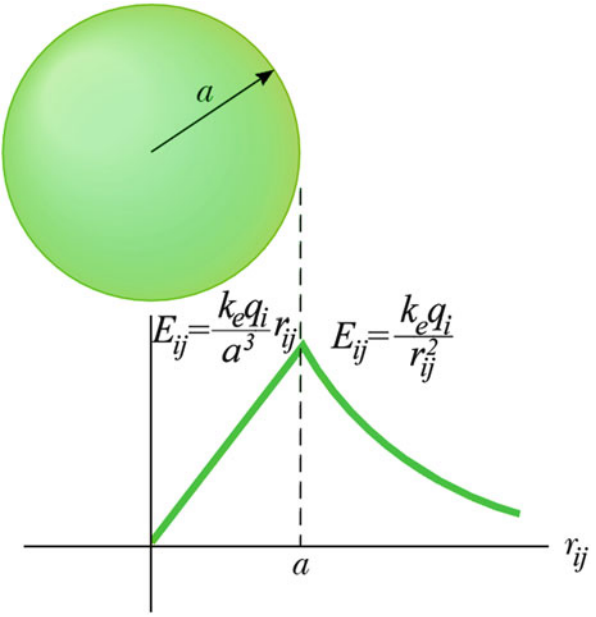

Fig. 3.2 A plot of E_{ij} versus r_{ij} for a uniformly charged insulating sphere [1]

$$E_{ij} = k_e \frac{q_i}{r_{ij}^2} \tag{3.2}$$

The magnitude of the electric field at a point inside the sphere can be obtained using Gauss's law. This is expressed as

$$E_{ij} = k_e \frac{q_i}{a^3} r_{ij} \tag{3.3}$$

Note that this result shows that $E_{ij} \to 0$ as $r_{ij} \to 0$. Therefore, the result eliminates the problem that would exist at $r_{ij} = 0$ if E_{ij} is varied as $1/r_{ij}^2$ inside the sphere as it does outside the sphere. That is, if $E_{ij} \propto 1/r_{ij}^2$ the field will be infinite at $r_{ij} = 0$, which is physically impossible. Hence, the electric field inside the sphere varies linearly with r_{ij}. The field outside the sphere is the same as that of a point charge q_i located at $r_{ij} = 0$. Also the magnitudes of the electric fields for a point at inside or outside the sphere coincide when $r_{ij} = a$. A plot of E_{ij} versus r_{ij} is shown in Fig. 3.2 [3].

In order to calculate the equivalent electric field at a point ($\mathbf{r}_j$) due to a group of point charges, the superposition principle is applied to fields which follows directly from the superposition of the electric forces. Thus, the electric field of a group of charges can be expressed as

$$E_j = \sum_{i=1, i\neq j}^{N} E_{ij} \tag{3.4}$$

where N is the total number of charged particles and E_{ij} is equal to

$$E_{ij} = \begin{cases} \dfrac{k_e q_i}{a^3} r_{ij} & \text{if } r_{ij} < a \\ \dfrac{k_e q_i}{r_{ij}^2} & \text{if } r_{ij} \geq a \end{cases} \tag{3.5}$$

In order to obtain both the magnitude and direction of the resultant force on a charge q_j at position $\mathbf{r}_j$ due to the electric field of a charge q_i at position $\mathbf{r}_i$, the full vector form is required which can be expressed as

$$\mathbf{F}_{ij} = E_{ij} q_j \frac{\mathbf{r}_i - \mathbf{r}_j}{\left|\left|\mathbf{r}_i - \mathbf{r}_j\right|\right|} \tag{3.6}$$

For multiple charged particles, this can be summarized as follows:

$$\mathbf{F}_j = k_e q_j \sum_{i, i\neq j} \left(\frac{q_i}{a^3} r_{ij} \cdot i_1 + \frac{q_i}{r_{ij}^2} \cdot i_2 \right) \frac{\mathbf{r}_i - \mathbf{r}_j}{\left|\left|\mathbf{r}_i - \mathbf{r}_j\right|\right|} \quad \left\langle \begin{matrix} i_1 = 1, i_2 = 0 \Leftrightarrow r_{ij} < a \\ i_1 = 0, i_2 = 1 \Leftrightarrow r_{ij} \geq a \end{matrix} \right. \tag{3.7}$$

3.2.1.2 Newtonian Mechanics Laws

Newtonian mechanics or classical mechanics studies the motion of objects. In the study of motion, the moving object is described as a particle regardless of its size. In general, a particle is a point-like mass having infinitesimal size. The motion of a particle is completely known if the particle's position in space is known at all times. The displacement of a particle is defined as its change in position. As it moves from an initial position $\mathbf{r}_{old}$ to a final position $\mathbf{r}_{new}$, its displacement is given by

$$\Delta\mathbf{r} = \mathbf{r}_{new} - \mathbf{r}_{old} \tag{3.8}$$

The slope of tangent line of the particle position represents the velocity of this particle as

$$\mathbf{v} = \frac{\mathbf{r}_{new} - \mathbf{r}_{old}}{t_{new} - t_{old}} = \frac{\mathbf{r}_{new} - \mathbf{r}_{old}}{\Delta t} \tag{3.9}$$

When the velocity of a particle changes with time, the particle is said to be accelerated. The acceleration of the particle is defined as the change in the velocity divided by the time interval during which that change has occurred:

$$\mathbf{a} = \frac{\mathbf{v}_{new} - \mathbf{v}_{old}}{\Delta t} \tag{3.10}$$

Using (3.8), (3.9), and (3.10), the displacement of any object as a function of time is obtained approximately as

$$\mathbf{r}_{new} = \frac{1}{2}\mathbf{a} \cdot \Delta t^2 + \mathbf{v}_{old} \cdot \Delta t + \mathbf{r}_{old} \tag{3.11}$$

Another law utilized in this article is Newton's second law which explains the question of what happens to an object that has a nonzero resultant force acting on it: the acceleration of an object is directly proportional to the net force acting on it and inversely proportional to its mass

$$\mathbf{F} = m \cdot \mathbf{a} \tag{3.12}$$

where m is the mass of the object.

Substituting (3.12) in (3.11), we have

$$\mathbf{r}_{new} = \frac{1}{2}\frac{\mathbf{F}}{m} \cdot \Delta t^2 + \mathbf{v}_{old} \cdot \Delta t + \mathbf{r}_{old} \tag{3.13}$$

3.2.2 *Presentation of Charged Search System*

In this section, a new efficient optimization algorithm is established utilizing the aforementioned physics laws, which is called Charged System Search (CSS). In the CSS, each solution candidate $\mathbf{X}_i$ containing a number of decision variables (i.e. $\mathbf{X}_i = \{x_{i,j}\}$) is considered as a charged particle. The charged particle is affected by the electrical fields of the other agents. The quantity of the resultant force is determined by using the electrostatics laws as discussed in Sect. 3.2.1.1 and the quality of the movement is determined using the Newtonian mechanics laws. It seems that an agent with good results must exert a stronger force than the bad ones, so the amount of the charge will be defined considering the objective function value, *fiti*). In order to introduce CSS, the following rules are developed:

Rule 1 Many of the natural evolution algorithms maintain a population of solutions which are evolved through random alterations and selection [4,5]. Similarly, CSS considers a number of Charged Particles (CP). Each CP has a magnitude of charge (q_i) and as a result creates an electrical field around its space. The magnitude of the charge is defined considering the quality of its solution, as follows

$$q_i = \frac{fit(i) - fitworst}{fitbest - fitworst}, \quad i = 1, 2, \ldots, N \tag{3.14}$$

where *fitbest* and *fitworst* are the so far best and the worst fitness of all particles;

fit(*i*) represents the objective function value or the fitness of the agent *i*; and *N* is the total number of CPs. The separation distance r_{ij} between two charged particles is defined as follows:

$$r_{ij} = \frac{\left|\left|\mathbf{X}_i - \mathbf{X}_j\right|\right|}{\left|\left|\left(\mathbf{X}_i + \mathbf{X}_j\right)/2 - \mathbf{X}_{best}\right|\right| + \varepsilon} \tag{3.15}$$

where $\mathbf{X}_i$ and $\mathbf{X}_j$ are the positions of the *i*th and *j*th CPs, $\mathbf{X}_{best}$ is the position of the best current CP, and ε is a small positive number to avoid singularities.

Rule 2 The initial positions of CPs are determined randomly in the search space

$$x_{i,j}^{(o)} = x_{i,\min} + rand \cdot (x_{i,\max} - x_{i,\min}), \quad i = 1, 2, \ldots, n \tag{3.16}$$

where $x_{i,j}^{(o)}$ determines the initial value of the *i*th variable for the *j*th CP; $x_{i,\min}$ and $x_{i,\max}$ are the minimum and the maximum allowable values for the *i*th variable; *rand* is a random number in the interval [0,1]; and *n* is the number of variables. The initial velocities of charged particles are zero

$$v_{i,j}^{(o)} = 0, \quad i = 1, 2, \ldots, n \tag{3.17}$$

Rule 3 Three conditions could be considered related to the kind of the attractive forces:

- Any CP can affect another one; i.e. a bad CP can affect a good one and vice versa ($p_{ij} = 1$).
- A CP can attract another if its electric charge amount (fitness with revise relation in minimizing problems) is better than other. In other words, a good CP attracts a bad CP

$$p_{ij} = \begin{cases} 1 & fit(j) > fit(i) \\ 0 & \text{else} \end{cases} \tag{3.18}$$

- All good CPs can attract bad CPs and only some of bad agents attract good agents, considering following probability function

$$p_{ij} = \begin{cases} 1 & \dfrac{fit(i) - fitbest}{fit(j) - fit(i)} > rand \vee fit(j) > fit(i) \\ 0 & \text{else} \end{cases} \tag{3.19}$$

According to the above conditions, when a good agent attracts a bad one, the exploitation ability for the algorithm is provided, and vice versa if a bad CP attracts a good CP, the exploration is provided. When a CP moves toward a good agent it improves its performance, and so the self-adaptation principle is guaranteed. Moving a good CP toward a bad one may cause losing the previous good solution or at least increasing the computational cost to find a good solution. To resolve this problem, a memory which saves the best so far solution can be considered. Therefore, it seems that the third kind of the above conditions is the best rule because of providing strong exploration ability and an efficient exploitation.

Rule 4 The value of the resultant electrical force acting on a CP is determined using (3.7) as

$$\mathbf{F}_j = q_j \sum_{i, i \neq j} \left(\frac{q_i}{a^3} r_{ij} \cdot i_1 + \frac{q_i}{r_{ij}^2} \cdot i_2 \right) p_{ij} (\mathbf{X}_i - \mathbf{X}_j), \quad \begin{cases} j = 1, 2, \ldots, N \\ i_1 = 1, i_2 = 0 \Leftrightarrow r_{ij} < a \\ i_1 = 0, i_2 = 1 \Leftrightarrow r_{ij} \geq a \end{cases} \tag{3.20}$$

where $\mathbf{F}_j$ is the resultant force acting on the jth CP, as illustrated in Fig. 3.3.

In this algorithm, each CP is considered as a charged sphere with radius a, which has a uniform volume charge density. In this paper, the magnitude of a is set to unity; however for more complex examples, the appropriate value for a must be defined considering the size of the search space. One can utilize the following equation as a general formula

$$a = 0.10 \times \max\left(\left\{ x_{i,\max} - x_{i,\min} \middle| i = 1, 2, \ldots, n \right\} \right) \tag{3.21}$$

According to this rule, in the first iteration where the agents are far from each other the magnitude of the resultant force acting on a CP is inversely proportional to the square of the separation between the particles. Thus the exploration power in this condition is high because of performing more searches in the early iterations. It is necessary to increase the exploitation of the algorithm and to decrease the exploration gradually. After a number of searches where CPs are collected in a small space and the separation between the CPs becomes small say 0.1, then the resultant force becomes proportional to the separation distance of the particles instead of being inversely proportional to the square of the separation distance. According to Fig. 3.4, if the first equation ($F_{ij} \propto 1/r_{ij}^2$) is used for $r_{ij} = 0.1$, we have $F_{ij} = 100 \times k_e q_i q_j$ that is a large value, compared to a force acting on a CP at $r_{ij} = 2$ ($F_{ij} = 0.25 \times k_e q_i q_j$), and this great force causes particles to get farther from each other instead of getting nearer, while the second one ($F_{ij} \propto r_{ij}$)

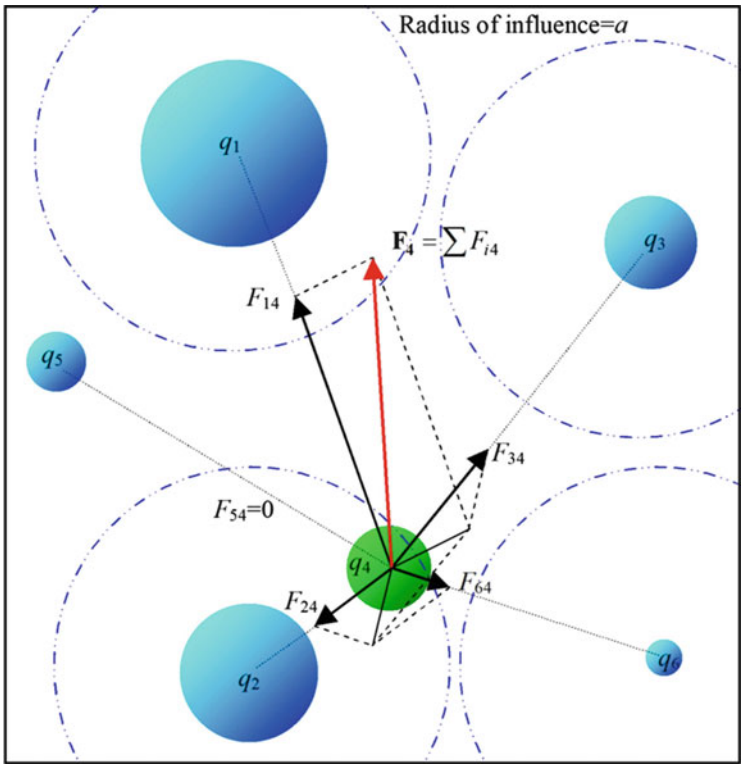

Fig. 3.3 Determining the resultant electrical force acting on a CP [1]

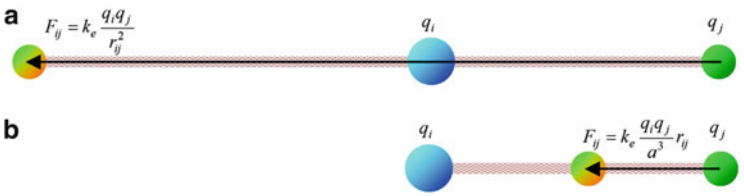

Fig. 3.4 A comparison between the equation [1] (**a**) $F_{ij} \propto 1/r_{ij}^2$ and (**b**) $F_{ij} \propto r_{ij}$ when $r_{ij} < a$

guaranties that a convergence will happen. Therefore, the parameter a separates the global search phase and the local search phase, i.e. when majority of the agents are collected in a space with radius a, the global search is finished and the optimizing process is continued by improving the previous results, and thus the local search starts. Besides, using these principles controls the balance between the exploration and the exploitation.

It should be noted that this rule considers the competition step of the algorithm. Since the resultant force is proportional to the magnitude of the charge, a better fitness (great q_i) can create a bigger attract force, so the tendency to move toward a good CP becomes more than a bad particle.

Rule 5 The new position and velocity of each CP is determined considering (3.9) and (3.13), as follows

$$\mathbf{X}_{j,new} = rand_{j1} \cdot k_a \cdot \frac{\mathbf{F}_j}{m_j} \cdot \Delta t^2 + rand_{j2} \cdot k_v \cdot \mathbf{V}_{j,old} \cdot \Delta t + \mathbf{X}_{j,old} \tag{3.22}$$

$$\mathbf{V}_{j,new} = \frac{\mathbf{X}_{j,new} - \mathbf{X}_{j,old}}{\Delta t} \tag{3.23}$$

where k_a is the acceleration coefficient; k_v is the velocity coefficient to control the influence of the previous velocity; and $rand_{j1}$ and $rand_{j2}$ are two random numbers uniformly distributed in the range of (0,1). Here, m_j is the mass of the CPs which is equal to q_j. Δt is the time step and is set to unity.

The effect of the pervious velocity and the resultant force acting on a CP can be decreased or increased based on the values of the k_v and k_a, respectively. Excessive search in the early iterations may improve the exploration ability; however it must be deceased gradually, as described before. Since k_a is the parameter related to the attracting forces, selecting a large value for this parameter may cause a fast convergence and vice versa a small value can increase the computational time. In fact k_a is a control parameter of the exploitation. Therefore, choosing an incremental function can improve the performance of the algorithm. Also, the direction of the pervious velocity of a CP is not necessarily the same as the resultant force. Thus, it can be concluded that the velocity coefficient k_v controls the exploration process and therefore a decreasing function can be selected. Thus, k_v and k_a are defined as

$$k_v = 0.5(1 - iter/iter_{\max}), \quad k_a = 0.5(1 + iter/iter_{\max}) \tag{3.24}$$

where $iter$ is the actual iteration number and $iter_{\max}$ is the maximum number of iterations. With this equation, k_v decreases linearly to zero while k_a increases to one when the number of iterations rises. In this way, the balance between the exploration and the fast rate of convergence is saved. Considering the values of these parameters, (3.22) and (3.23) can be rewritten as

$$\begin{aligned}\mathbf{X}_{j,new} = {} & 0.5 rand_{j1} \cdot (1 + iter/iter_{\max}) \cdot \sum_{i,i \neq j} \left(\frac{q_i}{a^3} r_{ij} \cdot i_1 + \frac{q_i}{r_{ij}^2} \cdot i_2 \right) p_{ij} (\mathbf{X}_i - \mathbf{X}_j) \\ & + 0.5 rand_{j2} \cdot (1 + iter/iter_{\max}) \cdot \mathbf{V}_{j,old} + \mathbf{X}_{j,old}\end{aligned} \tag{3.25}$$

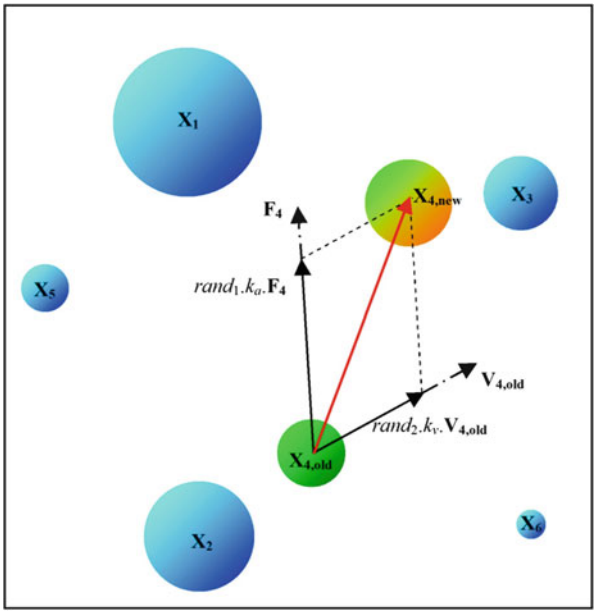

Fig. 3.5 The movement of a CP to the new position [1]

$$\mathbf{V}_{j,new} = \mathbf{X}_{j,new} - \mathbf{X}_{j,old} \tag{3.26}$$

Figure 3.5 illustrates the motion of a CP to its new position using this rule. The rules 5 and 6 provide the cooperation step of the CPs, where agents collaborate with each other by information transferring.

Rule 6 Considering a memory which saves the best CP vectors and their related objective function values can improve the algorithm performance without increasing the computational cost. To fulfill this aim, Charged Memory (CM) is utilized to save a number of the best so far solutions. In this paper, the size of the CM (i.e. *CMS*) is taken as $N/4$. Another benefit of the CM consists of utilizing this memory to guide the current CPs. In other words, the vectors stored in the CM can attract current CPs according to (3.20). Instead, it is assumed that the same number of the current worst particles cannot attract the others.

Rule 7 There are two major problems in relation to many metaheuristic algorithms; the first problem is the balance between exploration and exploitation in the beginning, during, and at the end of the search, and second is how to deal with an agent violating the limits of the variables.

The first problem is solved naturally through the application of above-stated rules; however, in order to solve the second problem, one of the simplest approaches is utilizing the nearest limit values for the violated variable. Alternatively, one can force the violating particle to return to its previous position or one can reduce the maximum value of the velocity to allow fewer particles to violate the variable boundaries. Although these approaches are simple, they are not sufficiently efficient

and may lead to reduced exploration of the search space. This problem has previously been addressed and solved using the harmony search-based handling approach [4,6]. According to this mechanism, any component of the solution vector violating the variable boundaries can be regenerated from the CM as

$$x_{i,j} = \begin{cases} \text{w.p. CMCR} ==> \text{select a new value for a variable from CM} \\ \qquad\qquad ==> \text{w.p. } (1 - \text{PAR}) \text{ do nothing} \\ \qquad\qquad ==> \text{w.p. PAR choose a neighboring value} \\ \text{w.p. } (1 - \text{CMCR}) ==> \text{select a new value randomly} \end{cases} \tag{3.27}$$

where “w.p.” is the abbreviation for "with the probability"; $x_{i,j}$ is the ith component of the CP j; The CMCR (the Charged Memory Considering Rate) varying between 0 and 1 sets the rate of choosing a value in the new vector from the historic values stored in the CM, and (1−CMCR) sets the rate of randomly choosing one value from the possible range of values. The pitch adjusting process is performed only after a value is chosen from CM. The value (1−PAR) sets the rate of doing nothing. For more details, the reader may refer to [4,6].

Rule 8 The terminating criterion is one of the followings:

- Maximum number of iterations: the optimization process is terminated after a fixed number of iterations, for example, 1,000 iterations.
- Number of iterations without improvement: the optimization process is terminated after some fixed number of iterations without any improvement.
- Minimum objective function error: the difference between the values of the best objective function and the global optimum is less than a pre-fixed anticipated threshold.
- Difference between the best and the worst CPs: the optimization process is stopped if the difference between the objective values of the best and the worst CPs becomes less than a specified accuracy.
- Maximum distance of CPs: the maximum distance between CPs is less than a pre-fixed value.

Now we can establish a new optimization algorithm utilizing the above rules. The following pseudo-code summarizes the CSS algorithm:

Level 1: Initialization

- **Step 1:** *Initialization*. Initialize CSS algorithm parameters; Initialize an array of Charged Particles with random positions and their associated velocities (Rules 1 and 2).
- **Step 2:** *CP ranking*. Evaluate the values of the fitness function for the CPs, compare with each other and sort increasingly.

- **Step 3:** *CM creation*. Store *CMS* number of the first CPs and their related values of the objective function in the CM.

Level 2: Search

- **Step 1:** *Attracting forces determination*. Determine the probability of moving each CP toward others (Rule 3), and calculate the attracting force vector for each CP (Rule 4).
- **Step 2:** *Solution construction*. Move each CP to the new position and find the velocities (Rule 5).
- **Step 3:** *CP position correction*. If each CP exits from the allowable search space, correct its position using Rule 7.
- **Step 4:** *CP ranking*. Evaluate and compare the values of the objective function for the new CPs; and sort them increasingly.
- **Step 5:** *CM updating*. If some new CP vectors are better than the worst ones in the CM, include the better vectors in the CM and exclude the worst ones from the CM (Rule 6).

Level 3: Terminating criterion controlling

- Repeat search level steps until a terminating criterion is satisfied (Rule 8).

The flowchart of the CSS algorithm is illustrated in Fig. 3.6.

3.3 Validation of CSS

In order to verify the efficiency of the new algorithm, some numerical examples are considered from literature. The examples contain 18 uni-modal and multi-modal functions. These numerical examples are presented in Sect. 3.3.1. The performance of the CSS to optimize these functions is investigated in Sect. 3.3.2. In Sect. 3.3.3, some well-studied engineering design problems taken from the optimization literature are used to illustrate the way in which the proposed method works.

3.3.1 Description of the Examples

In this section a number of benchmark functions chosen from [7] are optimized using CSS and compared to GA and some of its variations to verify the efficiency of CSS. The description of these test problems is provided in Table 3.1. When the dimension is selected as 2, a perspective view and the related contour lines for some of these functions are illustrated in Fig. 3.7.

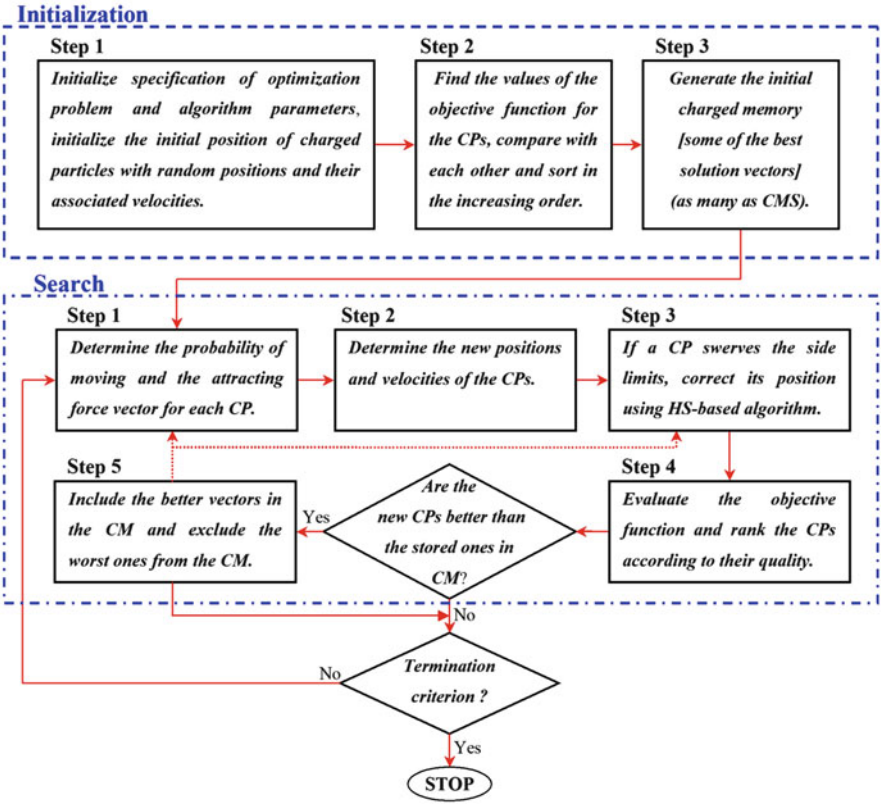

Fig. 3.6 The flowchart of the CSS [1]

3.3.2 Results

Similar to the other met-heuristics, for the CSS a large value for the number of CPs increases the search strength of the algorithm as well as the computational cost and vice versa a small number causes a quick convergence without performing a complete search. Here, the number of CPs is set to 20 and the maximum number of the permitted iterations is considered as 200. These values seem to be suitable for finding the optimum results. The value of HMCR is set to 0.95 and that of PAR is taken as 0.10 [4]. The results obtained by CSS are listed in Table 3.2 along with those obtained by GA and some of its variations, which are directly derived from [7]. The numbers denote the average number of function evaluations from 50 independent runs for every objective function described in Sect. 3.1. The numbers in parentheses represent the fraction of successful runs in which the algorithm has located the global minimum with predefined accuracy, which is taken as $\varepsilon = f_{min} - f_{final} = 10^{-4}$. The absence of the parentheses denotes that the algorithm has been successful in all independent runs. Although the GEN-S-M-LS finds good results in

Table 3.1 Specifications of the benchmark problems

Function name	Interval	Function	Global minimum
Aluffi-Pentiny	$\mathbf{X} \in [-10, 10]^2$	$f(\mathbf{X}) = \frac{1}{4}x_1^4 - \frac{1}{2}x_1^2 + \frac{1}{10}x_1 + \frac{1}{2}x_2^2$	−0.352386
Bohachevsky 1	$\mathbf{X} \in [-100, 100]^2$	$f(\mathbf{X}) = x_1^2 + 2x_2^2 - \frac{3}{10}\cos(3\pi x_1) - \frac{4}{10}\cos(4\pi x_2) + \frac{7}{10}$	0.0
Bohachevsky 2	$\mathbf{X} \in [-50, 50]^2$	$f(\mathbf{X}) = x_1^2 + 2x_2^2 - \frac{3}{10}\cos(3\pi x_1)\cos(4\pi x_2) + \frac{3}{10}$	0.0
Becker and Lago	$\mathbf{X} \in [-10, 10]^2$	$f(\mathbf{X}) = (\lvert x_1\rvert - 5)^2 + (\lvert x_2\rvert - 5)^2$	0.0
Branin	$0 \le x_2 \le 15 - 5 \le x_1 \le 10$	$f(\mathbf{X}) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos(x_1) + 10$	0.397887
Camel	$\mathbf{X} \in [-5, 5]^2$	$f(\mathbf{X}) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	−1.0316
Cb3	$\mathbf{X} \in [-5, 5]^2$	$f(\mathbf{X}) = 2x_1^2 - 1.05x_1^5 + \frac{1}{6}x_1^6 + x_1x_2 + x_2^2$	0.0
Cosine mixture	$n = 4, \mathbf{X} \in [-1, 1]^n$	$f(\mathbf{X}) = \sum_{i=1}^{n} x_i^2 - \frac{1}{10}\sum_{i=1}^{n}\cos(5\pi x_i)$	−0.4
DeJoung	$\mathbf{X} \in [-5.12, 5.12]^3$	$f(\mathbf{X}) = x_1^2 + x_2^2 + x_3^2$	0.0
Exponential	$n = 2, 4, 8, \mathbf{X} \in [-1, 1]^n$	$f(\mathbf{X}) = -\exp\left(-0.5\sum_{i=1}^{n} x_i^2\right)$	−1
Goldstein and price	$\mathbf{X} \in [-2, 2]^2$	$f(\mathbf{X}) = \left[1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - {}^1 14x_2 + 6x_1x_2 + 3x_2^2\right] \times \left[30 + (2x_1 - 3x_2)^2(18 - 32x_1 - 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2\right]^2$	3.0
Griewank	$\mathbf{X} \in [-100, 100]^2$	$f(\mathbf{X}) = 1 + \frac{1}{200}\sum_{i=1}^{2} x_i^2 - \prod_{i=1}^{2}\cos\left(\frac{x_i}{\sqrt{i}}\right)$	0.0
Hartman 3	$\mathbf{X} \in [0, 1]^3$	$f(\mathbf{X}) = -\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{3} a_{ij}(x_j - p_{ij})^2\right)$ $a = \begin{bmatrix} 3 & 10 & 30 \\ 0.1 & 10 & 35 \\ 3 & 10 & 30 \\ 0.1 & 10 & 35 \end{bmatrix}$, $c = \begin{bmatrix} 1 \\ 1.2 \\ 3 \\ 3.2 \end{bmatrix}$ and	−3.862782

Hartman 6	$\mathbf{X} \in [0, 1]^6$	$p = \begin{bmatrix} 0.3689 & 0.117 & 0.2673 \\ 0.4699 & 0.4387 & 0.747 \\ 0.1091 & 0.8732 & 0.5547 \\ 0.03815 & 0.5743 & 0.8828 \end{bmatrix}$ $a = \begin{bmatrix} 10 & 3 & 17 & 3.5 & 1.7 & 8 \\ 0.05 & 10 & 17 & 0.1 & 8 & 14 \\ 3 & 3.5 & 17 & 10 & 17 & 8 \\ 17 & 8 & 0.05 & 10 & 0.1 & 14 \end{bmatrix}$, $c = \begin{bmatrix} 1 \\ 1.2 \\ 3 \\ 3.2 \end{bmatrix}$ and $p = \begin{bmatrix} 0.1312 & 0.1696 & 0.5569 & 0.0124 & 0.8283 & 0.5886 \\ 0.2329 & 0.4135 & 0.8307 & 0.3736 & 0.1004 & 0.9991 \\ 0.2348 & 0.1451 & 0.3522 & 0.2883 & 0.3047 & 0.6650 \\ 0.4047 & 0.8828 & 0.8732 & 0.5743 & 0.1091 & 0.0381 \end{bmatrix}$	-3.322368
Rastrigin	$\mathbf{X} \in [-1, 1]^2$	$f(\mathbf{X}) = \sum_{i=1}^{2} \left(x_i^2 - \cos(18x_i)\right)$	-2.0
Rosenbrock	$\mathbf{X} \in [-30, 30]^n$, $n = 2$	$f(\mathbf{X}) = \sum_{i=1}^{n-1} 100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2$	0.0

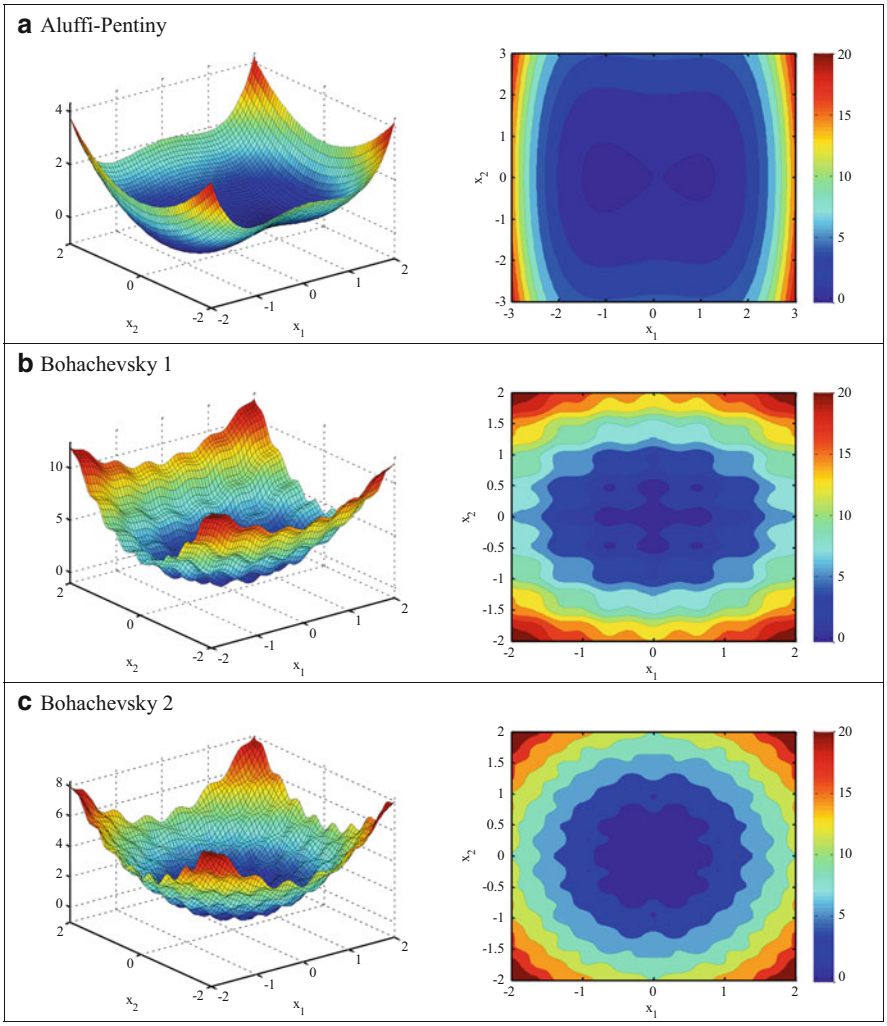

Fig. 3.7 (continued)

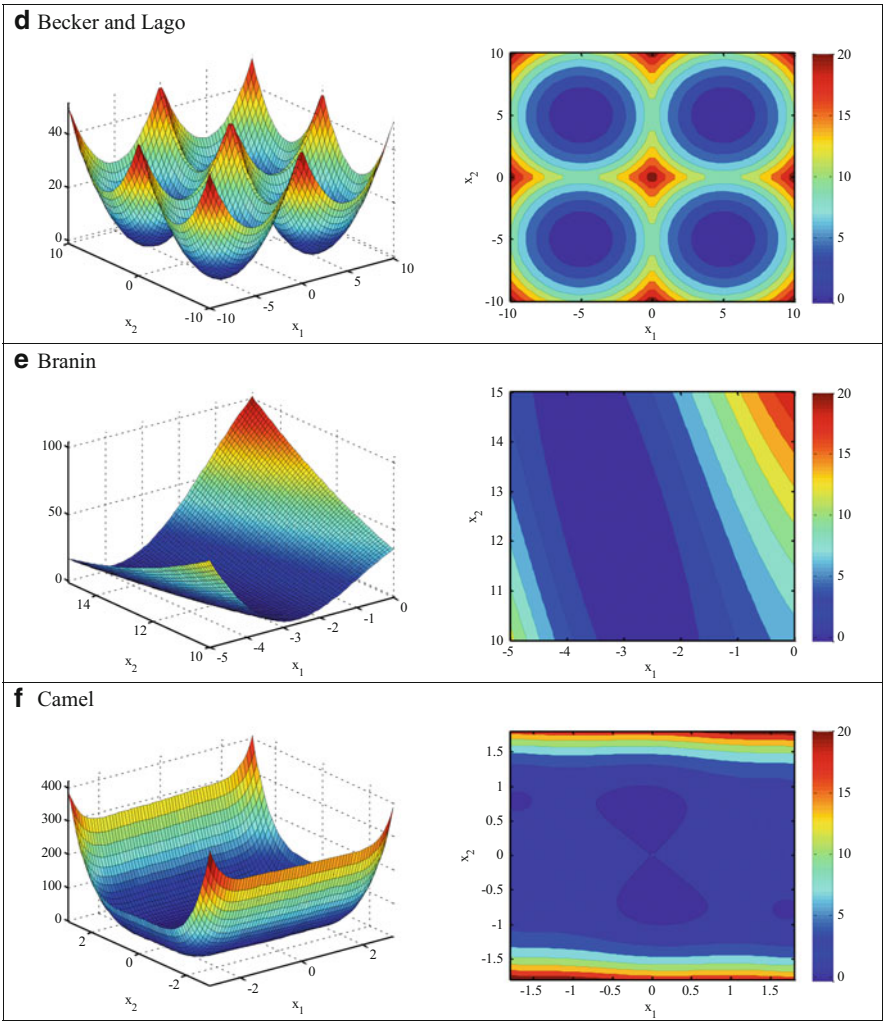

Fig. 3.7 (continued)

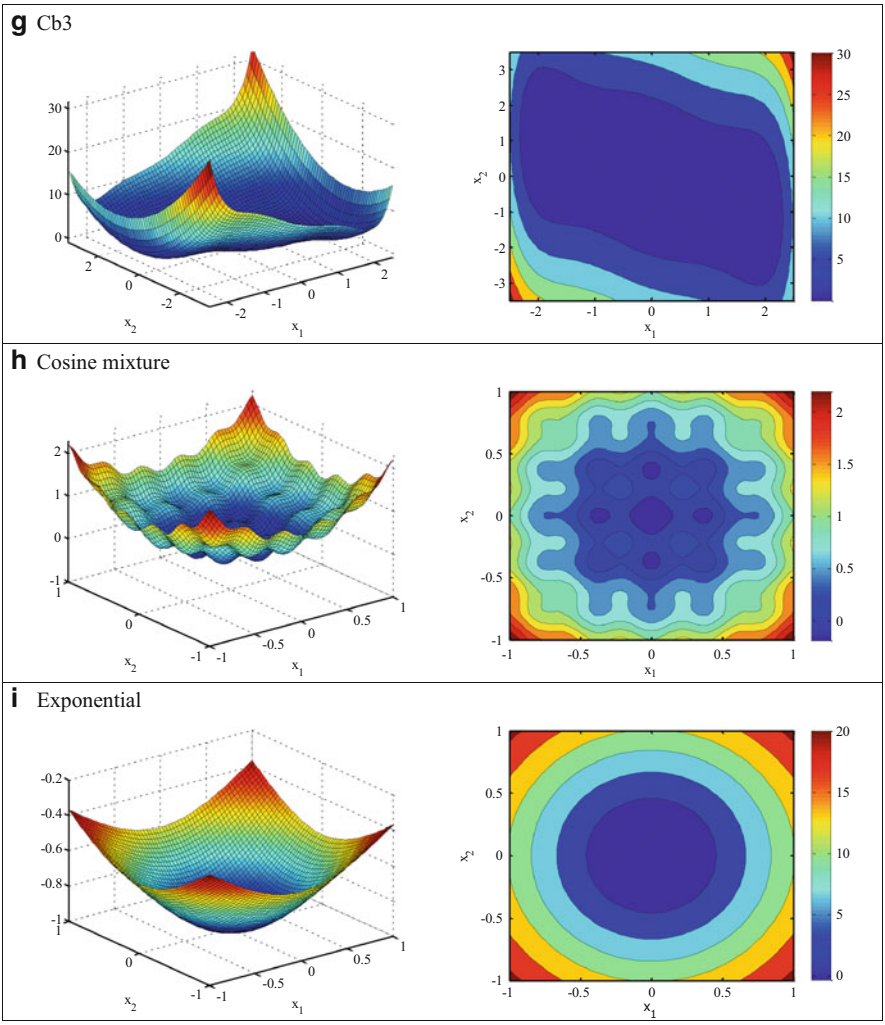

Fig. 3.7 (continued)

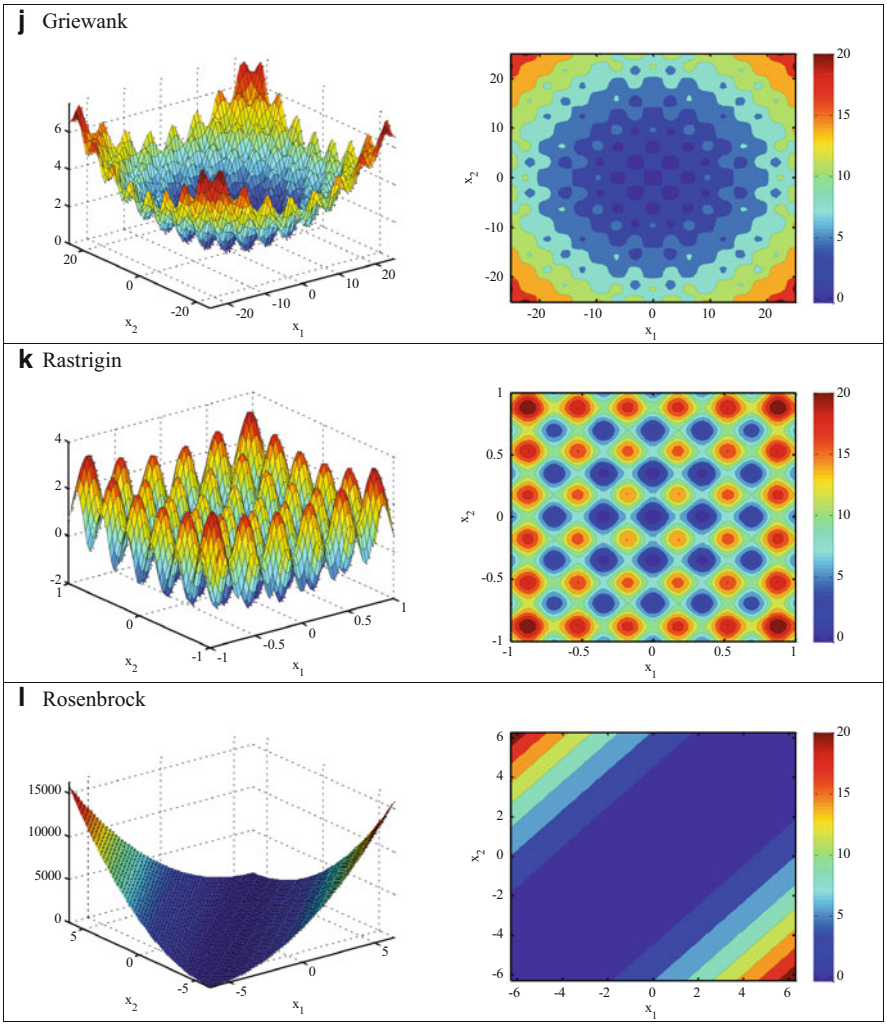

Fig. 3.7 A perspective view and the related contour lines for some of function when $n = 2$ [1]

some cases, it must be noted that GEN-S-M-LS utilizes some auxiliary mechanisms such as an improved stopping rule, a new mutation mechanism, a repeated application of a local search procedure. To sum up, comparison of the results demonstrates that CSS has a faster convergence than original GA and its variations.

In order to have some general idea about the way the CSS works, Fig. 3.8 is prepared to show the positions of the current CPs and the stored CPs in the CM for the first example. It can be seen that in the first iterations, the CPs investigate the entire search space to discover a favorite space (global search). When this favorite

Table 3.2 Performance comparison for the benchmark problems

Function	GEN	GEN–S	GEN–S–M	GEN–S–M–LS	*CSS*
AP	1,360 (0.99)	1,360	1,277	1,253	804
Bf1	3,992	3,356	1,640	1,615	1,187
Bf2	20,234	3,373	1,676	1,636	742
BL	19,596	2,412	2,439	1,436	423
Branin	1,442	1,418	1,404	1,257	852
Camel	1,358	1,358	1,336	1,300	575
Cb3	9,771	2,045	1,163	1,118	436
CM	2,105	2,105	1,743	1,539	1,563
Dejoung	9,900	3,040	1,462	1,281	630
Exp2	938	936	817	807	132
Exp4	3,237	3,237	2,054	1,496	867
Exp8	3,237	3,237	2,054	1,496	1,426
Goldstein and Price	1,478	1,478	1,408	1,325	682
Griewank	18,838 (0.91)	3,111 (0.91)	1,764	1,652 (0.99)	1,551
Hartman3	1,350	1,350	1,332	1,274	860
Hartman6	2,562 (0.54)	2,562 (0.54)	2,530 (0.67)	1,865 (0.68)	1,783
Rastrigin	1,533 (0.97)	1,523 (0.97)	1,392	1,381	1,402
Rosenbrock	9,380	3,739	1,675	1,462	1,452
Total	112,311 (96.72)	41,640 (96.77)	29,166 (98.16)	25,193 (98.16)	17,367

space containing a global optimum is discovered, the movements of the CPs are limited to this space in order to provide more exploitation (local search).

For many heuristic algorithms it is common property that if all the agents get gathered in a small space, i.e. if the agents are trapped in part of the search space, escaping from this may be very difficult. Since prevailing forces for the CSS algorithm are attracting forces, it looks as if the above problem has remained unsolved for this method. However, having a good balance between the exploration and the exploitations, and considering three steps containing self-adaptation, cooperation and competition in the CSS, can solve this problem. As illustrated in Fig. 3.9 which shows the positions of the CPs for the first example when all the initial agents are located in a small part of the space, CSS can escape from this space and go toward the favorite space.

3.4 Charged System Search for Structural Optimization

3.4.1 Statement of the Optimization Design Problem

For optimum design of structures the objective function can be expressed as

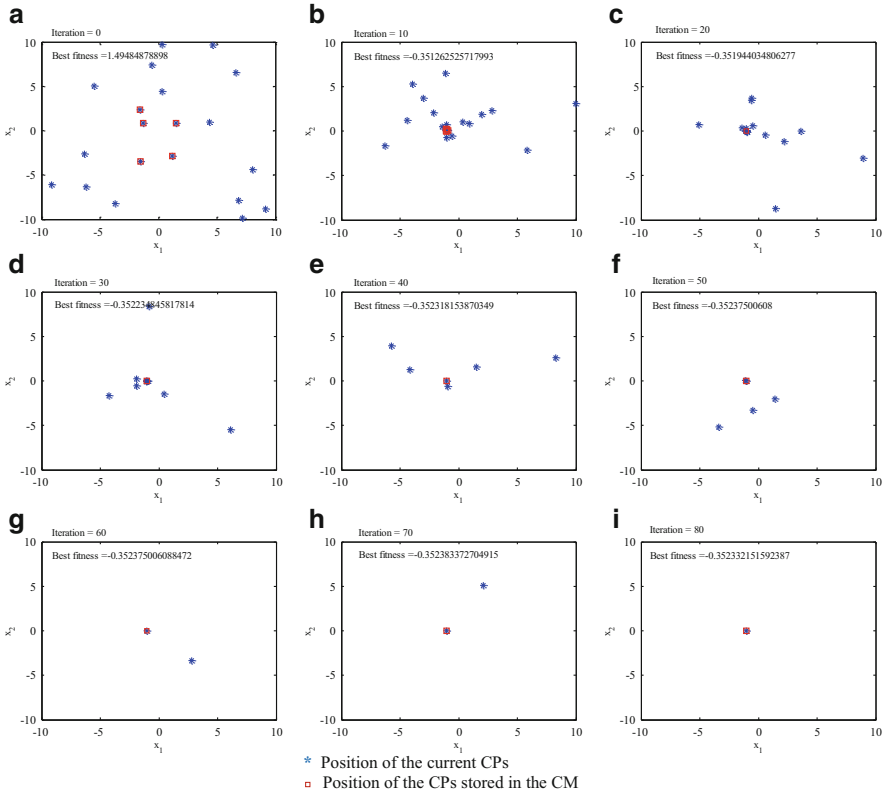

Fig. 3.8 The positions of the current CPs and the stored CPs in the CM for the first example [1]

$$\text{minimize} \qquad W(\mathbf{X}) = \sum_{i=1}^{n} \gamma_i \cdot x_i \cdot L_i \tag{3.28}$$

where $W(\mathbf{X})$ is the weight of the structure; n is the number of members making up the structure; γ_i represents the material density of member i; L_i is the length of member i; x_i is the cross-sectional area of member i chosen between $x_{\min}$ and $x_{\max}$; and *min* is the lower bound and *max* is the upper bound. This minimum design also has to satisfy inequality constraints that limit design variable sizes and structural responses, Lee and Geem [8].

3.4.1.1 Constraint Conditions for Truss Structures

For truss structures, the constraints are as follows:

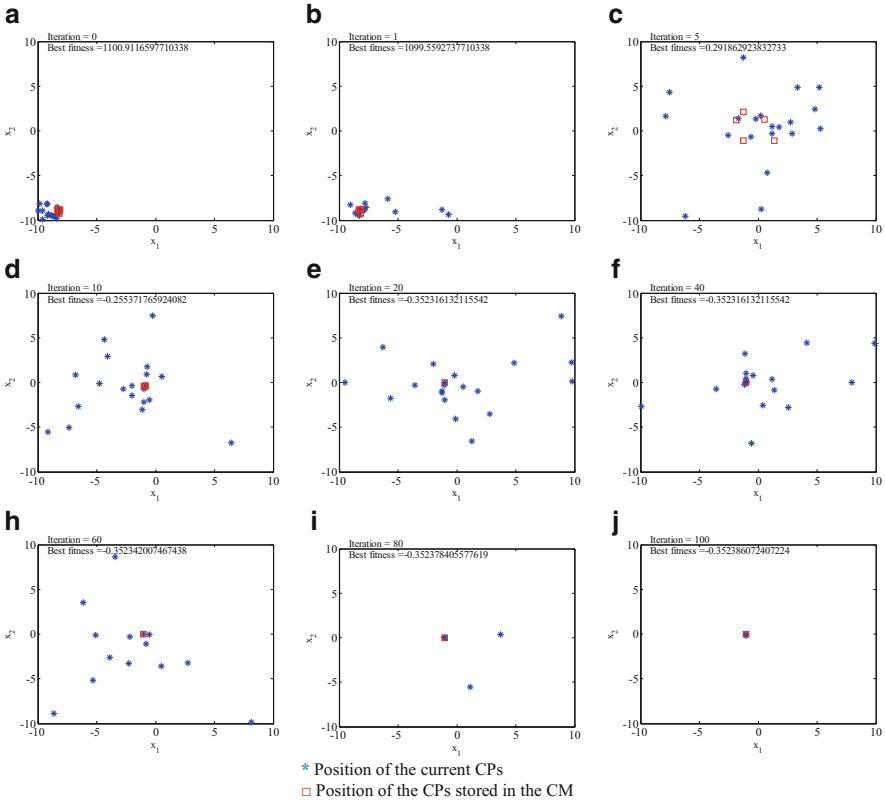

Fig. 3.9 The positions of the CPs for the first example when the all initial agents are introduced in a small part of the space [1]

$$\begin{aligned} &\delta_{\min} \le \delta_{\mathrm{i}} \le \delta_{\max} \qquad i = 1, 2, ..., m \\ &\sigma_{\min} \le \sigma_{\mathrm{i}} \le \sigma_{\max} \qquad i = 1, 2, ..., n \\ &\sigma_i^b \le \sigma_{\mathrm{i}} \le 0 \qquad\quad i = 1, 2, ..., nc \end{aligned} \tag{3.29}$$

in which m is the number of nodes; nc denotes the number of compression elements; σ_i and δ_i are the stress and nodal deflection, respectively; σ_i^b represents allowable buckling stress in member i when it is in compression.

3.4.1.2 Constraint Conditions for Frame Structures

For the frame structures, according to the AISC-ASD [9] code, the constraints are as follows:

The stress limitations:

$$\frac{f_a}{F_a}+\frac{f_{bx}}{F_{bx}}+\frac{f_{by}}{F_{by}}\le 1,\quad \text{For}\quad \frac{f_a}{F_a}\le 0.15 \tag{3.30}$$

$$\frac{f_a}{F_a}+\frac{C_{mx}f_{bx}}{\left(1-\frac{f_a}{F'_{ex}}\right)F_{bx}}+\frac{C_{my}f_{by}}{\left(1-\frac{f_a}{F'_{ey}}\right)F_{by}}\le 1,\quad \text{For}\quad \frac{f_a}{F_a}>0.15 \tag{3.31}$$

$$\frac{f_a}{0.6F_y}+\frac{f_{bx}}{F_{bx}}+\frac{f_{by}}{F_{by}}\le 1,\quad \text{For}\quad \frac{f_a}{F_a}>0.15 \tag{3.32}$$

The slenderness ratio limitation:

$$\begin{cases}\lambda_{\mathrm{i}}=\dfrac{k_iL_i}{r_i}\le 300\ \text{For tension members}\\ \lambda_{\mathrm{i}}=\dfrac{k_iL_i}{r_i}\le 200\ \text{For compression members}\end{cases} \tag{3.33}$$

where f_a $(=P/A_i)$ represents the computed axial stress. The computed flexural stresses due to bending of the member about its major (x) and minor (y) principal axes are denoted by f_{bx} and f_{by}, respectively. F'_{ex} and F'_{ey} denote the Euler stresses about principal axes of the member that are divided by a factor of safety of 23/12. The allowable bending compressive stresses about major and minor axes are designated by F_{bx} and F_{by}. C_{mx} and C_{my} are the reduction factors, introduced to counterbalance overestimation of the effect of secondary moments by the amplification factors $\left(1-\frac{f_a}{F'_{ex}}\right)$. For unbraced frame members, these factors are taken as 0.85. For braced frame members without transverse loading between their ends, these are calculated from $C_m=0.6-0.4M_1/M_2$, where M_1/M_2 is the ratio of smaller end moment to the larger end moment. Finally, for braced frame members having transverse loading between their ends, these factors are determined from the formula $C_m=1+\psi(f_a/F'_e)$ based on a rational approximate analysis outlined in ASD-AISC [9] Commentary-H1, where ψ is a parameter that considers maximum deflection and maximum moment in the member. F_a stands for the allowable axial stress under axial compression force alone, and is calculated depending on elastic or inelastic bucking failure mode of the member according to the slenderness ratio:

$$F_a=\begin{cases}\left[\left(1-\dfrac{\lambda_i^2}{2C_C^2}\right)Fy\right]\Big/\left(\dfrac{5}{3}+\dfrac{3\lambda_i}{8C_C}-\dfrac{\lambda_i^3}{8C_C^3}\right) & \text{For } \lambda_i<C_C\\ \dfrac{12\pi^2E}{23\lambda_i^2} & \text{For } \lambda_i\ge C_C\end{cases} \tag{3.34}$$

where $E=$ the modulus of elasticity; $F_y=$ the yield stress of steel; $C_c=$ the slenderness ratio dividing the elastic and inelastic buckling regions ($C_C=$

$\sqrt{2\pi^2 E/F_y}$); λ_i = the slenderness ratio ($\lambda_i = kL_i/r_i$); k = the effective length factor; and r_i = the governing radius of gyration. For an axially loaded bracing member whose slenderness ratio exceeds 120, F_a is increased by a factor of $(1.6 - L_i/200r_i)$ considering relative unimportance of the member. Equation (23) represents the slenderness limitations imposed on all members such that maximum slenderness ratio is limited to 300 for members under tension, and to 200 for members under compression loads.

Geometric constraints:

Geometric constraints are considered between beams and columns framing into each other at a common joint for practicality of an optimum solution generated. For the two beams B1 and B2 and the column shown in Fig. 3.10, the following geometric constraints are written (Saka and Hasançebi [10]):

$$b_{fb} \leq b_{fc} \tag{3.35}$$

$$b'_{fb} \leq \left(d_c - 2t_f\right) \tag{3.36}$$

where b_{fb}, b'_{fb} and b_{fc} are the flange width of the beam B1, the beam B2 and the column, respectively, d_c is the depth of the column, and t_f is the flange width of the column. Equation (3.35) ensures that the flange width of the beam B1 remains smaller than that of the column. On the other hand, (3.36) enables that flange width of the beam B2 remains smaller than clear distance between the flanges of the column.

Maximum lateral displacement:

$$\frac{\Delta_T}{H} \leq R \tag{3.37}$$

Inter-story displacement constraints:

$$\frac{d_i}{h_i} \leq R_I, \quad i = 1, 2, \ldots, ns \tag{3.38}$$

where Δ_T is the maximum lateral displacement. H is the height of the frame structure. R is the maximum drift index (= 1/400). d_i is the inter-story drift. h_i is the story height of the ith floor. ns represents the total number of stories. R_I is the inter-story drift index permitted by the code of the practice (= 1/400).

3.4.1.3 Design Loads for Frame Structures

The frame examples are subjected to various gravity loads in addition to lateral wind forces. The gravity loads acting on floor slabs cover dead (D), live (L) and snow (S) loads. All the floors excluding the roof are subjected to a design dead load of 2.88 kN/m^2 and a design live load of 2.39 kN/m^2. The roof is subjected to a

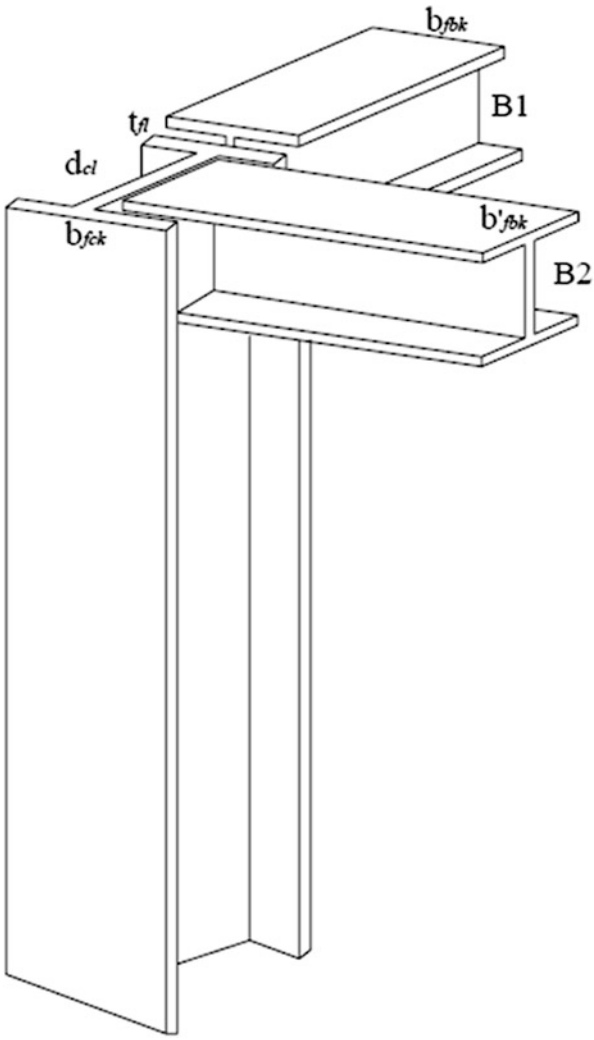

Fig. 3.10 Beam-column geometric constraints [2]

design dead load of 2.88 kN/m^2 plus snow load. The design snow load is computed using the equation (7-1) in ASCE 7-05 [11], resulting in a design snow pressure of 0.75 kN/m^2. The calculated gravity loads are applied as uniformly distributed loads on the beams using distribution formulas developed for slabs. The design wind loads (W) are also computed according to ASCE 7-05 using the following equation:

$$p_w = \left(0.613 K_z K_{zt} K_d V^2 I\right)\left(G C_p\right) \tag{3.39}$$

where p_w is the design wind pressure in kN/m^2; K_z (=1.07) is the velocity exposure coefficient; K_{zt} (=1.0) is the topographic factor, K_d (=0.85) is the wind directionality factor; I (=1.15) is the importance factor; and V (=46.94 m/s) is the basic

wind; G (=0.85) is the gust factor, and C_p (=0.8 for windward face and −0.5 for leeward face) is the external pressure coefficient. The calculated wind loads are applied as uniformly distributed lateral loads on the external beams of the frames located on windward and leeward facades at every floor level.

The load combination per AISC-ASD specification is considered as

$$\begin{aligned}&(\mathrm{D}+\mathrm{L}+\mathrm{S}+\mathrm{W_x}).\\&(\mathrm{D}+\mathrm{L}+\mathrm{S}+\mathrm{W_y}).\end{aligned}$$

It should be noted that for wind forces in the above load combinations two cases are considered. In the first case, the wind loading is acting along x-axis, whereas in the second one it is applied along y-axis.

3.4.2 CSS Algorithm-Based Structural Optimization Procedure

As defined in the previous section, there are some problem-specific constraints in structural optimization problems that must be handled. The penalty function method has been the most popular constraint-handling technique due to its simple principle and ease of implementation. In utilizing the penalty functions, if the constraints are between the allowable limits, the penalty will be zero; otherwise, the amount of penalty is obtained by dividing the violation of allowable limit to the limit itself. Since the CSS is independent of the type of penalty function, one can easily utilize another approach in the application of CSS.

Detailed procedure of the proposed CSS algorithm-based method to determine optimal design of structures is shown in Fig. 3.11. Considering the rules defined for the CSS in Sect. 3, and utilizing the penalty functions to handle the problem-specific constraints, the CSS algorithm-based structural optimization procedure can be divided into the following three phases:

Phase 1: Initialization CSS algorithm parameters such as N, CMS, k_v, k_a and design variable bounds are initialized. An array of CPs with random positions and their associated velocities considering variable bounds are randomly generated that are equal to the size of the N. The generated CPs are analyzed and the values of the fitness function for the CPs considering the weight of the structure and the penalty functions are evaluated. Then, CPs are ranked in an increasing order. CMS number of the first CPs and their related values of the fitness function are stored in the CM.

Phase 2: Search Each CP moves to the new position considering the probability of motion (3.24), the magnitude of the attracting force vector (3.25) and the motion laws (3.26) and (3.27). If each CP exits from the allowable search space, its position is corrected using the harmony-based algorithm. Then, the new CPs are analyzed to

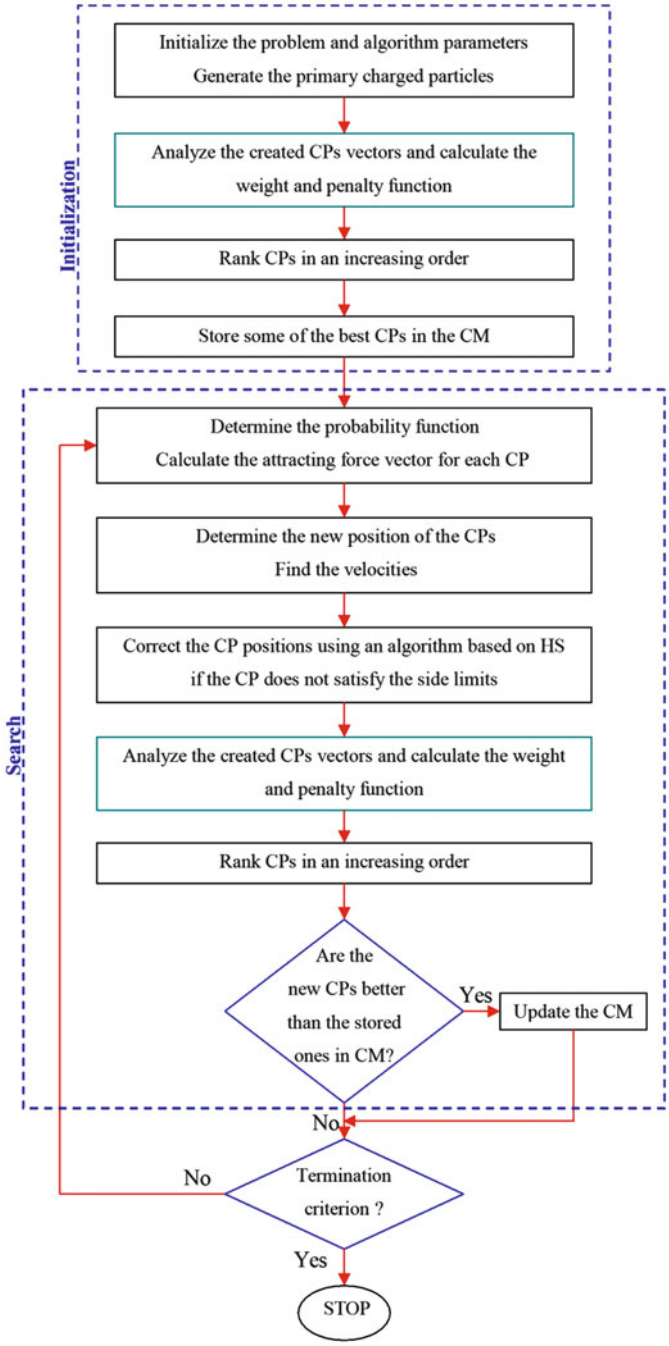

Fig. 3.11 The flowchart of the CSS for the truss structures [2]

evaluate the fitness function and to sort them increasingly. Then, some of the good new CPs are stored in the CM and the worst ones are excluded from the CM.

Phase 3: Terminating Criterion Controlling Search level is continued until a terminating criterion is satisfied.

3.5 Numerical Examples

In this section, three truss and two frame structures are optimized utilizing the new algorithm. The final results are then compared to the solutions of other advanced heuristic methods to demonstrate the efficiency of this work. For the CSS algorithm, a population of 20 CPs is used for the first and the second truss examples and a population of 50 candidates is selected for the remaining examples. The effect of the pervious velocity and the resultant force affecting a CP can decrease or increase based on the values of the k_v and k_a. Here, k_v and k_a are defined as

$$\begin{aligned} k_v &= c(1 - iter/iter_{\max}) \\ k_a &= c(1 + iter/iter_{\max}) \end{aligned} \tag{3.40}$$

where *iter* is the iteration number, and $iter_{\max}$ is the maximum number of the iterations, c is set to 0.5 and 0.2 when the population of 20 and 50 CPs are selected, respectively. With this equation, k_v decreases linearly while k_a increases when the number of iterations raises. In this way, the balance between the exploration and the fast rate of convergence is saved.

In order to investigate the effect of the initial solution on the final result and because of the stochastic nature of the algorithm, each example is independently solved several times. The initial population in each of these runs is generated in a random manner according to Rule 2. The first two truss examples are optimized by the CSS algorithm for 50 times, while performance comparisons of the CSS method in other examples based on 20 evaluations. The algorithms are coded in Matlab and structures are analyzed using the direct stiffness method.

3.5.1 A Benchmark Truss

The topology and nodal numbering of a 25-bar spatial truss structure, shown in Fig. 3.12, are known as a benchmark example in the field of structural optimization. The material density is considered as 0.1 lb/in^3 (2767.990 kg/m^3) and the modulus of elasticity is taken as 10,000 ksi (68,950 MPa). Twenty five members are categorized into eight groups, as follows: (1) A_1, (2) A_2–A_5, (3) A_6–A_9, (4) A_{10}–A_{11}, (5) A_{12}–A_{13}, (6) A_{14}–A_{17}, (7) A_{18}–A_{21}, and (8) A_{22}–A_{25}.

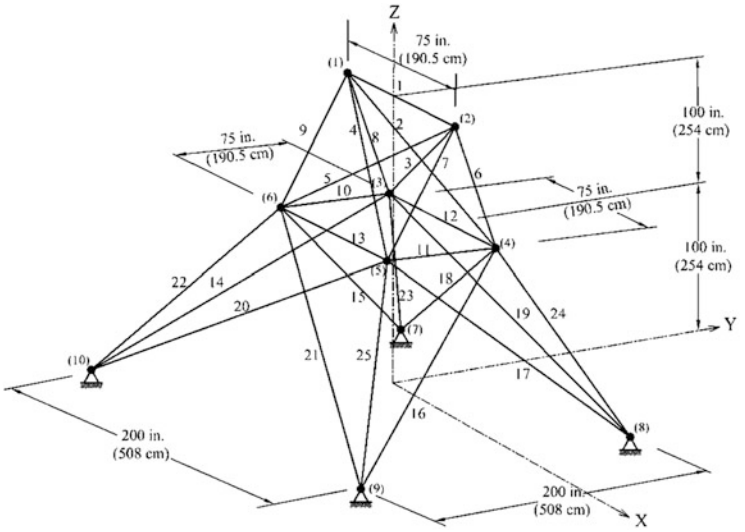

Fig. 3.12 Schematic of a twenty five-bar spatial truss [2]

Table 3.3 Loading conditions for the 25-bar spatial truss

	Case 1			Case 2		
Node	P_X *kips* (kN)	P_Y *kips* (kN)	P_Z *kips* (kN)	P_X *kips* (kN)	P_Y *kips* (kN)	P_Z *kips* (kN)
1	0.0	20.0 (89)	−5.0 (22.25)	1.0 (4.45)	10.0 (44.5)	−5.0 (22.25)
2	0.0	−20.0 (89)	−5.0 (22.25)	0.0	10.0 (44.5)	−5.0 (22.25)
3	0.0	0.0	0.0	0.5 (2.22)	0.0	0.0
6	0.0	0.0	0.0	0.5 (2.22)	0.0	0.0

Table 3.4 Member stress limitation for the 25-bar spatial truss

Element group		Compressive stress limitations, ksi (MPa)	Tensile stress limitations, ksi (MPa)
1	A_1	35.092 (241.96)	40.0 (275.80)
2	A_2–A_5	11.590 (79.913)	40.0 (275.80)
3	A_6–A_9	17.305 (119.31)	40.0 (275.80)
4	A_{10}–A_{11}	35.092 (241.96)	40.0 (275.80)
5	A_{12}–A_{13}	35.092 (241.96)	40.0 (275.80)
6	A_{14}–A_{17}	6.759 (46.603)	40.0 (275.80)
7	A_{18}–A_{21}	6.959 (47.982)	40.0 (275.80)
8	A_{22}–A_{25}	11.082 (76.410)	40.0 (275.80)

This spatial truss is subjected to two loading conditions shown in Table 3.3. Maximum displacement limitations of ±0.35 in (±8.89 mm) are imposed on every node in every direction and the axial stress constraints vary for each group as shown in Table 3.4. The range of cross-sectional areas varies from 0.01 to 3.4 in^2 (0.6452 cm^2 to 21.94 cm^2).

Table 3.5 Performance comparison for the 25-bar spatial truss

	Optimal cross-sectional areas (in^2)								
	Rajeev and Krishnamoorthy	Schutte and Groenwold	Lee and Geem	Kaveh et al.	Kaveh and Talatahari			Present work [2]	
Element group	GA [14]	PSO [15]	HS [8]	IACS [13]	PSACO [4]	HPSACO [4]	HBB–BC [12]	in^2	cm^2
1 A_1	0.10	0.010	0.047	0.010	0.010	0.010	0.010	0.010	0.065
2 A_2–A_5	1.80	2.121	2.022	2.042	2.052	2.054	1.993	2.003	12.923
3 A_6–A_9	2.30	2.893	2.950	3.001	3.001	3.008	3.056	3.007	19.400
4 A_{10}–A_{11}	0.20	0.010	0.010	0.010	0.010	0.010	0.010	0.010	0.065
5 A_{12}–A_{13}	0.10	0.010	0.014	0.010	0.010	0.010	0.010	0.010	0.065
6 A_{14}–A_{17}	0.80	0.671	0.688	0.684	0.684	0.679	0.665	0.687	4.432
7 A_{18}–A_{21}	1.80	1.611	1.657	1.625	1.616	1.611	1.642	1.655	10.677
8 A_{22}–A_{25}	3.0	2.717	2.663	2.672	2.673	2.678	2.679		17.161
Best weight (lb)	546	545.21	544.38	545.03	545.04	544.99	545.16	545.10	2,424.7 N
Average weight (lb)	N/A	546.84	N/A	545.74	N/A	545.52	545.66	545.58	2,426.8 N
Std Dev(lb)	N/A	1.478	N/A	0.620	N/A	0.315	0.367	0.412	1.833 N
No. of analyses	N/A	9,596	15,000	3,520	28,850	9,875	12,500	7,000	

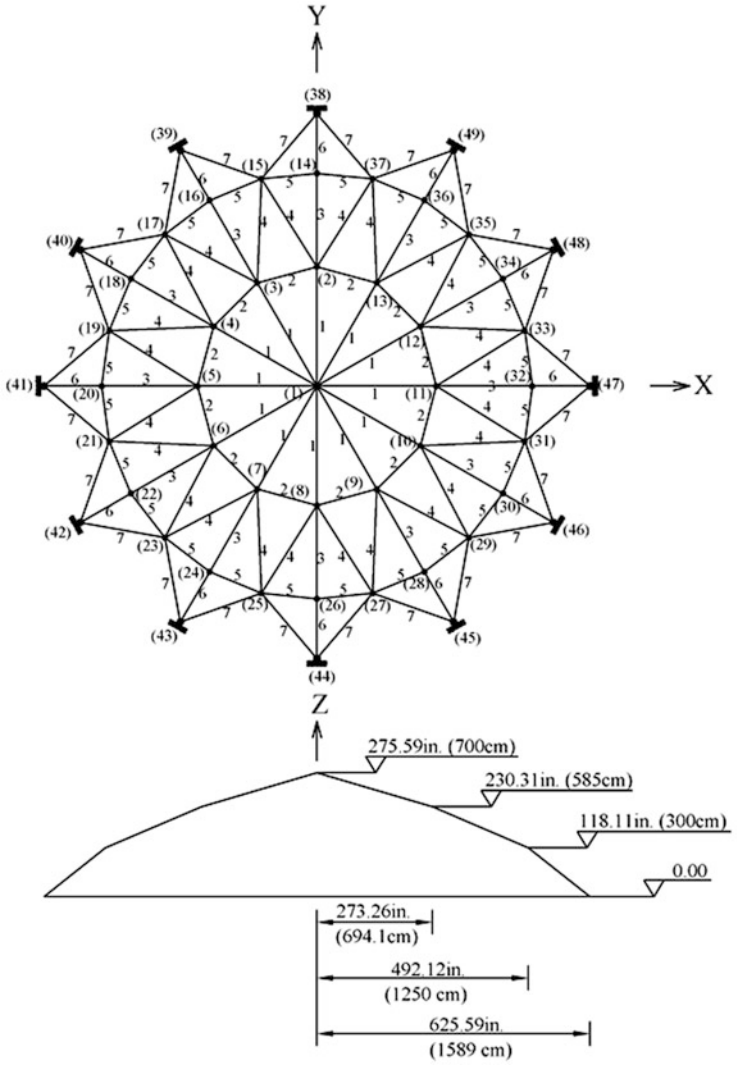

Fig. 3.13 Schematic of a 120-bar dome shaped truss [2]

The CSS algorithm achieves the best solution after 7,000 searches. However, the HBB–BC (Kaveh and Talatahari [12]) and HPSACO (Kaveh and Talatahari [4]) algorithms find the best solution after about 12,500 and 9,875 analyses respectively, which are 50 % and 41 % more than the present work. The best weight of the CSS is 545.10 lb. Although the CSS approach has slightly worse performance than the improved methods IACS (Kaveh et al. [13]) and HPSACO (Kaveh and Talatahari [4]), it performs better than other algorithms (GA (Rajeev and Krishnamoorthy [14]), PSO (Schutte and Groenwold [15]) and HS (Lee and Geem [8]) when the best

Table 3.6 Performance comparison for the 120-bar dome truss

	Optimal cross-sectional areas (in^2)						
	Kaveh et al.	Kaveh and Talatahari				Present work [2]	
Element group	IACS [13]	PSOPC [4]	PSACO [4]	HPSACO [4]	HBB–BC [12]	in^2	cm^2
1 A_1	3.026	3.040	3.026	3.095	3.037	3.027	19.529
2 A_2	15.06	13.149	15.222	14.405	14.431	14.606	94.232
3 A_3	4.707	5.646	4.904	5.020	5.130	5.044	32.542
4 A_4	3.100	3.143	3.123	3.352	3.134	3.139	20.252
5 A_5	8.513	8.759	8.341	8.631	8.591	8.543	55.116
6 A_6	3.694	3.758	3.418	3.432	3.377	3.367	21.723
7 A_7	2.503	2.502	2.498	2.499	2.500	2.497	16.110
Best weight (lb)	33320.52	33481.2	33263.9	33248.9	33287.9	33251.9	147912 N
No. of analyses	3,250	150,000	32,600	10,000	10,000	7,000	

weight, the average weight or the standard deviation are compared. Table 3.5 presents a comparison of the performance of the CSS algorithm and other heuristic algorithms.

3.5.2 A 120-Bar Dome Truss

The topology and group numbers of 120-bar dome truss are shown in Fig. 3.13. The modulus of elasticity is 30,450 ksi (210,000 MPa), and the material density is 0.288 lb/in^3 (7971.810 kg/m^3). The yield stress of steel is taken as 58.0 ksi (400 MPa). The dome is considered to be subjected to vertical loading at all the unsupported joints. These loads are taken as −13.49 kips (−60 kN) at node 1, −6.744 kips (−30 kN) at nodes 2–14, and −2.248 kips (−10 kN) at the rest of the nodes. The minimum cross-sectional area of all members is 0.775 in^2 (2 cm^2) and the maximum cross-sectional area is taken as 20.0 in^2 (129.03 cm^2). The constraints are considered as:

1. Stress constraints (according to the AISC-ASD (1989) code):

$$\begin{cases} \sigma_i^+ = 0.6F_y & for\ \ \sigma_i \geq 0 \\ \sigma_i^- & for\ \ \sigma_i < 0 \end{cases} \tag{3.41}$$

   where σ_i^- is calculated considering the slenderness ratio (3.34).
2. Displacement limitations of ±0.1969 in (±5 mm) are imposed on all nodes in x, y and z directions.

Table 3.6 illustrates the best solution vectors, the corresponding weights and the required number of analyses for convergence in the present algorithm and

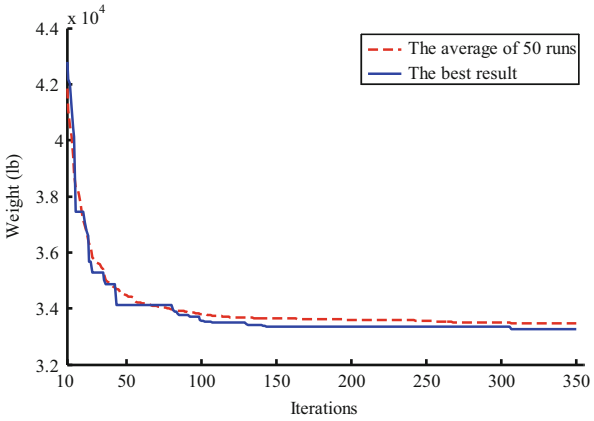

Fig. 3.14 Convergence history of the 120-bar dome shaped truss for the CSS algorithm [2]

some of other heuristic methods. Except IACS which uses two auxiliary mechanisms for searching, the CSS optimization has best convergence rates. Figure 3.14 shows the best and average convergence history for the results of the CSS. In addition, CSS and HPSACO find the best result among the other metaheuristics. A comparison of the allowable and existing stresses and displacements of the 120-bar dome truss structure using CSS is shown in Fig. 3.15. The maximum value for displacement is equal to 0.19689 in (5 mm) and the maximum stress ratio is equal to 99.98 %.

3.5.3 A 26-Story Tower Space Truss

The 26-story tower space truss containing 942 elements and 244 nodes is considered as the large-scale truss example. Fifty-nine design variables are used to represent the cross-sectional areas of 59 element groups in this structure, employing the symmetry of the structure. Figure 3.16 shows the geometry and the 59 element groups. The material density is 0.1 lb/in^3 (2,767.990 kg/m^3) and the modulus of elasticity is 10,000 ksi (68,950 MPa).

The members are subjected to the stress limits of $\pm$25 ksi (172.375 MPa) and the four nodes of the top level in the x, y, and z directions are subjected to the displacement limits of $\pm$15.0 in (38.10 cm) (about 1/250 of the total height of the tower). The allowable cross-sectional areas in this example are selected from 0.1 to 20.0 in^2 (from 0.6452 cm^2 to 129.032 cm^2). The loading on the structure consists of:

1. The vertical load at each node in the first section is equal to -3 kips (-13.344 kN);
2. The vertical load at each node in the second section is equal to -6 kips (-26.688 kN);

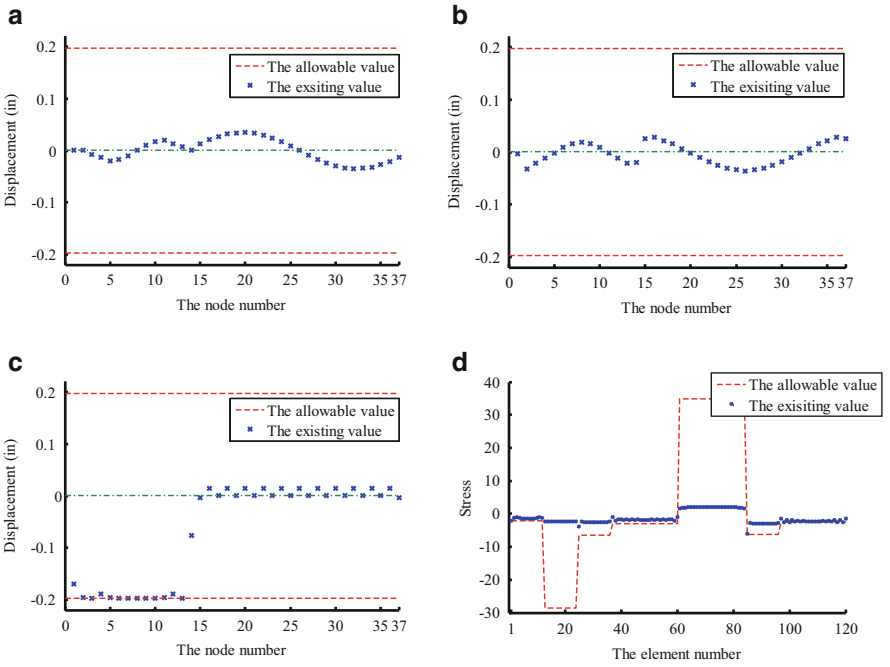

Fig. 3.15 Comparison of the allowable and existing constraints for the 120-bar dome shaped truss using the CSS [2]. (**a**) Displacement in the x direction, (**b**) displacement in the y direction, (**c**) displacement in the z direction and (**d**) stress

3. The vertical load at each node in the third section is equal to −9 kips (−40.032 kN);
4. The horizontal load at each node on the right side in the x direction is equal to −1 kips (−4.448kN);
5. The horizontal load at each node on the left side in the x direction is equal to 1.5 kips (6.672kN);
6. The horizontal load at each node on the front side in the y direction is equal to −1 kips (−4.448kN);
7. The horizontal load at each node on the back side in the x direction is equal to 1 kips (4.448kN).

The CSS method achieved a good solution after 15,000 analyses and found an optimum weight of 47,371 lb (210,716 N). The best weights for the GA, PSO, BB–BC and HBB–BC are 56,343 lb (250,626 N), 60,385 lb (268,606 N), 53,201 lb (236,650 N) and 52,401 lb (233,091 N), respectively. In addition, CSS has better performance in terms of the optimization time, standard deviation and the average weight. Table 3.7 provides the statistic information for this example. The stress constraints are dominant in this example. The maximum value of stress ratio is equal to 96.7 %. Figure 3.17 compares the allowable and existing stresses in the

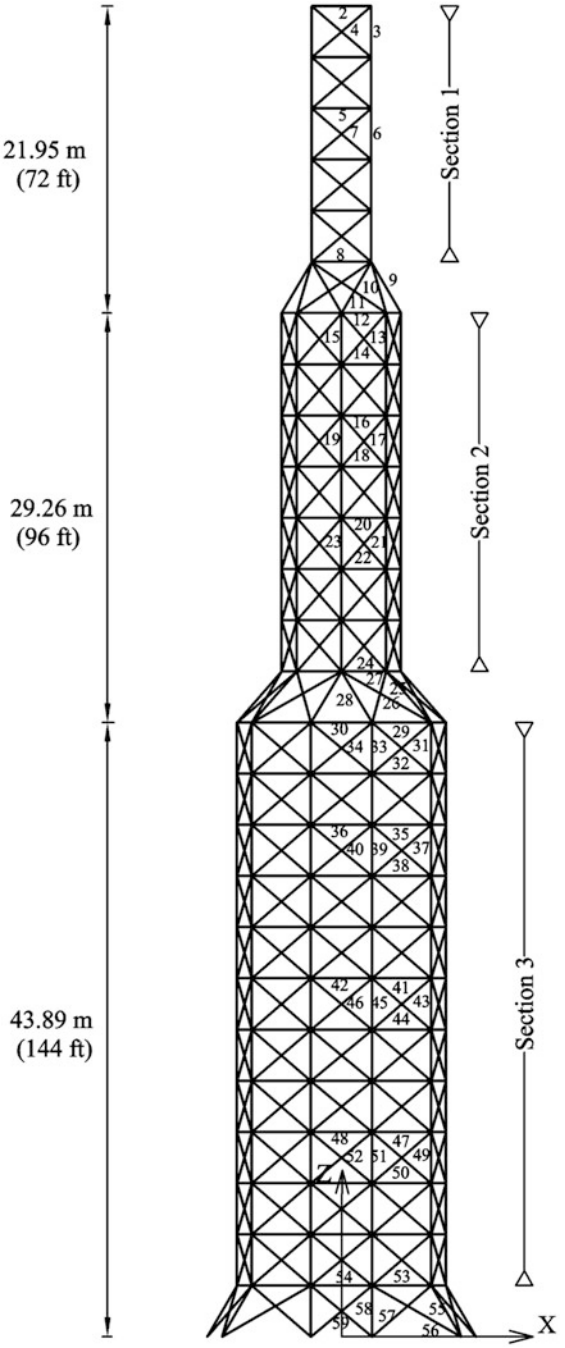

Fig. 3.16 Schematic of a 26-story tower truss [2]

Table 3.7 Performance comparison for the 26-story tower spatial truss

	Kaveh and Talatahari [12]				
	GA	PSO	BB–BC	HBB–BC	Present work [2]
Best weight (lb)	56,343 (250,626 N)	60,385 (26,8606 N)	53,201 (236,650 N)	52,401 (233,091 N)	47,371 (210,716 N)
Average weight (lb)	63,223 (281230 N)	75,242 (334,693 N)	55,206 (245,568 N)	53,532 (238,122 N)	48,603 (216,197 N)
Std Dev (lb)	6,640.6 (29,539 N)	9,906.6 (44,067 N)	2,621.3 (11,660 N)	1,420.5 (6,318 N)	950.4 (4,227 N)
No. of analyses	50,000	50,000	50,000	30,000	15,000
Optimization time (s)	4,450	3,640	3,162	1,926	1,340

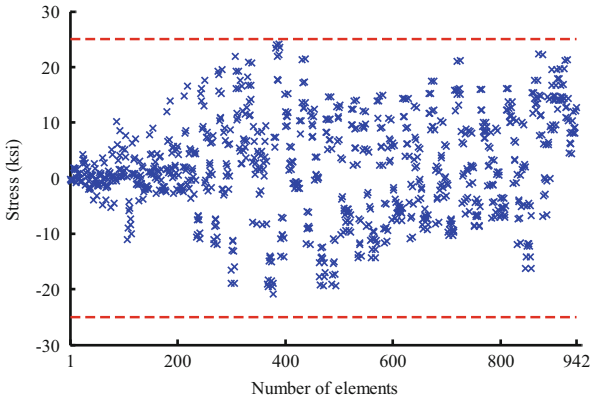

Fig. 3.17 Comparison of the allowable and existing stress constraints for the 26-story tower truss truss using the CSS [2]

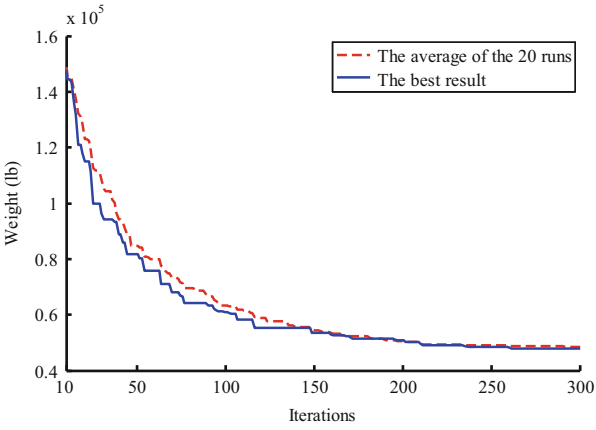

Fig. 3.18 Convergence history of the 26-story tower truss for the CSS algorithm [2]

elements for the CSS result. The convergence history is shown in Fig. 3.18. The final designs obtained by the CSS technique for this example is given in Table 3.8.

3.5.4 An Unbraced Space Frame

A 10-story space steel frame consisting of 256 joints and 568 members is shown in Fig. 3.19. This problem has been formerly studied by Saka and Hasançebi [10] to evaluate the performance of a HS-based technique in real size optimum design of steel frameworks considering ASD-AISC as the code of the practice.

The columns in a story are collected in three member groups as corner columns, inner columns and outer columns, whereas beams are divided into two groups as inner beams and outer beams. The corner columns are grouped together as having the same section in the first three stories and then over two adjacent stories

Table 3.8 The optimum design of the CSS algorithm for the 26-story tower spatial truss

	Optimal cross-sectional areas (cm^2)							
	Members	Area		Members	Area		Members	Area
1	A_1	0.962	21	A_{21}	2.780	41	A_{41}	0.417
2	A_2	2.557	22	A_{22}	0.430	42	A_{42}	0.679
3	A_3	1.650	23	A_{23}	3.048	43	A_{43}	19.584
4	A_4	0.402	24	A_{24}	5.112	44	A_{44}	0.533
5	A_5	0.657	25	A_{25}	19.352	45	A_{45}	1.640
6	A_6	18.309	26	A_{26}	0.476	46	A_{46}	0.618
7	A_7	0.346	27	A_{27}	2.887	47	A_{47}	0.531
8	A_8	3.076	28	A_{28}	19.500	48	A_{48}	1.374
9	A_9	2.235	29	A_{29}	4.772	49	A_{49}	19.656
10	A_{10}	3.813	30	A_{30}	5.063	50	A_{50}	0.888
11	A_{11}	0.856	31	A_{31}	15.175	51	A_{51}	4.456
12	A_{12}	1.138	32	A_{32}	1.176	52	A_{52}	0.386
13	A_{13}	3.374	33	A_{33}	0.839	53	A_{53}	10.398
14	A_{14}	0.573	34	A_{34}	1.394	54	A_{54}	18.834
15	A_{15}	19.530	35	A_{35}	0.153	55	A_{55}	18.147
16	A_{16}	1.512	36	A_{36}	0.247	56	A_{56}	3.280
17	A_{17}	2.667	37	A_{37}	18.673	57	A_{57}	2.972
18	A_{18}	0.478	38	A_{38}	0.696	58	A_{58}	4.927
19	A_{19}	17.873	39	A_{39}	1.395	59	A_{59}	0.288
20	A_{20}	0.335	40	A_{40}	0.422			
Weight (N)		210716						

thereafter, as are corner columns, inner columns, outer columns, inner beams and outer beams. This results in a total of 25 distinct design groups.

The optimum design of the space frame described above is carried out using the CSS and compared with those of the simulated annealing (SA), evolution strategies (ESs), particle swarm optimizer (PSO), tabu search optimization (TSO), simple genetic algorithm (SGA), ant colony optimization (ACO), and harmony search (HS) methods (Saka and Hasançebi [10]). In each optimization technique the number of iterations was taken as 50,000, when ASD-AISC is used as the code of the practice. Our investigation shows that 12,500 analyses are sufficient as the maximum number of analyses for the CSS. This shows that the CSS can reach a similar result as the other methods with smaller number of analyses. The design history of each run by each technique is shown in Fig. 3.20.

The optimum design attained by the CSS method for this example is 225,654.0 kg, while it is 228,588.3 kg for the ESs. Among the metaheuristic algorithms, the adaptive harmony search algorithm is the third best which is 1.6 % heavier than the one obtained by evolutionary strategies algorithm. The optimum result for the TSO, SA, ACO, SGA and PSO is 235,167.5 kg, 238,756.5 kg, 241,470.31 kg, 245,564.80 kg and 253,260.23 kg, respectively. The minimum weights as well as W-section designations obtained by some of the best algorithms are provided in Table 3.9.

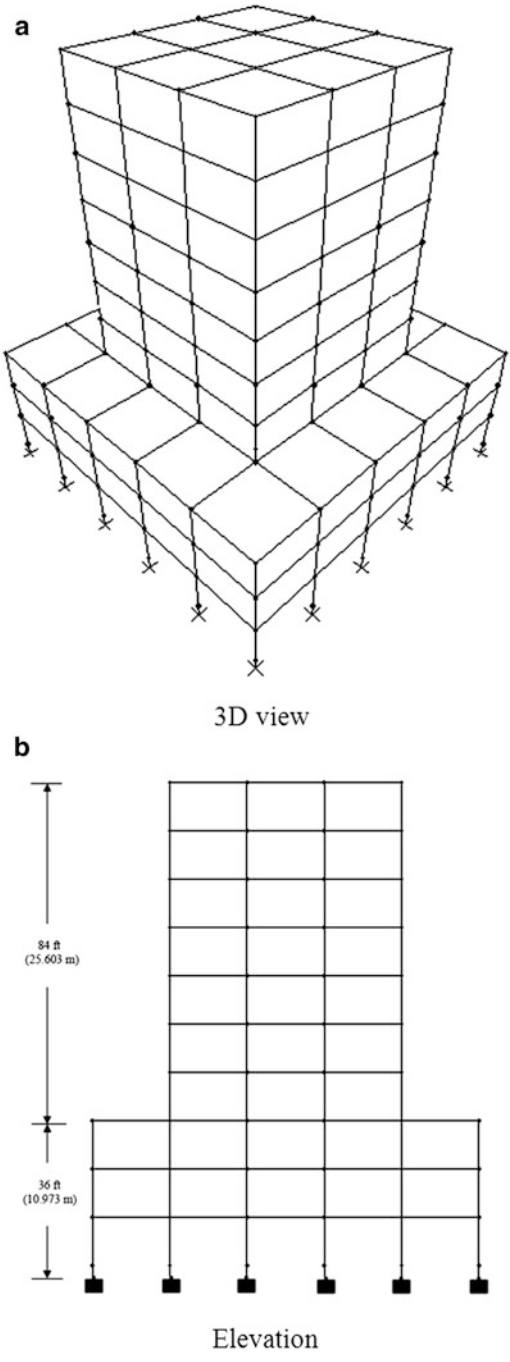

Fig. 3.19 (continued)

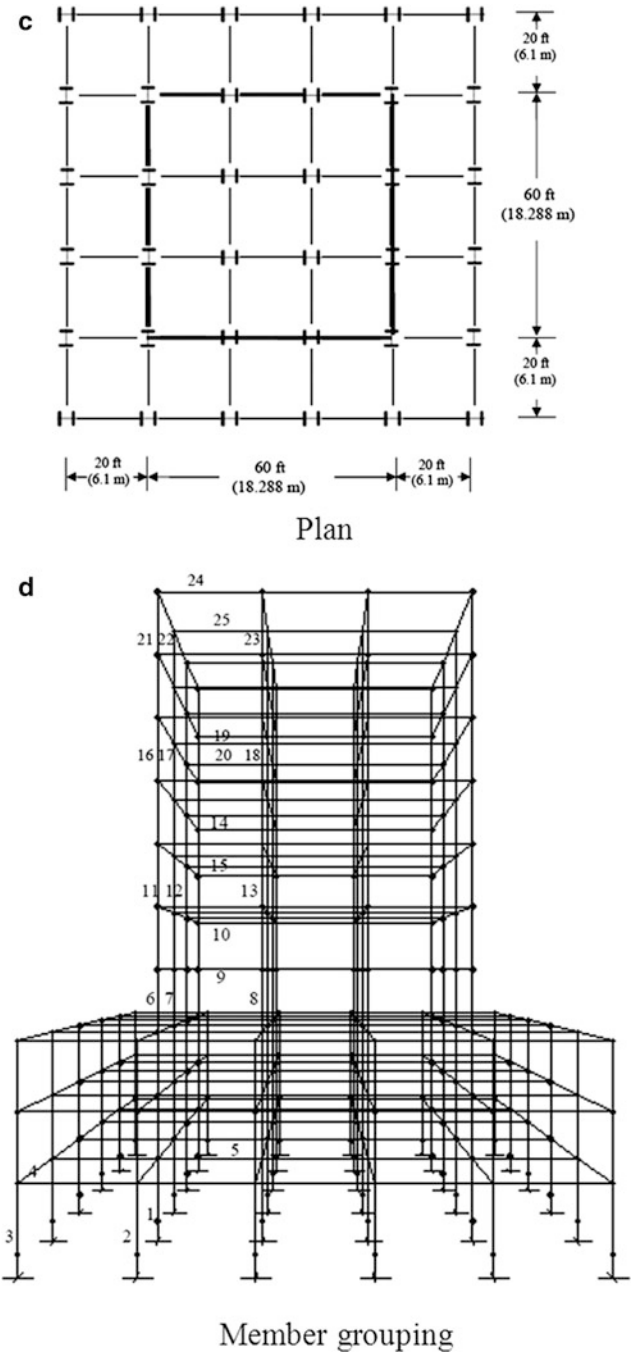

Fig. 3.19 Schematic of an unbraced space frame [2]

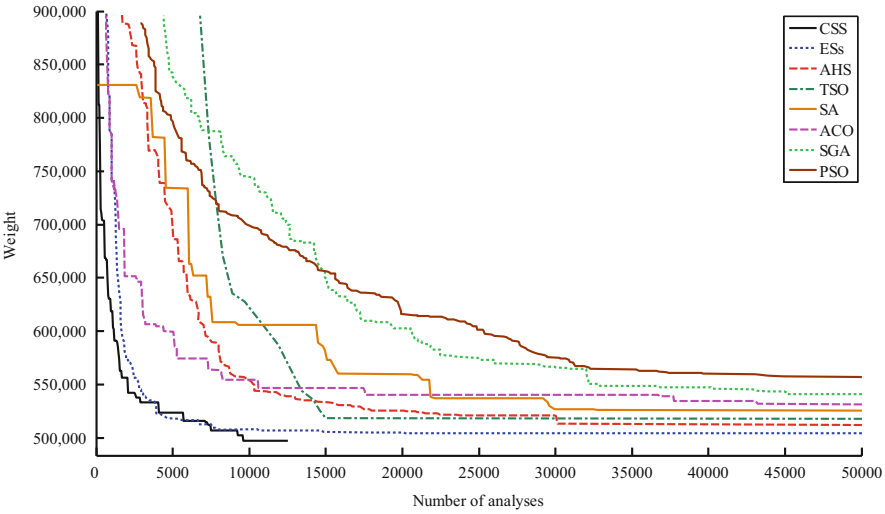

Fig. 3.20 Comparison of the convergence history for the unbraced space frame [2]

3.5.5 A Braced Space Frame

The second frame example considered in this paper is a 36-story braced space steel frame consisting of 814 joints and 1,860 members, as shown in Fig. 3.21, Saka and Hasançebi [10]. An economical and effective stiffening of the frame against lateral forces is achieved through exterior diagonal bracing members located on the perimeter of the building, which also participate in transmitting the gravity forces.

The 1,860 frame members are collected in 72 different member groups, considering the symmetry of the structure and the practical fabrication requirements. That is, the columns in a story are collected in three member groups as corner columns, inner columns and outer columns, whereas beams are divided into two groups as inner beams and outer beams. The corner columns are grouped together as having the same section over three adjacent stories, as are inner columns, outer columns, inner beams and outer beams. Bracing members on each facade are designed as three-story deep members, and two bracing groups are specified in every six stories.

The minimum weight design of the frame is equal to 2,301.69 ton for the CSS algorithm while it is 2,383.60 ton and 4,438.17 ton for the adaptive harmony search and the simple harmony search algorithms, respectively. Figure 3.22 shows the design history graph obtained for this example. In the optimum design process, CSS finds the optimum design after 12,500 analyses, while ES needs 50,000 searches to determine the optimum solution.

Table 3.9 Optimal design for the unbraced space frame

	Optimal W-shaped sections				
	Saka and Hasançebi (2009)				
Element group	SA	TSO	AHS	ESs	Present work [2]
1	W14X193	W14X193	W14X176	W14X193	W14X132
2	W8X48	W8X48	W14X48	W8X48	W21X55
3	W8X40	W8X40	W10X39	W10X39	W12X40
4	W10X22	W10X22	W10X22	W10X22	W10X33
5	W21X44	W21X50	W24X55	W21X50	W21X50
6	W12X65	W10X54	W12X65	W10X54	W12X65
7	W14X145	W14X120	W14X145	W14X109	W14X99
8	W14X145	W14X159	W14X159	W14X176	W14X120
9	W24X65	W21X44	W14X30	W18X40	W21X44
10	W24X55	W18X40	W18X40	W18X40	W21X44
11	W10X49	W10X45	W10X54	W10X49	W14X61
12	W14X90	W14X90	W14X90	W14X90	W10X88
13	W14X120	W12X120	W14X120	W14X109	W14X99
14	W16X36	W12X44	W14X34	W14X30	W18X35
15	W16X40	W16X36	W18X40	W16X36	W12X50
16	W12X40	W10X33	W8X31	W12X45	W21X68
17	W12X65	W12X65	W12X65	W12X65	W16X57
18	W12X26	W14X34	W18X35	W10X22	W24X68
19	W12X72	W12X79	W12X79	W12X79	W16X36
20	W16X36	W14X30	W14X30	W14X30	W16X31
21	W8X24	W10X39	W10X22	W8X35	W10X33
22	W10X49	W12X45	W10X45	W10X39	W16X31
23	W8X24	W12X35	W8X31	W8X31	W8X28
24	W12X26	W6X20	W10X22	W8X18	W8X18
25	W12X26	W12X26	W12X26	W14X30	W10X26
Weight (kg)	238,756.5	235,167.5	232,301.2	228,588.3	225,654.0

3.6 Discussion

3.6.1 Efficiency of the CSS Rules

Solution of a number of design examples shows the superiority of the CSS algorithm to the other existing metaheuristics. To investigate the effect of some utilized rules, a number of the CSS-based algorithms are defined as follows:

Case 1: Rule 3 is changed as:
The kind of the electric forces between two charged particles is selected randomly.
Case 2: Rule 4 is changed as:
Any CP can act on another one; i.e. a bad CP can affect a good one and vice versa ($p_{ij} = 1$).
Case 3: Rule 4 is changed as:

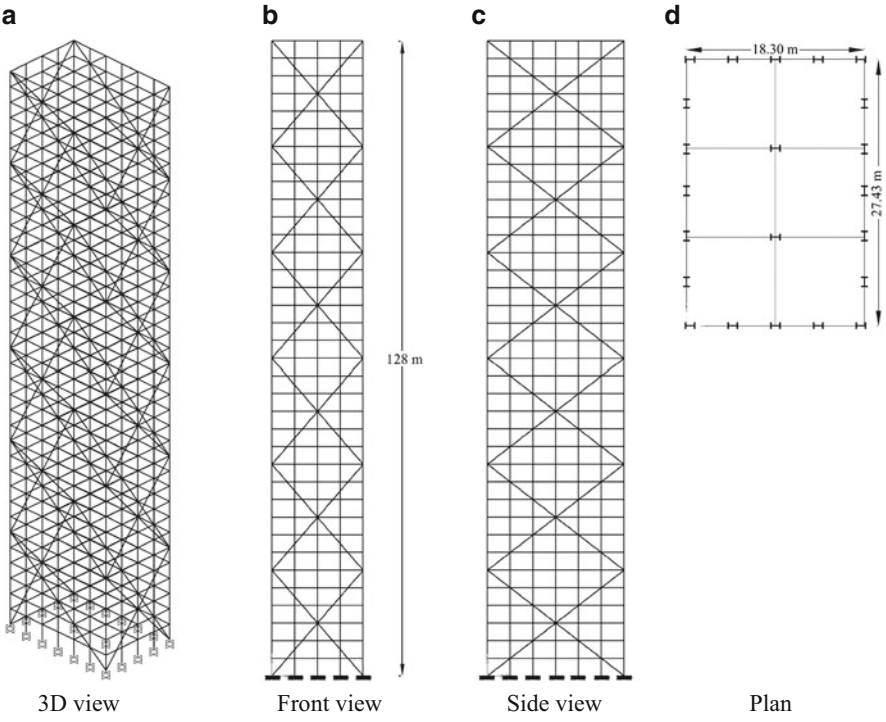

Fig. 3.21 Schematic of a braced space frame [2]

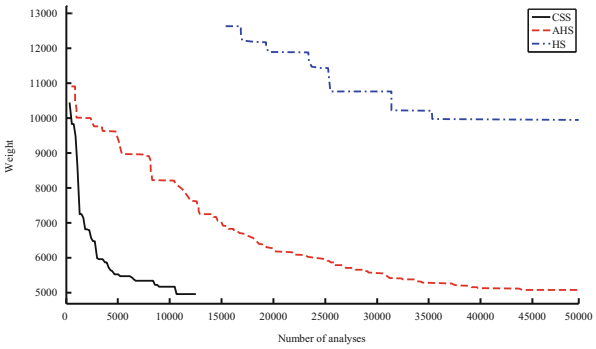

Fig. 3.22 Comparison of the convergence history for the braced space frame [2]

Only good CPs can attract bad CPs.
Case 4: Rule 5 is changed as:
Always $i_1 = 0$ and $i_2 = 1$.
Case 5: Rule 5 is changed as:
Always $i_1 = 1$ and $i_2 = 0$.

Table 3.10 Investigation on the performance of various CSS-based algorithms for the 25-bar truss in 50 runs

	Case 1	Case 2	Case 3	Case 4	Case 5
Best weight (lb)	551.31	551.10	545.99	546.28	550.55
Average weight (lb)	554.75	552.39	549.42	547.06	550.90
Std. Dev. (lb)	1.210	0.885	1.467	0.707	0.686

Table 3.10 shows the results of the 50 runs of the first example for the each case. Comparing the result of Case 1 with the result of the original CSS (Table 3.5) confirms that considering repulsive forces between CPs reduces the power of the algorithm. Although when a good agent attracts a bad one, the exploitation ability for the algorithm is provided, and vice versa if a bad CP attracts a good CP, the exploration is provided, however differences between the results of the Cases 2 and 3 with the original CSS indicated that when all bad agents attract good ones, a disorder will be created and when only good CPs attract bad ones the convergence will occur very soon and a complete search will not be performed. As a result, at least the computational cost to reach a good solution may increase for the condition of the Cases 2 and 3.

3.6.2 Comparison of the PSO and CSS

Both the CSS and the PSO are multi-agent algorithms in which the position of each agent is obtained by adding the agent's movement to its previous position; however the movement strategies are different. In the PSO algorithm, each particle continuously focuses and refocuses on the effort of its search according to both local best and global best, while the CSS approach uses the governing laws from electrical physics and the governing laws of motion from the Newtonian mechanics to determine the amount and the direction of a charged particle' movement. The potency of the PSO is summarized to find the direction of an agent' movement, while the CSS method can determine not only the directions but also the amount of movements. In the PSO, the direction of an agent is calculated using only two best positions containing local best and global best. However, in the CSS the agent direction is calculated based on the overall forces resulted by the best agents stored in the CM and some of the best current CPs. CSS can distinguish finishing the global phase and change the movement updating equation for the local phase to have a better balance between the exploration and exploitation. While one of the greatest disadvantages of the PSO approach is the existence of some difficulties in controlling the balance between the exploration and exploitation due to ignoring the effect of other agents, Kaveh and Talatahari [4].

3.6.3 *Efficiency of the CSS*

CSS utilizes the Coulomb and Gauss laws to determine the direction and the amount of the movement of each agent and uses some laws of the Newtonian mechanics to complete the movement process. Compared to other metaheuristics, CSS has less computing cost and can determine the optimum result with a smaller number of analyses. Due to having a good balance between the exploration and exploitation, the performance of the CSS in both global search stage (initial iterations) and the local search stage (last iterations) is good. The comparison of the CSS results with those of the other heuristics shows the robustness of the present algorithm and demonstrates the efficiency of the algorithm to find optimum design of structures.

References

1. Kaveh A, Talatahari S (2010) A novel heuristic optimization method: charged system search. Acta Mech 213(3–4):267–286
2. Kaveh A, Talatahari S (2010) Optimal design of truss structures via the charged system search algorithm. Struct Multidiscip Optim 37(6):893–911
3. Halliday D, Resnick R, Walker J (2008) Fundamentals of physics, 8th edn. Wiley, USA
4. Kaveh A, Talatahari S (2009) Particle swarm optimizer, ant colony strategy and harmony search scheme hybridized for optimization of truss structures. Comput Struct 87(5–6):267–283
5. Coello CAC (2002) Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. Comput Meth Appl Mech Eng 191(11–12):1CA.245–1CA.287
6. Kaveh A, Talatahari S (2009) A particle swarm ant colony optimization for truss structures with discrete variable. J Construct Steel Res 65(8–9):1558–1568
7. Tsoulos IG (2008) Modifications of real code genetic algorithm for global optimization. Appl Math Comput 203:598–607
8. Lee KS, Geem ZW (2004) A new structural optimization method based on the harmony search algorithm. Comput Struct 82:781–798
9. American Institute of Steel Construction (AISC) (1989) Manual of steel construction-allowable stress design, 9th edn. AISC, Chicago, IL
10. Saka MP, Hasançebi O (2009) Design code optimization of steel structures using adaptive harmony search algorithm. In: Geem ZW (ed) Chapter 3 of a book entitled: harmony search algorithms for structural design. Springer, Heidelberg
11. ASCE 7-05, Minimum design loads for building and other structures, Standards ASCE/SEI 7-05
12. Kaveh A, Talatahari S (2009) Size optimization of space trusses using Big Bang–Big Crunch algorithm. Comput Struct 87:1129–1140
13. Kaveh A, Farahmand Azar B, Talatahari S (2008) Ant colony optimization for design of space trusses. Int J Space Struct 23(3):167–181
14. Rajeev S, Krishnamoorthy CS (1992) Discrete optimization of structures using genetic algorithms. J Struct Eng ASCE 118(5):1233–50
15. Schutte JJ, Groenwold AA (2003) Sizing design of truss structures using particle swarms. Struct Multidiscip Optim 25:261–269

Chapter 4
Magnetic Charged System Search

4.1 Introduction

This chapter consists of two parts. In first part, the standard Magnetic Charged System Search (MCSS) is presented and applied to different numerical examples to examine the efficiency of this algorithm. The results are compared to those of the original charged system search method [1].

In the second part, an improved form of the MCSS algorithm, denoted by IMCSS, is presented and also its discrete version is described. The IMCSS algorithm is applied to optimization of truss structures with continuous and discrete variables to demontrate the performance of this algorithm in the field of structural optimization [2].

4.2 Magnetic Charged System Search Method

One of the most recent metaheuristic algorithms is the Charged System Search (CSS) presented in Chap. 3, which uses the Coulomb and Gauss laws from physics and Newtonian laws from mechanics to guide the Charged Particles (CPs) to explore the locations of the optimum [3].

In this chapter, an improved CSS algorithm which is called Magnetic Charged System Search (MCSS) is presented. The new algorithm utilizes the governing laws for magnetic forces, and includes magnetic forces in addition to electrical forces. The movements of CPs due to the total force (Lorentz force) are determined using Newtonian mechanical laws.

A. Kaveh, *Advances in Metaheuristic Algorithms for Optimal Design of Structures*,
DOI 10.1007/978-3-319-05549-7_4, © Springer International Publishing Switzerland 2014

4.2.1 Magnetic Laws

4.2.1.1 Magnetic Fields

There is a relation between electric and magnetic forces and these forces are called electromagnetic forces. The region surrounding any stationary or moving charged particle contains electric fields. In addition to electric field, the region surrounding any moving charged particle also contains magnetic fields. The existence of the magnetic field near the moving charged particles was Oersted's discovery in 1819. He has shown that a compass needle deflected by a current-carrying conductor. Shortly after this discovery, Biot and Savart proposed a mathematical expression so-called Biot-Savar law that provides the magnitude of magnetic field at any point of the space in terms of the electric current that produces the field, Fig. 4.1. Biot-Savar law is expressed [4] as:

$$d\mathbf{B} = \frac{\mu_0}{4\pi} \frac{Id\mathbf{s} \times \hat{\mathbf{r}}}{r^2} \tag{4.1}$$

Here, $d\mathbf{B}$ is the magnetic field at point P and μ_0 is a constant called the permeability of free space, and r is the distance between $d\mathbf{s}$ to P.

Consider a straight wire with radius of R carrying electric current of magnitude I which is uniformly distributed through the cross-section of the wire, Fig. 4.2a. By utilizing Biot-Savar law, the magnetic field produced by wire at a point like P outside the wire, can be determined as:

$$B = \frac{\mu_0}{2\pi} \frac{I}{r} \quad \text{when } r \geq R \tag{4.2}$$

The magnitude of the magnetic field inside the wire can be obtained using Ampère's law,

$$B = \left(\frac{\mu_0}{2\pi} \frac{I}{R^2} \right) \times r \quad \text{when } r < R \tag{4.3}$$

With this formulation for magnetic field, the magnitude of the field inside the wire increases linearly from $r = 0$ to $r = R$ ($B \propto r$), and outside of the wire, it is inversely proportional to the distance ($B \propto 1/r$), and decreases by increasing the distance. When $r = R$, the (4.2) and (4.3) have an overlap, and both give identical magnitude for the magnetic field. A plot of these two equations from [4] is shown in Fig. 4.2b.

If there are many wires in a space, in order to calculate the total magnitude of the magnetic field in a specified point, the equivalent magnetic field should be calculated by considering the principle of superposition, and summing the magnetic fields produced by each wire. Therefore, the total magnetic field at a specified point P, due to a group of wires, can be obtained as:

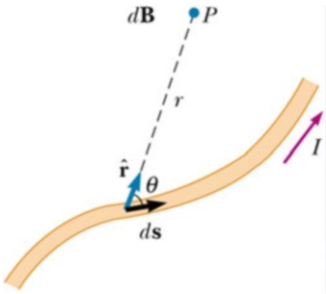

Fig. 4.1 The magnitude of the magnetic field $d\mathbf{B}$ at point P due to current I through a length element $d\mathbf{s}$ given by Biot-Savar law [1]

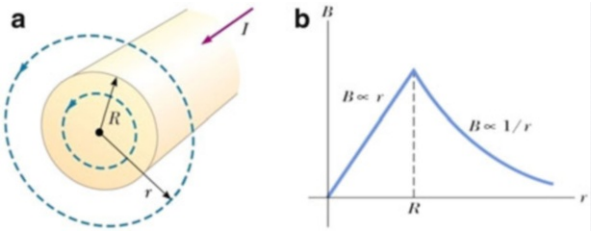

Fig. 4.2 (**a**) A wire carrying electric current I that is uniformly distributed in its cross-section. (**b**) A plot of distribution of magnetic field produced by a wire in the space [1]

$$B_P = \sum_{i=1}^{n} B_{ip} \tag{4.4}$$

where B_P is the total magnetic field at point P, n is the number of wires in the space, and B_{ip} is the magnetic field created by the ith wire at point P which can be expressed as:

$$B_{ip} = \begin{cases} \dfrac{\mu_0}{2\pi}\dfrac{I}{r} & for \;\; r \geq R \\ \left(\dfrac{\mu_0}{2\pi}\dfrac{I}{R^2}\right) \times r & for \;\; r < R \end{cases} \tag{4.5}$$

4.2.1.2 Magnetic Forces

When a charged particle moves in a magnetic field, a magnetic force $\mathbf{F}_B$ will be imposed on that moving charged particle. Experiments on charged particles moving in a magnetic field results in the following:

- The magnitude of the magnetic force $\mathbf{F}_B$ exerted on the charged particle is proportional to the charge q and to the speed v of the particle.
- The magnitude and direction of the magnetic force $\mathbf{F}_B$ depend on the velocity of the particle and magnitude and direction of magnetic field $\mathbf{B}$.

By summarizing these observations, an expression for calculating the magnetic force is obtained [4] as:

$$\mathbf{F}_B = q\mathbf{v} \times \mathbf{B} \tag{4.6}$$

where **B** is the magnetic field exerted on the particle. Here, the only source of the magnetic field is the magnetic field produced by the wires. Thus, the magnitude of the **B** can be calculated using (4.5).

4.2.2 A Brief Introduction to Charged System Search Algorithm

The Charged system search (CSS) algorithm, as explained in Chap. 3, takes its inspiration from the physic laws governing a group of charged particles, CPs. These charge particles are sources of the electric fields, and each CP can exert electric force on other CPs. Using the Newtonian mechanic laws, the movement of each CP due to the electric force can be determined. The CSS algorithm is summarized in a step-by-step form as follows:

Step 1. Initialization

The initial positions of the CPs are randomly determined using a uniform source, and the initial velocities of the particles are set to zero. A memory is used to save a number of best results. This memory is called the Charged Memory (CM).

Step 2. Determination of electric forces and the corresponding movements

- Force Determination. Each charged particle imposes electric forces on the other CPs according to the magnitude of its charge. The charge of the each CP is:

$$q_i = \frac{fit(i) - fitworst}{fitbest - fitworst} \tag{4.7}$$

where fit(i) is the objective function value of the ith CP, *fitbest* and *fitworst* are the so far best and worst fitness among all of the CPs, respectively.

In addition to electric charge, the magnitude of the electric forces exerted on the CPs is depended on their separation distance that is,

$$r_{ij} = \frac{||\mathbf{X}_i - \mathbf{X}_j||}{||(\mathbf{X}_i + \mathbf{X}_j)/2 - \mathbf{X}_{best}|| + \varepsilon} \tag{4.8}$$

where $\mathbf{X}_i$ and $\mathbf{X}_j$ are the position of the ith and jth CPs, and r_{ij} is the separation distance these CPs. $\mathbf{X}_{best}$ is the position of the best current CP, and ε is a small positive number to prevent singularity.

The probability of the attraction of the *i*th CP by the *j*th CP is expressed as:

$$p_{ij} = \begin{cases} 1 \Leftrightarrow \dfrac{fit(i) - fitbest}{fit(j) - fit(i)} > rand, or, fit(j) > fit(i) \\ 0 \Leftrightarrow else. \end{cases} \tag{4.9}$$

The electric resultant force $\mathbf{F}_{E,j}$, acting on the *j*th can be calculated by the following equation:

$$\mathbf{F}_{E,j} = q_j \sum_{i,i \neq j} \left(\frac{q_i}{a^3} r_{ij} \cdot w_1 + \frac{q_i}{{r_{ij}}^2} \cdot w_2 \right) \cdot p_{ji} \cdot \left(\mathbf{X}_i - \mathbf{X}_j \right), \quad \begin{cases} w_1 = 1, w_2 = 0 \Leftrightarrow r_{ij} < R \\ w_1 = 0, w_2 = 1 \Leftrightarrow r_{ij} \geq R \\ j = 1, 2, \ldots, N \end{cases} \tag{4.10}$$

- Movements Calculations. According to the determined forces, each CP moves to its new position, and attain velocity as:

$$\mathbf{X}_{j,new} = rand_{j1} \cdot k_a \cdot \frac{\mathbf{F}_j}{m_j} \cdot \Delta t^2 + rand_{j2} \cdot k_v \cdot \mathbf{V}_{j,old} \cdot \Delta \mathrm{t} + \mathbf{X}_{j,old}, \tag{4.11}$$

$$\mathbf{V}_{j,new} = \frac{\mathbf{X}_{j,new} - \mathbf{X}_{j,old}}{\Delta \mathrm{t}} \tag{4.12}$$

where $rand_{j1}$ and $rand_{j2}$ are two random numbers that uniformly distributed in the range (0, 1). k_a is the acceleration coefficient, k_v is the velocity coefficient, and m_j is the mass of particle that is considered equal to q_j. The magnitudes of the k_a and k_v are set to 0.5 which are linearly increased and decreased as:

$$k_a = 0.5(1 + iter/iter_{\max}), \quad k_v = 0.5(1 - iter/iter_{\max}) \tag{4.13}$$

where *iter* is the current iteration number, and $iter_{max}$ is the maximum number of iterations.

Step 3. Charged Memory (CM) Updating

If among all of the new CPs, there are better CP or CPs that have better objective function value than the worst ones in the CM, these should be included in the CM, and the worst ones in the CM are excluded from the CM.

Step 4. Checking the Termination Criteria

Steps 2 and 3 are reiterated until one of the specified terminating criteria is satisfied.

4.2.3 Magnetic Charged System Search Algorithm

4.2.3.1 Combination of Magnetic and Electric forces

The inspiration of the standard CSS algorithm is based on a group of charged particles that exert electric forces on each other based on their charges and their separation distances. After computing the electric forces, each particle moves and its movement is calculated by using Newtonian mechanics laws. Therefore, we have charged particles that move in the search space. In physics, it has been shown that when a charged particle moves, it produces magnetic field. This magnetic field can exert a magnetic force on other charged particles. Thus, in addition to the electric forces we should consider magnetic forces. In physics, when a charged particle moves with velocity $\mathbf{v}$ in the presence of both an electric field $\mathbf{E}$ and a magnetic field $\mathbf{B}$, experiences both an electric force $q\mathbf{E}$ and a magnetic force $q\mathbf{v} \times \mathbf{B}$. The total force, known as the Lorentz force [4], exerting on the charged particle is:

$$\sum \mathbf{F} = \mathbf{F}_B + \mathbf{F}_E = q\mathbf{v} \times \mathbf{B} + q\mathbf{E} = q \cdot (\mathbf{v} \times \mathbf{B} + \mathbf{E}) \tag{4.14}$$

Where $\mathbf{F}$ is the Lorentz force. Thus, MCSS, considers the magnetic force as an additional force with the purpose of making the new algorithm closer to the nature of the movement of charged particles. From optimization point of view, this new force records additional information about the movement of the CPs, and it improves the performance of the standard CSS.

4.2.3.2 MCSS Algorithm

The MCSS algorithm is based on its original version, standard CSS. The difference between these two algorithms is that CSS only considers the electric force, but MCSS includes magnetic forces besides electric forces. The main structure of the algorithm is the same as the standard CSS, but in MCSS changes are made in part of the algorithm where the forces are computed. By using the aforementioned physical laws about magnetic fields and forces, the magnetic forces are determined. Each solution candidate $\mathbf{X}_i$ known as CP (charged particle) contains electrical charge. These CPs produce electric fields, and exert electric forces on each other. When a CP moves, it creates a magnetic field in the space, and this magnetic field imposes magnetic forces on other CPs.

As explained previously, the source of the magnetic fields is the movement of the CPs. For computing these fields, we assumed that CPs move in virtual straight wires with radius of R. Thus, the path of movement of each particle consists of straight wires. These straight wires change their directions by each movement of the CPs, but during the movement, each wire remains straight, Fig. 4.3. The places that a wire changes its direction, is the position of the CP at the end of its movement.

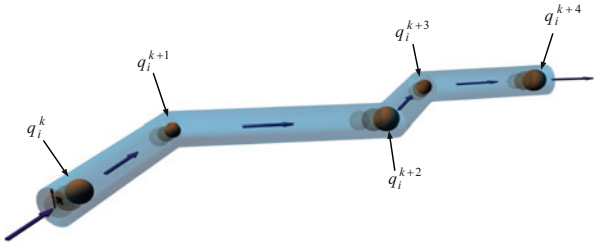

Fig. 4.3 The schematic view of a virtual wire (movement path of a CP), q_i^k is the charge of the ith CP at end of the kth movement (kth iteration) [1]

When the CP starts a new movement, the direction of its movement may differ from its previous one, so the direction of the wire which includes the CP during its movement also changes. According to magnetic laws presented in Sect. 4.2.1, a conducting wire carrying electric current can creates magnetic fields in the space. Now our virtual wires contain charged particles that move on them. By each movement of the CPs, their charges are altered, so during the movement the magnitude of the charge is not constant, and changes during the movement. This movement of CPs can be comprehended as an electric current in the virtual wire. The current of a wire is the rate at which charge flows through one specified cross-section of the wire. If Δq is the amount of charge that passes through this area in a time interval Δt, the average current I_{avg} will be equal to the charge that passes through the cross-section per unit time:

$$I_{avg} = \frac{\Delta q}{\Delta q} \tag{4.15}$$

Since the time intervals of each movement are set to unity, the average current will be equal to the variation of the charge. For computing the variation of the charges, we consider the start and the end points of the movement of CPs. By taking these assumptions into account, (4.15) can be written as:

$$\left(I_{avg}\right)_{ik} = q_i^k - q_i^{k-1} \tag{4.16}$$

where $(I_{avg})_{ik}$ is the average current in the ith wire of ith CP in the kth movement (iteration), and q_i^{k-1} and q_i^k are the charges of the ith CP at the start and end of its kth movement, respectively. Equation (4.16) shows that by this definition for the electric current, the concept of quantity represents the variation of the objective function of each CP in each movement. By this definition, the electric current can be both positive and negative values. A positive one indicates that the movement produced an improvement in the charge of the CP. In other words, since the charge of a CP is a quantity of its quality or objective function value, a positive electric current means an improvement and a negative electric current means an deterioration in the quality of the CP.

Charge of the CPs is defined by (4.7). This expression for computing electric charges results in values between 0 to 1. This is due to normalization of the

objective function of each CP in that expression. Therefore, the charges of the worst and best CP are always zero and unity, respectively. Now, consider the situation that the worst CP moves in the search space, at the end of the movement, it may attain a better objective function value, but it may still be the worst CP, so its charge will still be zero. This means that there may be some situations that the objective function of a CP improves but its charge does not change because charge is a relative quantity. It seems necessary to modify the electric current expression in a way that the concept of electric current is saved, and the aforementioned problem is solved. In relation with this problem, two alternative expressions for computing electric current are proposed. The first one is:

$$\left(I_{avg}\right)_{ik} = \frac{q_{i,k} - q_{i,k-1}}{q_{i,k}} \tag{4.17}$$

Where $q_{i,k}$ and $q_{i,k\ -\ 1}$ are the charge of the ith CP at the start of the kth and $k - 1$th iterations, respectively. This equation gives a normalized value for the variation of the ith CP. The second proposed relation is expressed as:

$$\left(I_{avg}\right)_{ik} = sign\left(df_{i,k}\right) \times \frac{\left|df_{i,k}\right| - df_{\min,k}}{df_{\max,k} - df_{\min,k}} \tag{4.18}$$

$$df_{i,k} = fit_k(i) - fit_{k-1}(i) \tag{4.19}$$

where $df_{i,k}$ is the variation of the objective function in the kth movement (iteration). $fit_k(i)$ and $fit_{k\ -\ 1}(i)$ are the values of the objective function of the ith CP at the start of the kth and $k - 1$th iterations, respectively. The quantity $df_{i,k}$ can attain both positive and negative values. If we consider absolute values of df for all of the current CPs, $df_{\max,k}$ and $df_{\min,k}$ will be the maximum and minimum values among these absolute values of df, respectively. Therefore, $df_{\max,k}$ and $df_{\min,k}$ are always positive quantities. It should be noted that here the second expression (4.18) and (4.19) is utilized for the computation of the electric current.

For computing the magnetic field in place of each particle, one must compute the distance of that particle from the virtual wire. This distance is assumed to be the same as (4.8). Thus, r_{ij} now means the distance between the ith wire and ith virtual CP to the jth charged particle.

In the expression for computing the magnetic force, (4.6), we should consider the velocity of the movement of CPs. In this case, due to the movements of both CPs (CP in the virtual wire and CP in the space) the relative velocity, $\mathbf{v}_{rel}$, is considered as:

$$\mathbf{v}_{rel} = \frac{\mathbf{X}_i - \mathbf{X}_j}{\Delta \mathrm{t}} \tag{4.20}$$

where $\mathbf{X}_i$ and $\mathbf{X}_j$ are the positions of the ith and jth CPs, the Δt is the time step that is set to unity. Therefore the relative velocity can be rewritten as:

$$\mathbf{v}_{rel} = \mathbf{X}_i - \mathbf{X}_j \tag{4.21}$$

By considering these assumptions, the magnetic force $\mathbf{F}_{\mathbf{B},ji}$ exerted on the jth CP due to the magnetic field produced by the ith virtual wire (ith CP) can be expressed as:

$$\mathbf{F}_{\mathbf{B},ji} = q_j \cdot \left(\frac{I_i}{R^2} r_{ij} \cdot z_1 + \frac{I_i}{r_{ij}} \cdot z_2\right) \cdot pm_{ji} \cdot \left(\mathbf{X}_i - \mathbf{X}_j\right), \quad \begin{cases} z_1 = 1, z_2 = 0 \Leftrightarrow r_{ij} < R \\ z_1 = 0, z_2 = 1 \Leftrightarrow r_{ij} \geq R \end{cases} \tag{4.22}$$

where q_i is the charge of the ith CP, R is the radius of the virtual wires, I_i is the average electric current in each wire, and pm_{ji} is the probability of the magnetic influence (attracting or repelling) of the ith wire (CP) on the jth CP. This term can be computed by the following expression:

$$pm_{ji} = \begin{cases} 1 \Leftrightarrow fit(i) > fit(j) \\ 0 \Leftrightarrow else \end{cases} \tag{4.23}$$

where *fit(i)* and *fit(j)* are the objective values of the ith and jth CP, respectively. This probability determines that only a good CP can affect a bad CP by the magnetic force. This magnetic probability is slightly different from the electric probability expressed by (4.9). The electric probability considers a chance for both good and bad CPs to attract each other, but the magnetic probability has allocated this chance only to good CPs. The purpose of this definition of magnetic probability is to reduce the parasite magnetic fields and reinforce the efficiency of the magnetic forces.

Investigating different terms of the magnetic force shows how this force can help the standard CSS algorithm. If I_i, electric current in virtual ith virtual wire is negative, according to the concept of the electric current, a negative value means that the ith CP did not experienced an improvement in the value of its objective function. Thus, a negative value will be multiplied by $(\mathbf{X}_i - \mathbf{X}_j)$, so this produces a repelling force. In this case, it is an ideal force. On the other hand, if the ith CP experiences an improvement in its movement, it will attract the jth CP. From optimization point of view, this kind of force can help the algorithm. It stores and applies the information of the movement of each CP. This information is lost in the standard CSS, but MCSS utilizes this information and increases the efficiency of algorithm.

Now by considering the group of the charged particles, the resultant magnetic force acting on each CP can be calculated using the following expression:

$$\mathbf{F}_{\mathbf{B},j} = q_j \cdot \sum_{i,i\neq j} \left(\frac{I_i}{R^2} r_{ij} \cdot z_1 + \frac{I_i}{r_{ij}} \cdot z_2 \right) \cdot pm_{ji} \cdot \left(\mathbf{X}_i - \mathbf{X}_j \right), \quad \begin{cases} z_1 = 1, z_2 = 0 \Leftrightarrow r_{ij} < R \\ z_1 = 0, z_2 = 1 \Leftrightarrow r_{ij} \geq R \\ j = 1, 2, \ldots, N \end{cases} \tag{4.24}$$

where $\mathbf{F}_{\mathbf{B},j}$ is the resultant magnetic force exerted on the jth charged particle.

The quantity R is the radius of the virtual wires, and if a charged particle places outside or inside of a virtual wire, the magnetic force that exerted on it is computed differently. With this formulation for magnetic force, in the early iterations where the agents are far from each other, their distances will be large values, and the magnetic force in this case will be inversely proportional to the distances. As a result, the magnitude of the magnetic force is relatively small, and this feature of the algorithm provides a good situation for search ability of the CPs in the early iterations which is ideal for optimization problems. After a number of iterations, CPs search the search space and most of them will be gathered in a small space. Now, the distances between CPs are decreased and a local search starts. In this case, if the magnetic force computed based on the inverse relation between distances, the magnitude of the forces will be increased due to decrease of the distances. These large forces may prevent the convergence of the algorithm in the local search. One of the solutions that can be proposed is that when the distances are relatively small, the magnetic force should be computed using the linear formulation of magnetic fields (4.3). This means that the formulation of the magnetic force for global and local phases should be separated, (4.24). A suitable value for R in (4.24) can be unity. However, by more investigating in the magnetic force formulation, it could be understood that the aforementioned problem can be solved automatically. If the value of the R is taken as zero, all of the magnetic fields produced by virtual wires can be calculated based on (4.2) Using this equation for small distances gives large values for the magnetic field, but when the values of distances are small, it means that the CPs are collected in a small space and their movements are small (Local Search). Thus, both $\mathbf{X}_i - \mathbf{X}_j$ and I_i are small values. By considering (4.24) for calculating the magnetic forces, it can be noted that a large value is multiplied by two small values, so the final value (magnetic force) is a normal value which helps the algorithm. Due to the ease of implementation, and better convergence rate the second solution is selected in this part and the magnetic force is revised in (4.25).

The term pm_{ji}, in the expression for calculating the magnetic force, provides competition ability for the CPs. According to the concept of the magnetic force in this algorithm, when a CP experience an improvement in its value of the objective function, should attract another CPs, regardless to its previous and current charge. However, by considering the term pm_{ji}, CPs with larger charges have more tendency to attract other CPs. The reason is that by considering this term, the redundant and parasite magnetic fields made by bad CPs are eliminated and it helps the efficiency of the algorithm.

It should be noted that in implementing the MCSS, the part of CSS algorithm related to computing forces should be changed. Both magnetic and electric forces should be computed, and superposed. The Lorentz force (total force) will be expressed as:

$$\sum \mathbf{F}_j = \mathbf{F}_{B,j} + \mathbf{F}_{E,j} = q_j \sum_{i,\, i \neq j} \left(\frac{I_i}{r_{ij}} \cdot pm_{ji} + \left(\frac{q_i}{a^3} r_{ij} \cdot w_1 + \frac{q_i}{{r_{ij}}^2} \cdot w_2 \right) \cdot p_{ji} \right) \cdot \left(\mathbf{X}_i - \mathbf{X}_j \right), \quad \begin{cases} w_1 = 1, w_2 = 0 \Leftrightarrow r_{ij} < R \\ w_1 = 0, w_2 = 1 \Leftrightarrow r_{ij} \geq R \\ j = 1, 2, \ldots, N \end{cases} \tag{4.25}$$

where $\mathbf{F}_j$ is the resultant Lorentz force (total force) acting on the jth CP.

Consider the ith CP among all of the CPs; this CP has a charge which is larger than a number of other CPs charge. Considering the rules of the CSS, the ith CP attracts all other CPs that have smaller charges. After computing the electric forces, all of the CPs move around the search space. Now, the ith CPs also moved to a new position. In this movement, the ith particle may experience deterioration in its objective function value. Due to this decrease, the new charge of the ith particle will be decreased, but its charge may still be larger than a number of CPs. According to the CSS algorithm, the ith particle still attracts all other CPs with smaller charges regardless of the failure of the ith CP in its last movement. From one perspective, this is logical that a good CP can attract bad CPs. This feature ensures the competition ability of the algorithm. However, from another point of view, if no attention is paid to the success or failure of the CPs in their last movement, a lot of useful information in optimization process will be lost. Thus, in the MCSS algorithm, magnetic forces are included to prevent the loss of this kind of information which benefits the algorithm. By this concept, the ith particle which has experienced a failure in its last movement, exerts repelling magnetic forces on the other CPs. In this situation, the direction of the magnetic forces and electrical ones that are acted on CPs by the ith CP is opposite.

That was a special case that the magnetic and electric forces were against each other. Most of the times, the magnetic and electric forces are in the same direction and they reinforce the effect of each other. Consequently, the exploitation ability of the algorithm is mostly reinforced. Because of this increase in exploitation ability, we can slightly modify k_v in (4.14) to increase the exploration ability of the algorithm. In fact, the MCSS algorithm guides the CPs with more information and the efficiency of the algorithm including a fast convergence is improved, and in comparison to the standard CSS, a better exploitation and exploration are provided.

4.2.4 Numerical Examples

In order to ensure the efficient performance of the MCSS algorithm, some numerical examples are solved and the results are compared to those of the standard CSS algorithm. The examples consist of 18 mathematical functions. The numerical examples are presented in Sect. 5.2.4.1. In Sect. 5.2.4.2 the results of the MCSS are presented and compared to those of the CSS and other optimization algorithms in the literature. Finally, in Sect. 5.2.5 three well-studied engineering design problems are solved by MCSS and the results are compared to those of the CSS.

4.2.4.1 Mathematical Benchmark Functions

Comparison Between MCSS, CSS and a Set of Genetic Algorithms

In this section, some mathematical benchmarks are chosen from [5], and optimized using the MCSS algorithm. The description of these mathematical benchmarks is provided in Table 4.1.

Numerical Results

In this section, the numerical results of optimization for the mathematical benchmarks are presented. In this investigation, some parameters of the algorithm such as, HMCR, PAR, CM size (CMS), the number of CPs, and the maximum number of iteration are modified. For eliminating the effect of such parameters in studying the performance of the algorithm, these parameters are considered the same as those of [6]. It should be noted that the number of CPs is set to 20, and the maximum number of iterations is considered as 200 for both CSS and MCSS algorithm. In Table 4.2, the results of the MCSS are compared to the results obtained by the CSS from [6], and GA and some of its variants derived from [5]. For a fair comparison between MCSS and CSS, the random initial solutions of each runs are the same. The numbers in Table 4.2 indicate the average number of function evaluation from 50 independent runs. The numbers in parenthesis, demonstrate the fraction of the unsuccessful to successful runs. The absence of a parenthesis means that the algorithm was successful in all of the runs. Each run of the algorithm is successful when that run determines a local minimum with predefined accuracy, i.e., $\varepsilon = |f_{\min} - f_{final}| = 10^{-4}$. The results verify the efficiency of the MCSS algorithm compared to the CSS and other Genetic algorithms. The existence of the magnetic forces in the MCSS provides a better exploration and exploitation for the algorithm. Thus, the convergence is speeded up. One of the important features of the MCSS algorithm is its ability to converge to the desired optimum with a few number of CPs and a small value for maximum number of iterations. The difference between the CSS algorithm and MCSS algorithm becomes more obvious when the number

Table 4.1 Description of the mathematical benchmarks

Function name	Interval	Function	Global minimum
Aluffi-pentiny	$\mathbf{X} \in [-10, \quad 10]^2$	$f(\mathbf{X}) = \frac{1}{4}{x_1}^4 - \frac{1}{2}{x_1}^2 + \frac{1}{10}x_1 + \frac{1}{2}{x_2}^2$	−0.352386
Bohachevsky 1	$\mathbf{X} \in [-100, \quad 100]^2$	$f(\mathbf{X}) = {x_1}^2 + 2{x_2}^2 - \frac{3}{10}\cos(3\pi x_1) - \frac{4}{10}\cos(4\pi x_2) + \frac{7}{10}$	0.0
Bohachevsky 2	$\mathbf{X} \in [-50, \quad 50]^2$	$f(\mathbf{X}) = {x_1}^2 + 2{x_2}^2 - \frac{3}{10}\cos(3\pi x_1) \cdot \cos(4\pi x_2) + \frac{3}{10}$	0.0
Becker and Lago	$\mathbf{X} \in [-10, \quad 10]^2$	$f(\mathbf{X}) = (\lvert x_1 \rvert - 5)^2 + (\lvert x_2 \rvert - 5)^2$	0.0
Branin	$0 \le x_2 \le 15$, $-5 \le x_1 \le 10$	$f(\mathbf{X}) = \left(x_2 - \frac{5.1}{4\pi^2}{x_1}^2 + \frac{5}{\pi}x_1\right)^2 + 10 \cdot \left(1 - \frac{1}{8\pi}\right)\cos(x_1) + 10$	0.397887
Camel	$\mathbf{X} \in [-5, \quad 5]^2$	$f(\mathbf{X}) = 4{x_1}^2 - 2.1{x_1}^4 + \frac{1}{3}{x_1}^6 + x_1x_2 - 4{x_2}^2 + 4{x_2}^4$	−1.0316
Cb3	$\mathbf{X} \in [-5, \quad 5]^2$	$f(\mathbf{X}) = 2{x_1}^2 - 1.05{x_1}^4 + \frac{1}{6}{x_1}^6 + x_1x_2 + {x_2}^2$	0.0
Cosine Mixture	$\mathbf{X} \in [-1, \quad 1]^n, n = 4$	$f(\mathbf{X}) = \sum_{i=1}^{n} {x_i}^2 - \frac{1}{10}\sum_{i=1}^{n}\cos(5\pi x_i)$	−0.4
DeJoung	$\mathbf{X} \in [-5.12, \quad 5.12]^3$	$f(\mathbf{X}) = {x_1}^2 + {x_2}^2 + {x_3}^2$	0.0
Exponential	$\mathbf{X} \in [-1, \quad 1]^n$, $n = 2, 4, 8$	$f(\mathbf{X}) = -\exp\left(-0.5 \cdot \sum_{i=1}^{n} {x_i}^2\right)$	−1.0
Goldstein and price	$\mathbf{X} \in [-2, \quad 2]^2$	$f(\mathbf{X}) = \left[1 + (x_1 + x_2 + 1)^2 \cdot (19 - 14x_1 + 3{x_1}^2 - 14x_2 + 6x_1x_2 + 3{x_2}^2)\right] \times \left[30 + (2x_1 - 3x_2)^2 \cdot (18 - 32x_1 - 12{x_1} + 48x_2 - 36x_1x_2 + 27{x_2}^2)\right]$	3.0
Griewank	$\mathbf{X} \in [-100, \quad 100]^2$	$f(\mathbf{X}) = 1 + \frac{1}{200}\sum_{i=1}^{2} {x_i}^2 - \prod_{i=1}^{2}\cos\left(\frac{x_i}{\sqrt{i}}\right)$	0.0
Hartman 3	$\mathbf{X} \in [-30, \quad 30]^3$	$f(\mathbf{X}) = -\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{3} a_{ij}(x_j - p_{ij})^2\right), a = \begin{bmatrix} 3 & 10 & 30 \\ 0.1 & 10 & 35 \\ 3 & 10 & 30 \\ 0.1 & 10 & 35 \end{bmatrix}, c = \begin{bmatrix} 1 \\ 1.2 \\ 3 \\ 3.2 \end{bmatrix}$	−3.862782

(continued)

Table 4.1 (continued)

Function name	Interval	Function	Global minimum
		$p = \begin{bmatrix} 0.3689 & .117 & 0.2673 \\ 0.4699 & 0.4387 & 0.747 \\ 0.1091 & 0.8732 & 0.5547 \\ 0.03815 & 0.5743 & 0.8828 \end{bmatrix}$	
Hartman 6	$\mathbf{X} \in [0, \quad 1]^6$	$f(\mathbf{X}) = -\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{4} a_{ij}\left(x_j - p_{ij}\right)^2\right), a = \begin{bmatrix} 10 & 3 & 17 & 3.5 & 1.7 & 8 \\ 0.05 & 10 & 17 & 0.1 & 8 & 14 \\ 3 & 3.5 & 1.7 & 10 & 17 & 8 \\ 17 & 8 & 0.05 & 10 & 0.1 & 14 \end{bmatrix}$	-3.322368
		$c = \begin{bmatrix} 1 \\ 1.2 \\ 3 \\ 3.2 \end{bmatrix}$, $p = \begin{bmatrix} 0.1312 & 0.1696 & 0.5569 & 0.0124 & 0.8283 & 0.5886 \\ 0.2329 & 0.4135 & 0.8307 & 0.3736 & 0.1004 & 0.9991 \\ 0.2348 & 0.1451 & 0.3522 & 0.2883 & 0.3047 & 0.6650 \\ 0.4047 & 0.8828 & 0.8732 & 0.5743 & 0.1091 & 0.0381 \end{bmatrix}$	
Rastrigin	$\mathbf{X} \in [-1, \quad 1]^2$	$f(\mathbf{X}) = \sum_{i=1}^{2} \left({x_i}^2 - \cos(18x_i)\right)$	-2.0
Rosenbrock	$\mathbf{X} \in [-30, \quad 30]^n, n = 2$	$f(\mathbf{X}) = \sum_{i=1}^{n-1} 100\left(x_{i+1} - {x_i}^2\right)^2 + (x_i - 1)^2$	0.0

Table 4.2 Performance comparison for the benchmark problems

Function	GEN	GEN-S	GEN-S-M	GEN-S-M-LS	CSS [6]	MCSS [1]
AP	1,360(0.99)	1,360	1,277	1,253	804	437
Bf1	3,992	3,356	1,640	1,615	1,187	542
Bf2	20,234	3,373	1,676	1,636	742	556
BL	19,596	2,412	2,439	1,436	423	481
Branin	1,442	1,418	1,404	1,257	852	351
Camel	1,358	1,358	1,336	1,300	575	384
Cb3	9,771	2,045	1,163	1,118	436	288
CM	2,105	2,105	1,743	1,539	1,563	538
Dejoung	9,900	3,040	1,462	1,281	630	387
Exp2	938	936	817	807	132	183
Exp4	3,237	3,237	2,054	1,496	867	317
Exp8	3,237	3,237	2,054	1,496	1,426	659
Goldstein and Price	1,478	1,478	1,408	1,325	682	450
Griewank	18,838(0.91)	3,111(0.91)	1,764	1,652(0.99)	1,551	1,272
Hartman3	1,350	1,350	1,332	1,274	860	344
Hartman6	2,562(0.54)	2,562(0.54)	2,530(0.67)	1,865(0.68)	1,783	908
Rastrigin	1,533(0.97)	1,523(0.97)	1,392	1,381	1,402	1,252
Rosenbrock	9,380	3,739	1,675	1,462	1,452	1,424
Total	112,311(96.7)	41,640(96.7)	29,166(98.16)	25,193(98.16)	17,367	10,773

of CPs and the number of iterations are set to small values. Thus, another comparison is performed to show the difference between the CSS and MCSS algorithm in unsuitable situations, i.e., small number of CPs and maximum number of permitted iterations. Therefore, the number of CPs is set to 10 and the maximum number of permitted iterations is considered as 100. This means that the computational cost is one quarter of the previous comparison. The results of this comparison are presented in Table 4.3. The numbers in the Table 4.3 are the optimum found by each algorithm. These are the average of 100 independent runs. The accuracy of the solutions in some cases may be unsatisfactory, but it should be noted that the number of CPs and maximum number of iterations are small. The purpose of this comparison is to magnify the difference between the CSS and MCSS algorithm, and verify the better performance of the MCSS in this situation. For more detailed presentation, Fig. 4.4 illustrates the optimization process and convergence.

Statistical Test

Now in the following we want to ensure that the results of MCSS in Table 4.3 are better than CSS algorithm. For this purpose, we apply a multi-problem analysis using statistical tests. We apply the test on the obtained errors by each algorithm. If we have the normality condition for our sample of results, a parametric pair t-test

Table 4.3 Numerical comparison of CSS and MCSS algorithms

Function	Global minimum	CSS	MCSS	CSS's error	MCSS's error
AP	−0.352386	−0.198721	−0.308396	0.153665	0.04399
Bf1	0.0	28.809183	0.088327	28.80918	0.088327
Bf2	0.0	8.938997	0.034876	8.938997	0.034876
BL	0.0	0.106252	6.781E-05	0.106252	6.78E-05
Branin	0.397887	3.960884	0.537231	3.562997	0.139344
Camel	−1.0316	−0.866765	−1.031591	0.164835	9E-06
Cb3	0.0	0.125161	6.517E-05	0.125161	6.52E-05
CM	−0.4	−0.230142	−0.352661	0.169858	0.047339
Dejoung	0.0	0.166451	6.891E-05	0.166451	6.89E-05
Exp2	−1.0	−0.999366	−0.999947	0.000634	5.3E-05
Exp4	−1.0	−0.990884	−0.999818	0.009116	0.000182
Exp8	−1.0	−0.949659	−0.999686	0.050341	0.000314
Goldstein and Price	3.0	15.729613	4.620501	12.72961	1.620501
Griewank	0.0	0.342795	0.105112	0.342795	0.105112
Hartman3	−3.862782	−3.491627	−3.816318	0.371155	0.046464
Hartman6	−3.322368	−2.054548	−3.292364	1.26782	0.030004
Rastrigin	−2.0	−1.875735	−1.917121	0.124265	0.082879
Rosenbrock	0.0	19.476846	3.117751	19.47685	3.117751

Number of CPs = 10, maximum number of iterations = 100

can be suitable. We first analyze a safe usage of parametric tests. We utilized two normality tests including: Kolmogorov-Smirnov, and Shapiro-Wilk test. The *p*-values of the normality tests over the sample results obtained by CSS and MCSS are shown in Table 4.4. If we consider a significance level $\alpha = 0.05$, all of the *p*-values in Table 4.4 will be less than 0.05. Thus the sample results do not follow a normal distribution. The Q–Q plot for sample results is illustrated in Fig. 4.5, and it can be understood that the normality conditions is not satisfied in both CSS and MCSS algorithms. This result was predictable because the sample size (the number of problems) is small. Therefore, a parametric test such as pair *t*-test is not appropriate in this case. Therefore we use Wilcoxcon test that is a non-parametric test for pairwise comparisons. The method of this test is described in [7]. The result of this test can be summarized as:

- The *p*-value obtained by Wilcoxcon test is 0.00. Consequently, the Wilcoxcon test considers a difference between the performance of these two algorithms assuming a significance level $\alpha = 0.05$. Therefore, because of better mean value of the MCSS algorithm results, MCSS outperforms its predecessor, CSS algorithm.

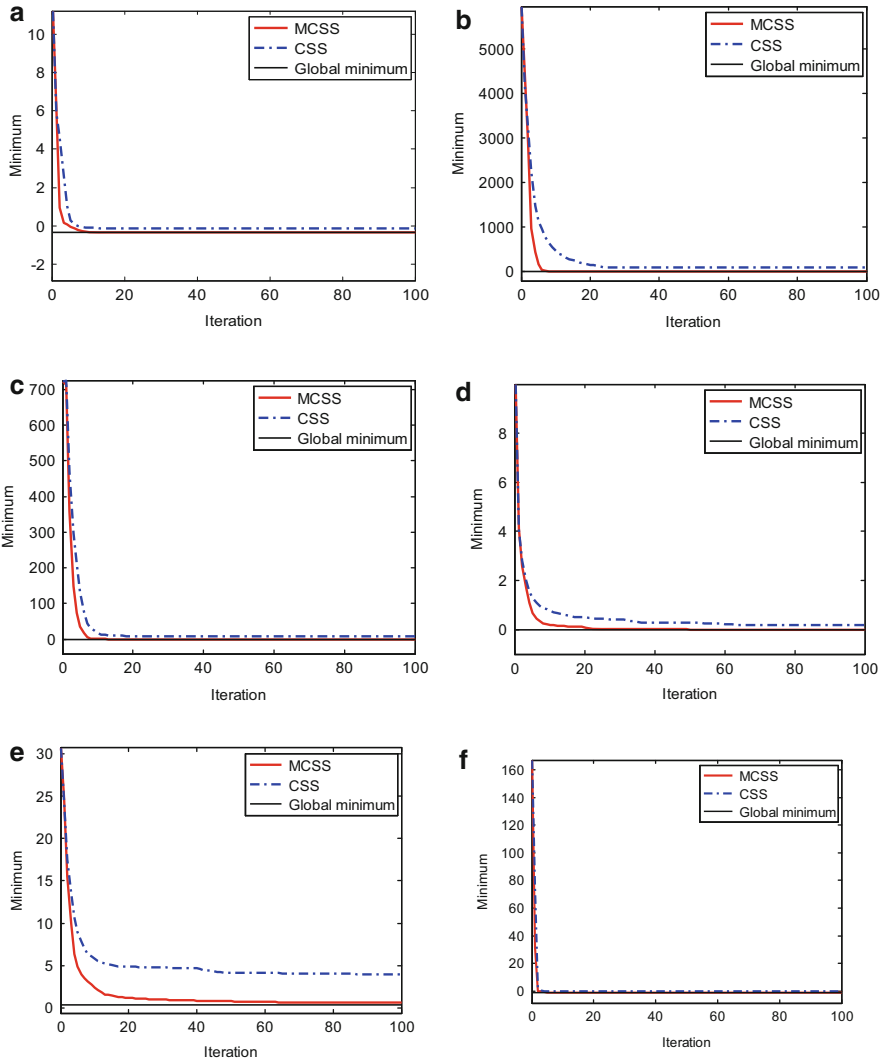

Fig. 4.4 (continued)

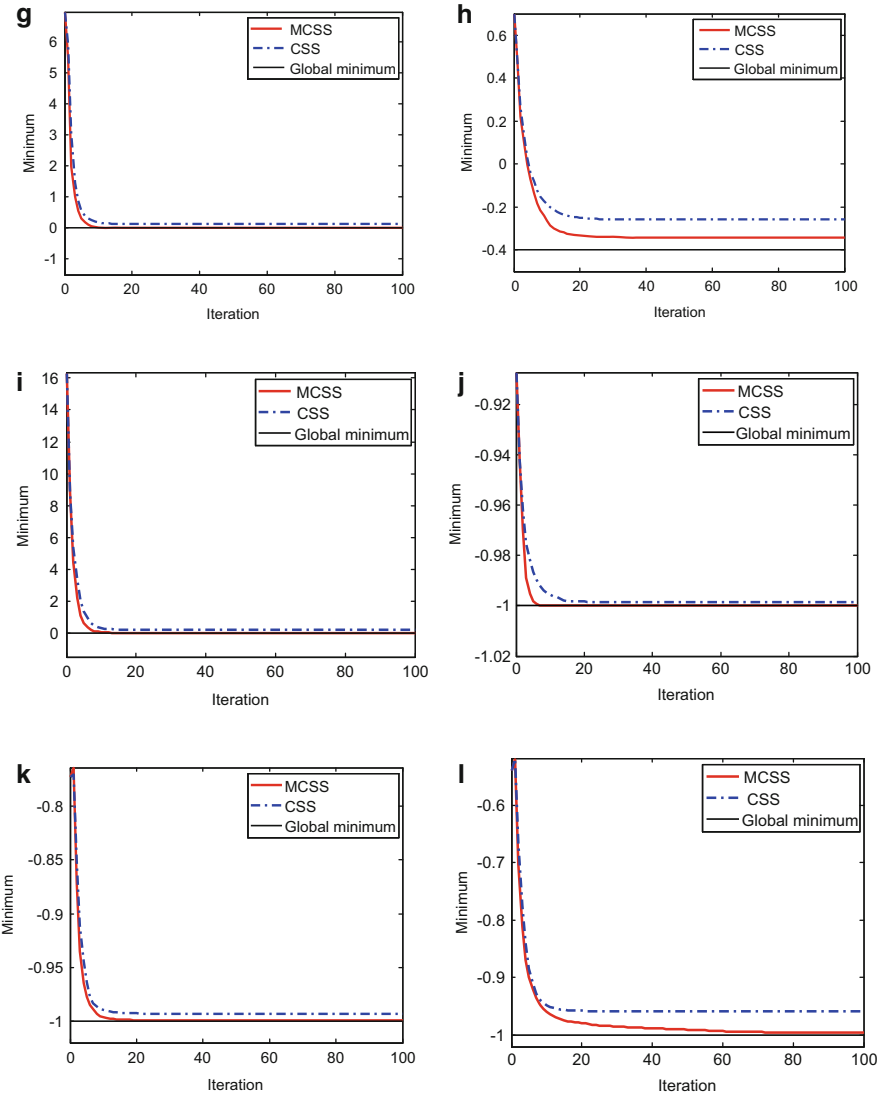

Fig. 4.4 (continued)

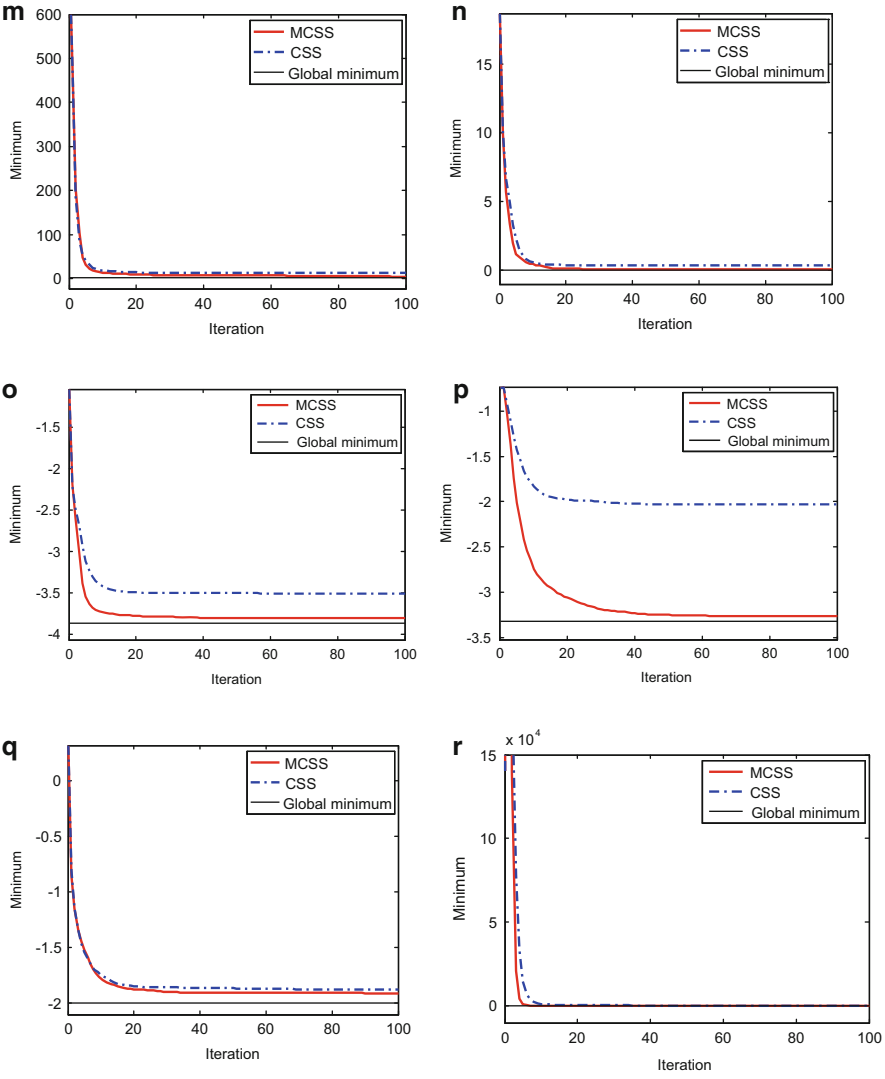

Fig. 4.4 Comparison of the convergence rate of optimizing mathematical benchmarks [1]: (**a**) AP, (**b**) Bf1, (**c**) Bf2, (**d**) BL, (**e**) Branin, (**f**) Camel, (**g**) Cb3, (**h**) CM, (**i**) Dejoung, (**j**) Exp2, (**k**) Exp4, (**l**) Exp8, (**m**) Goldstein and price, (**n**) Griewank, (**o**) Hartman3, (**p**) Hartman6, (**q**) Rastrigin, (**r**) rosenbrock

Table 4.4 Normality tests and their *p*-values over multiple-problem analysis

Algorithm	Kolmogorov-Smirnov	Shapiro-Wilk
CSS	0.00	0.00
MCSS	0.00	0.00

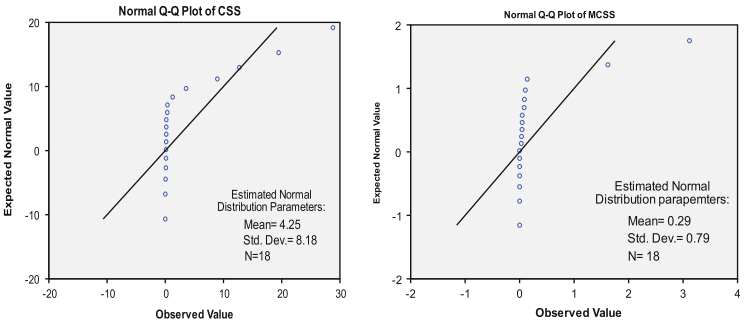

Fig. 4.5 Normal Q–Q plots of the sample results of the CSS and MCSS algorithms [1]

4.2.4.2 Comparison Between MCSS and Other State-of-Art Algorithms

Description of Test Functions and Algorithms

In the following section, the set of test functions designed for Special Session on Real Parameter Optimization organized in the 2005 I.E. Congress on Evolutionary Computation (CEC 2005) are solved by the MCSS algorithm. The detailed description of test functions is presented by Suganthan et al. [8]. The set of these test functions consists of the following functions:

- 5 displaceUnimodals functions (f1–f5)
- Sphere function d.
- Schewefel's problem 1.2 displaced.
- Elliptical function rotated widely conditioned
- Schwefel's problem 1.2 displaced with noise in the fitness.
- Schwefel's problem 2.6 with global optimum in the frontier.
- 20 Multimodals functions (f6–f7)
- 7 basic functions
- Rosenbrock function displaced.
- Griewank function displaced and rotated without frontiers.
- Ackley function displaced and rotated with the global optimum in the frontier.
- Rastrigin function displaced.
- Rastrigin function displaced and rotated.
- Weierstrass function displaced and rotated.
- Schewefel's problem 2.13.
- 2 expanded functions.

- 11 hybrid functions. Each one of these has been defined through compositions of 10 out of 14 previous functions (different in each case).

The characteristics of this experiment is the same as what has been suggested by Suganthan et al. [8]. Each function is solved by MCSS in 25 independent runs, and the average error of the best CP is recorded. The number of CPs is set to 25. The dimension of the test functions is set to 10 (D = 10), and algorithm performs 10,000 function evaluation. The termination criterion is either reaching the maximum number of function evaluation or achieving error less than 10^{-8}. Table 4.5 shows the official results of the participated algorithms obtained from Garcia et al. [9]. The description of each algorithm is given in [19]. The results of the MCSS algorithm are added to Table 4.5. The values of Table 4.5 indicate the average error rate of each algorithm. This value can be considered as a means for measuring the performance of each algorithm.

Numerical Results and Statistical Test

As the results in Table 4.5 show, MCSS has a good performance and its average error rates are good, however, there are some cases that MCSS performs slightly weaker than some other algorithms. For a fair comparison, we have to use statistical test to judge about the performance of MCSS in comparison to other algorithms. We want to find out whether the results of MCSS have a significant difference in comparison to the other algorithms. This analysis is multiple-problem analysis; therefore, a non-parametric test is more suitable in this case. We utilized the Welcoxon's test. This test performs pairwise comparisons between two algorithms. In this test, MCSS is compared to some other remaining algorithms.

Table 4.6 summarizes the results of applying the Wilcoxin test. Table 4.6 includes sum of ranking and *p*-value of each comparison. The method of this test is simply described in [7]. The significance level α is considered as 0.05. In each comparison when the corresponding *p*-value is less than 0.05, it means that two compared algorithms behave differently, and the one with smaller mean value of error rate has a better performance.

The *p*-value in pairwise comparison is independence from another one. If we draw a conclusion involving more than one pairwise comparison in Wilcoxcon's analysis, an accumulated error which is merged up by combination of pairwise comparisons will be obtained. In statistics terms, the Family Wise Error Rate (FWER) will be lost. FWER is defined as the probability of making one or more false discoveries among all the hypotheses when performing multiple pairwise tests (Garcia et al. [9]). The true statistical significance for combining pairwise comparisons is given by:

Table 4.5 Average error rates of the algorithms in CEC'2005 and MCSS algorithm

Function	BLX-GL50	BLX-MA	CoEVO	DE	DMS-L-PSO	EDA	G-CMA-ES	K-PCX	L-CMA-ES	L-SaDE	SPC-PNX	MCSS
f1	1E-09	1E-09	1E-09	1E-09	1E-09	1E-09	1E-09	1E-09	1E-09	1E-09	1E-09	1E-09
f2	1E-09	1E-09	1E-09	1E-09	1E-09	1E-09	1E-09	1E-09	1E-09	1E-09	1E-09	1E-09
f3	570.5	47,710	1E-09	1.94E-06	1E-09	21.21	1E-09	0.415	1E-09	0.01672	1.08E5	0.101
f4	1E-09	2E-08	1E-09	1E-09	0.001885	1E-09	1E-09	7.94E-07	1.76E6	1.42E-05	1E-09	1E-09
f5	1E-09	0.02124	2.133	1E-09	1.14E-06	1E-09	1E-09	48.5	1E-09	0.012	1E-09	1E-09
f6	1E-09	1.49	12.46	0.159	6.89E-08	0.04182	1E-09	0.478	1E-09	1.2E-08	18.91	0.014
f7	0.01172	0.1971	0.03705	0.146	0.04519	0.4205	1E-09	0.231	1E-09	0.02	0.08261	1.E-09
f8	20.35	20.19	20.27	20.4	20	20.34	20	20	20	20	20.99	20
f9	1.154	0.4379	19.19	0.955	1E-09	5.418	0.239	0.119	44.9	1E-09	4.02	0.012
f10	4.975	5.643	26.77	12.5	3.622	5.289	0.0796	0.239	40.8	4.969	7.304	2.152
f11	2.334	4.557	9.029	0.847	4.623	3.944	0.934	6.65	3.65	4.891	1.91	3.823
f12	406.9	74.3	604.6	31.7	2.4001	442.3	29.3	149	209	4.5E-07	259.5	2.503
f13	0.7498	0.7736	1.137	0.977	0.3689	1.841	0.696	0.653	0.494	0.22	0.8379	0.552
f14	2.172	2.03	3.706	3.45	2.36	2.63	3.01	2.35	4.01	2.915	3.046	2.432
f15	400	269.6	293.8	259	4.854	365	228	510	211	32	253.8	153.46
f16	93.49	101.6	177.2	113	94.76	143.9	91.3	95.9	105	101.2	109.6	90.567
f17	109	127	211.8	115	110.1	156.8	123	97.3	549	114.1	119	102.12
f18	420	803.3	901.4	400	760.7	483.2	332	752	497	719.4	439.6	741.73
f19	449	762.8	844.5	420	714.3	564.4	326	751	516	704.9	380	317.27
f20	446	800	862.9	460	822	651.9	300	813	442	713	440	502.31
f21	689.3	721.8	634.9	492	536	484	500	1,050	404	464	680.1	436.61
f22	758.6	670.9	778.9	718	692.4	770.9	729	659	740	734.9	749.3	642.49
f23	638.9	926.7	834.6	572	730.3	640.5	559	1,060	791	664.1	575.9	630.38
f24	200	224	313.8	200	224	200	200	406	865	200	200	194.17
f25	403.6	395.7	257.3	923	365.7	373	374	406	442	375.9	406	356.51
Mean	224.6815	2,144.922	272.4173	189.7254	203.5414	213.4814	152.6623	273.1534	70,635.3	194.261	191.12	167.97

Table 4.6 The Wilcoxcon test results

MCSS *vs.*	R^+	R^-	*p*-Value
BLX-GL50	46	185	0.016
BLX-MA	6	270	0.000
CoEVO	14	239	0.000
DE	59	172	0.050
DMS-L-PSO	57	196	0.024
EDA	20	211	0.314
G-CMA-ES	70	120	0.001
K-PCX	20	233	0.025
L-CMA-ES	45	165	0.048
L-SaDE	65.5	187	0.027
SPC-PNX	52	179	0.001

$$p = 1 - \prod_{i=1}^{i=k-1} (1 - pH_i) \tag{4.26}$$

where k is the number of pairwise comparisons considered, and pH_i is the p-value of each comparison. For more information, the reader may refer to [9].

Considering the values of Table 4.6, the p-value of all of the comparisons except MCSS *vs.* G-CMA-ES is less than significance level $\alpha = 0.05$, it cannot be concluded that MCSS is better than all of algorithms except G-CMA-ES because we have to consider FWER in making a conclusion in multiple pairwise comparisons. The MCSS outperforms all of the algorithms except G-CMA-ES considering independence pairwise comparisons due to the fact that the achieved p-values are less than $\alpha = 0.05$. The true p-value for multiple pairwise comparisons can be computed using (4.26):

$$\begin{aligned} p = 1 - \big((1 - 0.16) \cdot (1 - 0.0) \cdot (1 - 0.0) \cdot (1 - 0.05) \cdot (1 - 0.24) \cdot (1 - 0.001) \\ \cdot (1 - 0.025) \cdot (1 - 0.048) \cdot (1 - 0.027) \cdot (1 - 0.001)\big) = 0.17765 \end{aligned} \tag{4.27}$$

Based on this algorithm, it can be claimed that the MCSS algorithm has a better performance in relation with all of the algorithms except G-CMA-ES with a p-value of 0.17765. As a result, if we consider a significance level $\alpha = 0.17765$, the confidence interval for the mentioned claim will be $100(1-\alpha) = 82.23$ %.

4.2.5 Engineering Examples

Three well-studied engineering design problems that have been solved by various optimization methods in the literature are used to examine the efficiency of the MCSS algorithm, and compare the results with those obtained by the CSS.

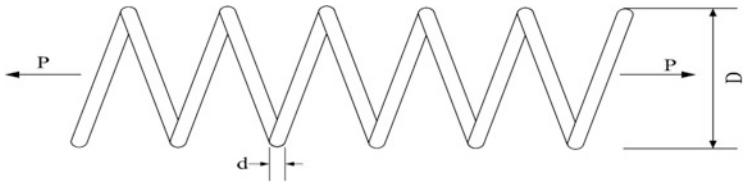

Fig. 4.6 A tension/compression spring

For handling constrains, a simple penalty function is utilized to prevent adding the effect of a robust constrain handling method on the performance of the algorithm.

Example 1 A tension/compression spring design problem
This is a well-known optimization problem which has been used to evaluate the efficiency of different optimization methods [6]. This problem is defined by Belegundu [10] and Arora [11] as depicted in Fig. 4.6. The objective of this optimization problem is to minimize the weight of tension/compression spring. This minimization involves some constrains, i.e., shear stress, frequency, and minimum deflection.

The design variables are the mean coil diameter $D(=x_1)$; the wire diameter $d(=x_2)$, and the number active coils $N(=x_3)$. By considering these decision variables, the cost function can be formulated as:

$$f_{\cos t}(\mathbf{X}) = (x_3 + 2)x_2{x_1}^2 \tag{4.28}$$

$$\begin{aligned} g_1(\mathbf{X}) &= 1 - \frac{{x_2}^3 x_3}{71785 \cdot {x_1}^4} \leq 0, \\ g_2(\mathbf{X}) &= \frac{4{x_2}^2 - x_1 x_2}{12566 \cdot ({x_2}{x_1}^3 - {x_1}^4)} + \frac{1}{5108 \cdot {x_1}^2} - 1 \leq 0, \\ g_3(\mathbf{X}) &= 1 - \frac{140.45 x_1}{{x_2}^2 x_3} \leq 0, \\ g_4(\mathbf{X}) &= \frac{x_1 + x_2}{1.5} - 1 \leq 0. \end{aligned} \tag{4.29}$$

The decision variables are limited as:

$$\begin{aligned} &0.05 \leq x_1 \leq 2, \\ &0.25 \leq x_2 \leq 1.3, \\ &2 \leq x_3 \leq 15. \end{aligned} \tag{4.30}$$

This problem has been solved with various methods by different researchers, Belegundu [10], Arora [11], Coello [12], Coello and Montes [13], He and Wang [14], Montes and Coello [15], and Kaveh and Talathari [14,26]. The results of the best solutions found by different methods are presented in Table 4.7. From Table 4.7

Table 4.7 Optimum results for the tension/compression spring design

Methods	Optimal design variables			
	$x_1(d)$	$x_2(D)$	$x_3(N)$	f_{cost}
Belegundu [10]	0.050000	0.315900	14.250000	0.0128334
Arora [11]	0.053396	0.399180	9.1854000	0.0127303
Coello [12]	0.051480	0.351661	11.632201	0.0127048
Coello and Montes [13]	0.051989	0.363965	10.890522	0.0126810
He and Wang [14]	0.051728	0.357644	11.244543	0.0126747
Montes and Coello [15]	0.051643	0.355360	11.397926	0.012698
Kaveh and Talatahari [16]	0.051865	0.361500	11.000000	0.0126432
Kaveh and Talathari (CSS) [6]	0.051744	0.358532	11.165704	0.0126384
Present work [1]	0.051645	0.356496	11.271529	0.0126192

Table 4.8 Statistical results of different methods for the tension/compression spring

Methods	Best	Mean	Worst	Standard deviation
Belegundu [10]	0.0128334	N/A	N/A	N/A
Arora [11]	0.0127303	N/A	N/A	N/A
Coello [12]	0.0127048	0.012769	0.012822	3.9390e-5
Coello and Montes [13]	0.0126810	0.012742	0.012973	5.9000e-5
He and Wang [14]	0.0126747	0.012730	0.012924	5.1985e-5
Montes and Coello [15]	0.012698	0.013461	0.16485	9.6600e-4
Kaveh and Talatahari [16]	0.0126432	0.012720	0.012884	3.4888e-5
Kaveh and Talathari (CSS) [6]	0.0126384	0.012852	0.013626	8.3564e-5
Present work [1]	0.0126192	0.012794	0.013962	5.3491e-5

it can be understood that the best solution found by MCSS is better than other methods. The statistical simulation results of 30 independent runs for MCSS are illustrated in Table 4.8 and compared to other methods.

Example 2 A welded beam design

One of the practical design problems which has been widely used as a benchmark to test the performance of different optimization methods, is the welded beam design problem as illustrated in Fig. 4.7. The goal of this optimization problem is to minimize the constructing cost of a welded beam that is subjected to different constrains, such as shear (τ) and bending (σ) stresses, buckling load (P_c), end deflection (δ), and end side constraint. Design variables are $h(= x_1)$,$l(= x_2)$, $t(= x_3)$ and $b(= x_4)$. By considering the set-up, welding labor, and the materials costs, the cost function can be expressed as:

$$f_{\cos t}(\mathbf{X}) = 1.1047{x_1}^2 x_2 + 0.04811 x_3 x_4 \cdot (14.0 + x_2) \tag{4.31}$$

Subjected to the following constrains:

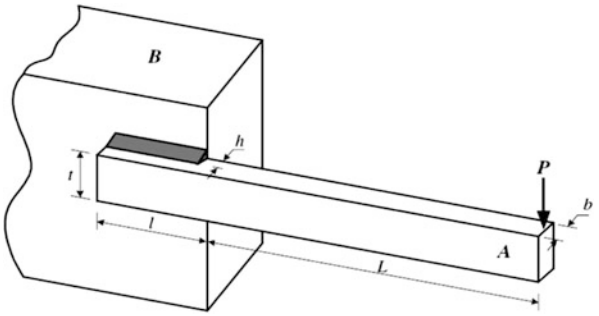

Fig. 4.7 A welded beam system

$$
\begin{aligned}
&g_1(\mathbf{X}) = \tau\big(\{x\}\big) - \tau_{\max} \le 0,\\
&g_2(\mathbf{X}) = \sigma\big(\{x\}\big) - \delta_{\max} \le 0,\\
&g_3(\mathbf{X}) = x_1 - x_4 \le 0,\\
&g_4(\mathbf{X}) = 0.10471{x_1}^2 + 0.04811x_3x_4 \cdot \big(14.0 + x_2\big) - 5.0 \le 0,\\
&g_5(\mathbf{X}) = 0.125 - x_1 \le 0,\\
&g_6(\mathbf{X}) = \delta\big(\{x\}\big) - \delta_{\max} \le 0,\\
&g_7(\mathbf{X}) = P - P_c\big(\{x\}\big) \le 0.
\end{aligned}
\tag{4.32}
$$

Where

$$
\begin{aligned}
&\tau(\mathbf{X}) = \sqrt{(\tau')^2 + 2\tau' \cdot \tau'' \frac{x_2}{2R} + \left(\tau''\right)^2},\\
&\tau' = \frac{P}{\sqrt{2}x_1 \cdot x_2}, \tau'' = \frac{MR}{J},\\
&M = P \cdot \left(L + \frac{x_2}{2}\right), R = \sqrt{\frac{{x_2}^2}{4} + \left(\frac{x_1 + x_2}{2}\right)^2},\\
&J = 2\Big\{\sqrt{2}x_1x_2\Big[\frac{{x_2}^2}{12} + \left(\frac{x_1 + x_3}{2}\right)^2\Big]\Big\},\\
&\sigma(\mathbf{X}) = \frac{6PL}{x_4 \cdot {x_3}^2}, \delta(\mathbf{X}) = \frac{4PL^3}{Ex_3^2x_4},\\
&P_c(\mathbf{X}) = \frac{4.013E\sqrt{\frac{x_3^2x_4^6}{36}}}{L^2}\left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right),\\
&P = 6,000\ \text{lb}, \quad L = 14\ \text{in}, E = 30 \times 10^6\text{psi}, \quad G = 12 \times 10^6\text{psi}
\end{aligned}
\tag{4.33}
$$

And variable boundaries are:

Table 4.9 Optimum results for the design of welded beam

	Optimal design variables				
Methods	$x_1(h)$	$x_2(l)$	$x_3(t)$	$x_4(b)$	f_{cost}
Regsdell and Phillips [17]					
APPROX	0.2444	6.2189	8.2915	0.2444	2.3815
DAVID	0.2434	6.2552	8.2915	0.2444	2.3841
SIMPLEX	0.2792	5.6256	7.7512	0.2796	2.5307
RANDOM	0.4575	4.7313	5.0853	0.6600	4.1185
Deb [18]	0.248900	6.173000	8.178900	0.253300	2.433116
Coello [12]	0.248900	3.420500	8.997500	0.210000	1.748309
Coello and Montes [13]	0.205986	3.471328	9.020224	0.206480	1.728226
He and Wang [14]	0.202369	3.544214	9.048210	0.205723	1.728024
Montes and Coello [15]	0.199742	3.612060	9.037500	0.206082	1.737300
Kaveh and Talathari [16]	0.205700	3.471131	9.036683	0.205731	1.724918
Kaveh and Talathari (CSS) [6]	0.205820	3.468109	9.038024	0.205723	1.724866
Present work [1]	0.205729	3.470493	9.036623	0.205729	1.724853

Table 4.10 Statistical results of different methods for the design of welded beam

Methods	Best	Mean	Worst	Standard deviation
Regsdell and Phillips [17]	2.3815	N/A	N/A	N/A
Deb [18]	2.433116	N/A	N/A	N/A
Coello [12]	1.748309	1.771973	1.785835	0.011220
Coello and Montes [13]	1.728226	1.792654	1.993408	0.074713
He and Wang [14]	1.728024	1.748831	1.782143	0.012926
Montes and Coello [15]	1.737300	1.813290	1.994651	0.070500
Kaveh and Talatahari [16]	1.724918	1.729752	1.775961	0.009200
Kaveh and Talathari (CSS) [6]	1.724866	1.739654	1.759479	0.008064
Present work [1]	1.724853	1.735438	1.753681	0.009527

$$\begin{aligned} &0.1 \leq x_1 \leq 2, \\ &0.1 \leq x_2 \leq 10, \\ &0.1 \leq x_3 \leq 10, \\ &0.1 \leq x_3 \leq 2. \end{aligned} \tag{4.34}$$

This is a well-studied problem that is solved by different researchers using different approaches. Regsdell and Phillips [17] solved it using mathematical-based methods. Deb [18], Coello [12], and Coello and Montes [13], solved it using GA-based algorithms. Also, He and Wang [14] solved it by CPSO, Montes and Coello [15] by Evolutionary strategies, and Kaveh and Talathari [16] by ACO. This problem is also solved by Kaveh and Talathari [6] utilizing the CSS algorithm. The results of the best solution found by each method are listed in Table 4.9. The best solution found by MCSS is better than other results in literature. The result of the MCSS is slightly better than that of the CSS, but the speed of the convergence is much higher compared to the CSS. The results of statistical simulation are

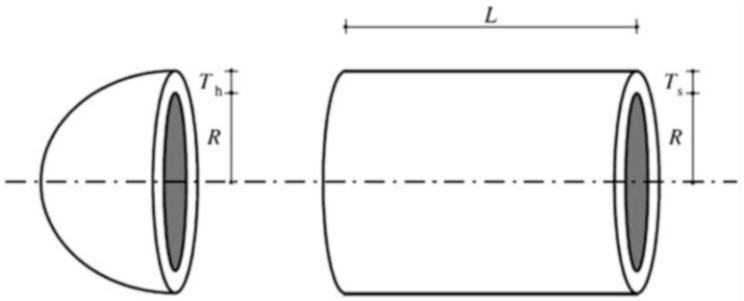

Fig. 4.8 A pressure vessel, and its design variables

presented in Table 4.10. Similar to the CSS algorithm MCSS has a small value for the standard deviation.

Example 3 A pressure vessel design problem
The objective of this optimization is to minimize the cost of fabricating a pressure vessel which is clapped at both ends by hemispherical heads as depicted in Fig. 4.8. The construction cost consists of the cost of materials, forming and welding [19]. The design variables are the thickness of the shell T_s $(= x_1)$, the thickness of the head T_h $(= x_2)$, the inner radius R $(= x_3)$, and the length of cylindrical section of the vessel L $(= x_4)$. T_s and T_h are integer multiples of 0.0625in, the available thickness of the rolled steel plates, but R and L are continuous variables. The mathematical expression of the cost function is:

$$f_{cost}(\mathbf{X}) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2 + 19.84x_1^2x_3, \tag{4.35}$$

The constrain areas are as follows:

$$\begin{aligned} g_1(\mathbf{X}) &= -x_1 + 0.0193x_3 \leq 0, \\ g_2(\mathbf{X}) &= -x_2 + 0.00954x_3 \leq 0, \\ g_3(\mathbf{X}) &= -\pi \cdot x_3^2 x_4 - \frac{4}{3}\pi \cdot x_3^3 + 1,296,000 \leq 0, \\ g_4(\mathbf{X}) &= x_4 - 240 \leq 0. \end{aligned} \tag{4.36}$$

The search space is defined as:

$$\begin{aligned} &0 \leq x_1 \leq 99, \\ &0 \leq x_2 \leq 99, \\ &10 \leq x_3 \leq 200, \\ &10 \leq x_3 \leq 200. \end{aligned} \tag{4.37}$$

Table 4.11 Optimum results for the design of welded beam

	Optimal design variables				
Methods	$x_1(T_s)$	$x_2(T_h)$	$x_3(R)$	$x_4(L)$	f_{cost}
Sandgren [19]	1.125000	0.625000	47.700000	117.701000	8,129.1036
Kannan and Kramer [20]	1.125000	0.625000	58.291000	43.690000	7,198.0428
Deb and Gene [21]	0.937500	0.500000	48.329000	112.679000	6,410.3811
Coello [12]	0.812500	0.437500	40.323900	200.000000	6,288.7445
Coello and Montes [13]	0.812500	0.437500	42.097398	176.654050	6,059.9463
He and Wang [14]	0.812500	0.437500	42.091266	176.746500	6,061.0777
Montes and Coello [15]	0.812500	0.437500	42.098087	176.640518	6,059.7456
Kaveh and Talatahari [16]	0.812500	0.437500	42.098353	176.637751	6,059.7258
Kaveh and Talathari (CSS) [6]	0.812500	0.437500	42.103624	176.572656	6,059.0888
Present work [1]	0.812500	0.437500	42.107406	176.525589	6,058.6233

Table 4.12 Statistical results of different methods for the design of welded beam

Methods	Best	Mean	Worst	Standard deviation
Sandgren [19]	8,129.1036	N/A	N/A	N/A
Kannan and Kramer [20]	7,198.0428	N/A	N/A	N/A
Deb and Gene [21]	6,410.3811	N/A	N/A	N/A
Coello [12]	6,288.7445	6,293.8432	6,308.1497	7.4133
Coello and Montes [13]	6,059.9463	6,177.2533	6,469.3220	130.9297
He and Wang [14]	6,061.0777	6,147.1332	6,363.8041	86.4545
Montes and Coello [15]	6,059.7456	6,850.0049	7,332.8798	426.0000
Kaveh and Talatahari [16]	6,059.7258	6,081.7812	6,150.1289	67.2418
Kaveh and Talathari (CSS) [6]	6,059.0888	6,067.9062	6,085.4765	10.2564
Present work [1]	6,058.6233	6,073.5931	6,108.5479	24.6712

Various types of methods have been used to solve this problem. Some of these approaches are as: a branch and bound method [19], an augmented Lagrangian multiplier approach [20], genetic adaptive search [21], a GA-based algorithm [12], a feasibility-based tournament selection scheme [13], a co-evolutionary particle swarm method [14], an evolution strategy [15], an improved ant colony optimization [16], and the CSS algorithm [6]. The results of the best solution found by different methods are presented in Table 4.11. MCSS algorithm found better solution compared to other techniques and the standard CSS. In Table 4.12 the results of statistical simulations are listed. The mean value of the 30 independent runs for MCSS is slightly weaker than that of the CSS, however, the best solution and speed of the convergence for MCSS is much higher.

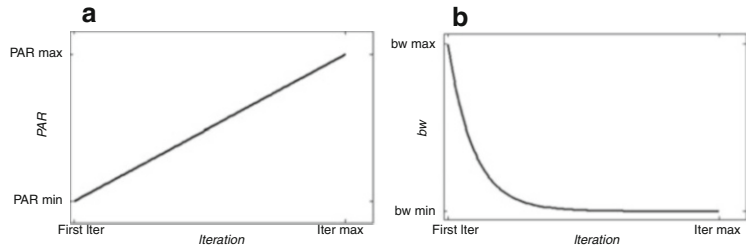

Fig. 4.9 Variation of (**a**) PAR and (**b**) bw versus iteration number [2]

4.3 Improved Magnetic Charged System Search

In this part, the improved version of magnetic charged system search (IMCSS) is presented and also utilized for optimization of truss structures. As mentioned earlier, the standard CSS and MCSS algorithms use harmony search-based approach for process of position correction of CPs. In this process, the CMCR and PAR parameters help the algorithm to find globally and locally improved solutions, respectively [22]. PAR and bw in HS scheme are very important parameters in fine-tuning of optimized solution vectors, and can be potentially useful in adjusting convergence rate of algorithm to optimal solution.

The traditional HS scheme uses fixed value for both PAR and bw. Small PAR values with large bw values can led to poor performance of the algorithm and increase the iterations needed to find optimum solution, also on the other hand small bw values in final iterations increase the fine-tuning of solution vectors, but in the first iterations bw must take a bigger value to enforce the algorithm to increase the diversity of solution vectors. Furthermore, large PAR values with small bw values usually led to the improvement of best solutions in final iterations and converged algorithm to optimal solution vector. To improve the performance of the HS scheme and eliminate the drawbacks lies with fixed values of PAR and bw, IMCSS algorithm uses an improved form of HS algorithm with varied PAR and bw for the step of position correction. PAR and bw change dynamically with iteration number as shown in Fig. 4.9 and expressed as follow [22]:

$$PAR(iter) = PAR_{\min} + \frac{(PAR_{\max} - PAR_{\min})}{iter_{\max}} \cdot iter \tag{4.38}$$

and

$$bw(iter) = bw_{\max} \exp(c \cdot iter), \tag{4.39}$$

$$c = \frac{Ln\left(bw_{\min}/bw_{\max}\right)}{iter_{\max}}, \tag{4.40}$$

where *PAR(iter)* and *bw(iter)* are the values of the PAR and bandwidth for each iteration, respectively, Subscripts *min* and *max* denote the minimum and maximum values for each parameter, respectively, and *iter* is the current iteration number.

4.3.1 A Discrete IMCSS

The IMCSS algorithm can be also applied to optimal design problem with discrete variables. One way to solve discrete problems using a continuous algorithm is to utilize a rounding function which changes the magnitude of a result to the nearest discrete value [23], as follow:

$$X_{j,new} = Fix\left(rand_{j1} \cdot k_a \cdot \frac{F_j}{m_j} \cdot \Delta t^2 + rand_{j2} \cdot k_v \cdot V_{j,old} \cdot \Delta \mathrm{t} + X_{j,old}\right), \quad (4.41)$$

where Fix(X) is a function which rounds each elements of vector X to the nearest allowable discrete value. Using this position updating formula, the agents will be permitted to select discrete values.

4.3.2 An Improved Magnetic Charged System Search for Optimization of Truss Structures with Continuous and Discrete Variables

4.3.2.1 Statement of the Optimization Problem

The aim of size optimization of truss structures is to find the optimum values for cross-sectional area of members A_i, in order to minimize the structural weight W, satisfying the constraints corresponding to the response of the structure. Thus, the optimal design problem can be expressed as:

$$\begin{array}{lll} \text{Find} & \mathrm{X} = [\mathrm{x}_1, \mathrm{x}_2, \mathrm{x}_3, \ldots, \mathrm{x}_\mathrm{n}] & \\ \text{to minimize} & \mathrm{Mer}(\mathrm{X}) = \mathrm{f}_{\mathrm{penalty}}\left(\mathrm{X}\right) \times \mathrm{W}(\mathrm{X}) & \\ \text{subject to} & \sigma_{\min} < \sigma_\mathrm{i} < \sigma_{\max} & \mathrm{i} = 1, 2, \ldots, \mathrm{nm} \\ & \delta_{\min} < \delta_\mathrm{i} < \delta_{\max} & \mathrm{i} = 1, 2, \ldots, \mathrm{nn} \end{array} \quad (4.42)$$

where X is the vector containing the design variables; for a discrete optimum design problem, the variables x_i are selected from an allowable set of discrete values; n is the number of member groups; *Mer(X)* is the merit function; *W(X)* is the cost function, which is taken as the weight of the structure; $f_{penalty}(X)$ is the penalty function which results from the violations of the constraints; nm is the number of members forming the structure; nn is the number of nodes; σ_i and δ_i are the stress of members and nodal displacements, respectively; *min* and *max* mean the lower and upper bounds of constraints, respectively. The cost function can be expressed as:

$$W(X) = \sum_{i=1}^{nm} \rho_i \cdot A_i \cdot L_i \tag{4.43}$$

where ρ_i is the material density of member i, L_i is the length of member i, and A_i is the cross-sectional area of member i.

The penalty function can be defined as:

$$f_{penalty}(X) = \left(1 + \varepsilon_1 \cdot \sum_{i=1}^{np} \left(\phi_{\sigma(i)}^k + \phi_{\delta(i)}^k\right)\right)^{\varepsilon_2}, \tag{4.44}$$

where np is the number of multiple loadings. Here ε_1 is taken as unity and ε_2 is set to 1.5 in the first iterations of the search process, but gradually it is increased to 3 [24]. ϕ_σ^k and ϕ_δ^k are the summation of stress penalties and nodal displacement penalties for kth charged particle which are mathematically expressed as:

$$\phi_\sigma = \sum_{i=1}^{nm} \max\left(\left|\frac{\sigma_i}{\overline{\sigma}_i}\right| - 1, 0\right), \tag{4.45}$$

$$\phi_\delta = \sum_{i=1}^{nm} \max\left(\left|\frac{\delta_i}{\overline{\delta}_i}\right| - 1, 0\right), \tag{4.46}$$

where σ_i, $\overline{\sigma}_i$ are the stress and allowable stress in member i, respectively, and δ_i, $\overline{\delta}_i$ are the displacement of the joints and the allowable displacement, respectively.

4.3.2.2 Numerical Examples

In this section, common truss optimization examples as benchmark problems are used for optimization using the proposed algorithm. This algorithm is applied to problems with both continuous and discrete variables. The final results are compared to those of previous studies to demonstrate the efficiency of the present method. The discrete variables are selected from American Institute of Steel Construction (AISC) Code [25], listed in Table 4.13.

In the proposed algorithm, for all of examples a population of 25 CPs is used and the value of CMCR is set to 0.95.

Example 1 A 10-bar planar truss structure

The 10-bar truss structure is a common problem in the field of structural optimization to verify the efficiency of a proposed optimization algorithm. The geometry and support conditions for this planar, cantilevered truss with loading condition is shown in Fig. 4.10.

There are 10 design variables in this example and a set of pseudo variables ranging from 0.1 to 35.0 in^2 (0.6452 cm^2 to 225.806 cm^2).

In this problem two cases are considered:

Table 4.13 The allowable steel pipe sections taken from AISC code

No.	Area (in^2)	Area (mm^2)	No.	Area (in^2)	Area (mm^2)
1	0.111	71.613	33	3.84	2,477.414
2	0.141	90.968	34	3.87	2,496.769
3	0.196	126.451	35	3.88	2,503.221
4	0.25	161.29	36	4.18	2,696.769
5	0.307	198.064	37	4.22	2,722.575
6	0.391	252.258	38	4.49	2,896.768
7	0.442	285.161	39	4.59	2,961.284
8	0.563	363.225	40	4.8	3,096.768
9	0.602	388.386	41	4.97	3,206.445
10	0.766	494.193	42	5.12	3,303.219
11	0.785	506.451	43	5.74	3,703.218
12	0.994	641.289	44	7.22	4,658.055
13	1	645.16	45	7.97	5,141.925
14	1.228	792.256	46	8.53	5,503.215
15	1.266	816.773	47	9.3	5,999.988
16	1.457	939.998	48	10.85	6,999.986
17	1.563	1,008.385	49	11.5	7,419.43
18	1.62	1,045.159	50	13.5	8,709.66
19	1.8	1,161.288	51	13.9	8,967.724
20	1.99	1,283.868	52	14.2	9,161.272
21	2.13	1,374.191	53	15.5	9,999.98
22	2.38	1,535.481	54	16	10,322.56
23	2.62	1,690.319	55	16.9	10,903.2
24	2.63	1,696.771	56	18.8	12,129.01
25	2.88	1,858.061	57	19.9	12,838.68
26	2.93	1,890.319	58	22	14,193.52
27	3.09	1,993.544	59	22.9	14,774.16
28	1.13	729.031	60	24.5	15,806.42
29	3.38	2,180.641	61	26.5	17,096.74
30	3.47	2,238.705	62	28	18,064.48
31	3.55	2,290.318	63	30	19,354.8
32	3.63	2,341.931	64	33.5	21,612.86

Case 1, $P_1 = 100$ kips (444.8 kN) and $P_2 = 0$, and Case 2, $P_1 = 150$ kips (667.2 kN) and $P_2 = 50$ kips (222.4 kN).

The material density is 0.1 lb/in^3 (2,767.990 kg/m^3) and the modulus of elasticity is 10,000 ksi (68,950 MPa). The members are subjected to the stress limits of ± 25 ksi (172.375 MPa) and all nodes in both vertical and horizontal directions are subjected to the displacement limits of ± 2.0 in (5.08 cm). Figure 4.11 shows a comparison of the convergence history of both cases for MCSS and IMCSS algorithms.

Tables 4.14 and 4.15 are provided for comparison of the optimal design results with those of the previous studies for both cases. In both cases the HS algorithm reach its best solutions after 20,000 analyses, and the PSO and PSOPC algorithms after 3,000 iterations (150,000 analyses). The HPSACO algorithm finds the best solution after 10,650 and 9,925 analyses, for Case 1 and Case 2, respectively.

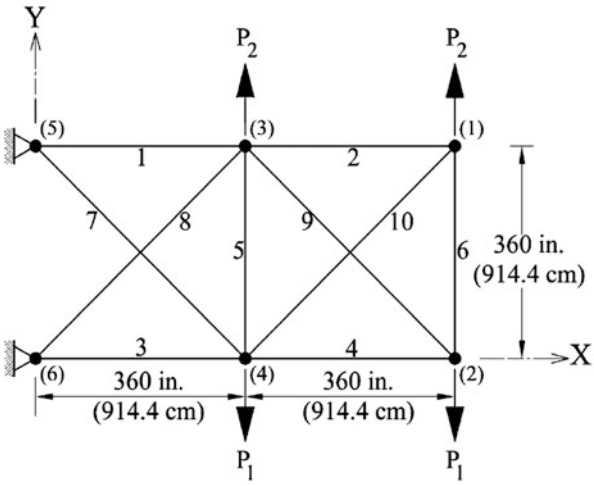

Fig. 4.10 Schematic of a10-bar planar truss structure

The MCSS and IMCSS algorithms achieve the best solutions after 355 iterations (8,875 analyses) and 339 iterations (8,475 analyses), respectively. The best weights of IMCSS are 5,064.6 lb for Case 1 and 4,679.15 for Case 2.

As seen in both Tables, although the best weights of IMCSS in both cases are a little bigger than the HPSACO, but it has lower penalty values rather than HPSACO, and therefore IMCSS has a lower merit function than HPSACO.

Example 2 A 52-bar planar truss

The 52-bar planar truss structure shown in Fig. 4.12 has been analyzed by Lee and Geem [27], Li et al. [28], Wu and Chow [30] and Kaveh and Talatahari [31].

The members of this structure are divided into 12 groups: (1) A1–A4, (2) A5–A10, (3) A11–A13, (4) A14–A17, (5) A18–A23, (6) A24–A26, (7) A27–A30, (8) A31–A36, (9) A37–A39, (10) A40–A43, (11) A44–A49, and (12) A50–A52.

The material density is 7,860.0 kg/m^3 and the modulus of elasticity is 2.07 $\times$ 105 MPa. The members are subjected to stress limitations of $\pm$180 MPa. Both of the loads, $P_x = 100$kN and $P_y = 200$kN, are considered.

Table 4.16 and Fig. 4.13 are provided for comparison of the optimal design results with the previous studies and convergence rates for the 52-bar planar truss structure, respectively.

Table 4.16 shows that, the best weight of MCSS and IMCSS algorithms are 1,904.05 lb and 1,902.61 lb, respectively, while for DHPSACO is 1,904.83 lb.

The MCSS and IMCSS algorithms find the best solutions after 4,225 and 4,075 analyses respectively, but the DHPSACO reach a good solution in 5,300 analyses. As it can be seen in the results of Table 4.16, the IMCSS algorithm achieve good optimal results than previous methods like MCSS, PSO, PSOPC, HPSO and DHPSACO algorithms.

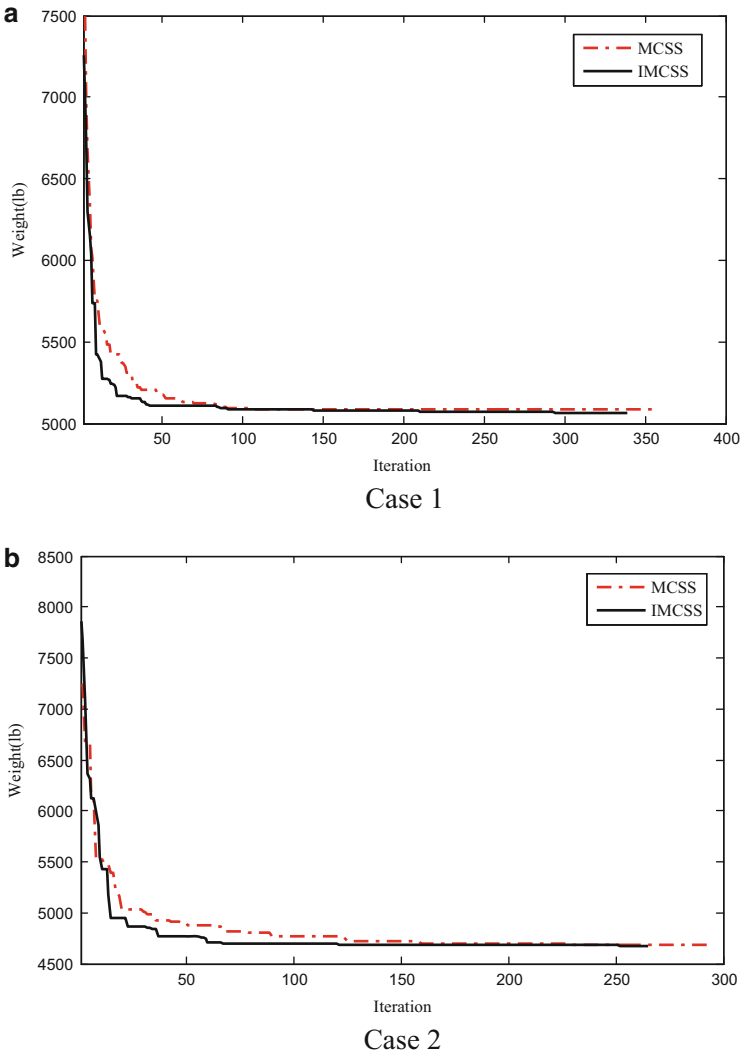

Fig. 4.11 Convergence history for the 10-bar planar truss structure using MCSS, IMCSS [2]

Example 3 A 72-bar spatial truss

In the 72-bar spatial truss structure which is shown in Fig. 4.14, the material density is 0.1 lb/in^3 (2,767.990 kg/m^3) and the modulus of elasticity is 10,000 ksi (68,950 MPa). The nodes are subjected to the displacement limits of ±0.25 in (±0.635 cm) and the members are subjected to the stress limits of ±25 ksi (±172.375 MPa).

All members of this spatial truss are categorized into 16 groups using symmetry: (1) A1–A4, (2) A5–A12, (3) A13–A16, (4) A17–A18, (5) A19–A22, (6) A23–A30,

Table 4.14 Optimal design comparison for the 10-bar planner truss (Case 1)

		Camp et al. [26]	Lee and Geem [27]	Li et al. [28]			Kaveh and Talatahari [29]	Present work [2]	
Element group		GA	HS	PSO	PSOPC	HPSO	HPSACO	MCSS	IMCSS
1	A1	28.92	30.15	33.469	30.569	30.704	30.307	29.5766	30.0258
2	A2	0.1	0.102	0.11	0.1	0.1	0.1	0.1142	0.1
3	A3	24.07	22.71	23.177	22.974	23.167	23.434	23.806	23.627
4	A4	13.96	15.27	15.475	15.148	15.183	15.505	15.887	15.973
5	A5	0.1	0.102	3.649	0.1	0.1	0.1	0.1137	0.1
6	A6	0.56	0.544	0.116	0.547	0.551	0.5241	0.1003	0.5167
7	A7	7.69	7.541	8.328	7.493	7.46	7.4365	8.6049	7.4567
8	A8	21.95	21.56	23.34	21.159	20.978	21.079	21.682	21.437
9	A9	22.09	21.45	23.014	21.556	21.508	21.229	20.303	20.744
10	A10	0.1	0.1	0.19	0.1	0.1	0.1	0.1117	0.1
Weight(lb)		**5,076.31**	**5,057.88**	**5,529.5**	**5,061**	**5,060.92**	**5,056.56**	**5,086.9**	**5,064.6**
Displacement Constraint		–	–	–	–	5.53E-07	9.92E-04	1.49E-05	5.85E-08
No. of analyses		N/A	20,000	150,000	150,000	N/A	10,650	8,875	8,475

Table 4.15 Optimal design comparison for the 10-bar planner truss (Case 2)

Element group		Lee and Geem [27]	Li et al. [28]			Kaveh and Talatahari [29]	Present work [2]	
		HS	PSO	PSOPC	HPSO	HPSACO	MCSS	IMCSS
1	A1	23.25	22.935	23.473	23.353	23.194	22.863	23.299
2	A2	0.102	0.113	0.101	0.1	0.1	0.120	0.1
3	A3	25.73	25.355	25.287	25.502	24.585	25.719	25.682
4	A4	14.51	14.373	14.413	14.25	14.221	15.312	14.510
5	A5	0.1	0.1	0.1	0.1	0.1	0.101	0.1
6	A6	1.977	1.99	1.969	1.972	1.969	1.968	1.969
7	A7	12.21	12.346	12.362	12.363	12.489	12.310	12.149
8	A8	12.61	12.923	12.694	12.894	12.925	12.934	12.360
9	A9	20.36	20.678	20.323	20.356	20.952	19.906	20.869
10	A10	0.1	0.1	0.103	0.101	0.101	0.100	0.1
Weight(lb)		**4,668.81**	**4,679.47**	**4,677.7**	**4,677.29**	**4,675.78**	**4,686.47**	**4,679.15**
Displacement constraint		–	–	–	0	7.92E-04	0	0
Stress constraint		–	–	–	2.49E-05	7.97E-05	0	0
No. of analyses		N/A	150,000	150,000	N/A	9,625	7,350	6,625

(7) A31–A34, (8) A35–A36, (9) A37–A40, (10) A41–A48, (11) A49–A52, (12) A53–A54, (13) A55–A58, (14) A59–A66 (15), A67–A70, and (16) A71–A72.

Two optimization cases are implemented:

Case 1: The discrete variables are selected from the set 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) or {0.65, 1.29, 1.94, 2.58, 3.23, 3.87, 4.52, 5.16, 5.81, 6.45, 7.10, 7.74, 8.39, 9.03, 9.68, 10.32, 10.97, 12.26, 12.90, 13.55, 14.19, 14.84, 15.48, 16.13, 16.77, 17.42, 18.06, 18.71, 19.36, 20.00, 20.65} (cm^2).

Case 2: The discrete variables are selected from AISC code in Table 4.13. Table 4.17 lists the values and directions of the two load cases applied to the 72-bar spatial truss.

Tables 4.18 and 4.19 are provided for comparison the results of MCSS and IMCSS algorithms with the results of the previous studies for both cases. The Convergence history for both algorithms is shown in Fig. 4.15.

In Case 1, the best weight of the IMCSS and DHPSACO algorithm are 385.54 lb (174.88 kg), while it is 389.49 lb, 388.94 lb, 387.94 lb, 400.66 lb for the MCSS, HPSO, HS, and GA, respectively. For the PSO and PSOPC algorithms, these algorithms do not get optimal results when the maximum number of iterations is reached. The IMCSS algorithm gets the best solution after 145 iterations (3,625 analyses) while it takes for MCSS and DHPSACO 216 iterations (5,400 analyses) and 213 iterations (5,330 analyses), respectively.

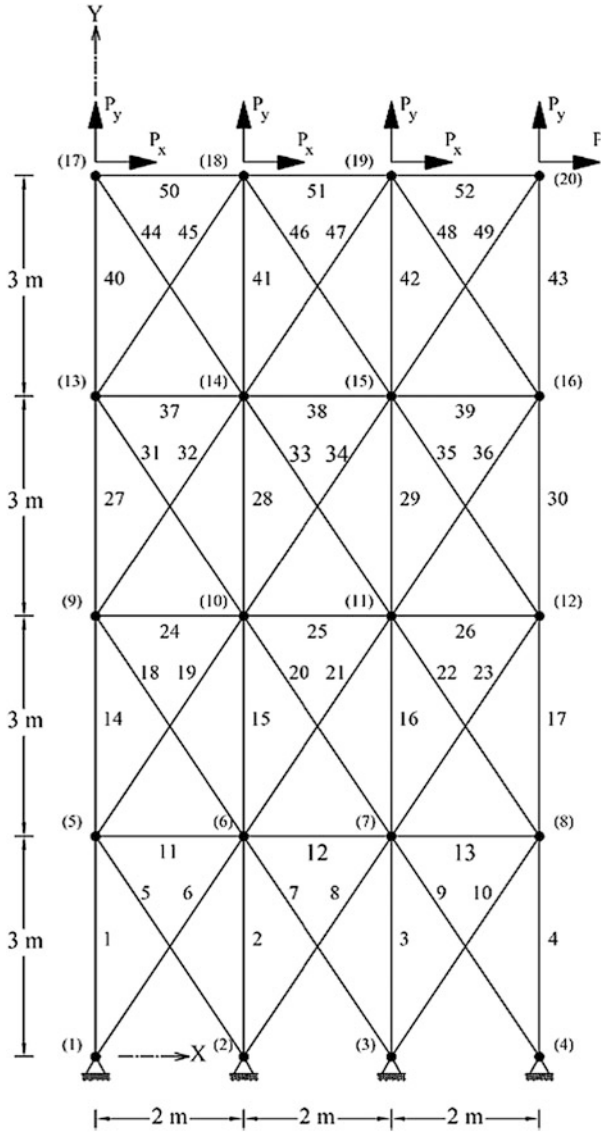

Fig. 4.12 Schematic of a 52-bar planar truss

In Case 2, the best obtained weight from IMCSS is 389.60 lb, but it is 393.13 lb, 389.87 lb, 392.84 lb, 393.06 lb and 393.38 lb for MCSS, CS, ICA, CSS and HPSACO algorithms, respectively. IMCSS algorithm finds the best solutions after 173 iterations (4,325 analyses), while MCSS, CS, ICA, CSS and HPSACO algorithms, need 4,775, 4,840, 4,500, 7,000 and 5,330 analyses to find the best solutions.

Table 4.16 Optimal design comparison for the 52-bar planar truss

Element group	Lee and Geem [27]	Li et al. [28]			Kaveh and Talatahari [31]	Present work [2]	
	HS	PSO	PSOPC	HPSO	DHPSACO	MCSS	IMCSS
1	4,658.055	4,658.055	5,999.988	4,658.055	4,658.055	4,658.055	4,658.055
2	1,161.288	1,374.19	1,008.38	1,161.288	1,161.288	1,161.288	1,161.288
3	506.451	1,858.06	2,696.38	363.225	494.193	363.225	494.193
4	3,303.219	3,206.44	3,206.44	3,303.219	3,303.219	3,303.219	3,303.219
5	940	1,283.87	1,161.29	940	1,008.385	939.998	939.998
6	494.193	252.26	729.03	494.193	285.161	506.451	494.193
7	2,290.318	3,303.22	2,238.71	2,238.705	2,290.318	2,238.705	2,238.705
8	1,008.385	1,045.16	1,008.38	1,008.385	1,008.385	1,008.385	1,008.385
9	2,290.318	126.45	494.19	388.386	388.386	388.386	494.193
10	1,535.481	2,341.93	1,283.87	1,283.868	1,283.868	1,283.868	1,283.868
11	1,045.159	1,008.38	1,161.29	1,161.288	1,161.288	1,161.288	1,161.288
12	506.451	1,045.16	494.19	792.256	506.451	729.031	494.193
Weight (kg)	1,906.76	2,230.16	2,146.63	1,905.49	1,904.83	1,904.05	1,902.61
No. of analyses	N/A	N/A	N/A	50,000	5,300	4,225	4,075

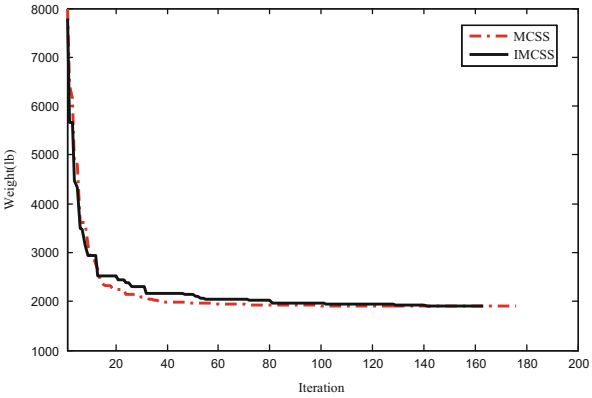

Fig. 4.13 Convergence history for the 52-bar planar truss structure using MCSS, IMCSS [2]

Example 4 A 120-bar dome shaped truss

The 120-bar dome truss was first analyzed by Soh and Yang [34] to obtain the optimal sizing and configuration variables, but for this study only sizing variables are considered to minimize the structural weight in this example, similar to Lee and Geem [27] and Keleşoğlu and Ülker [35].

The geometry of this structure is shown in Fig. 4.16. The modulus of elasticity is 30,450 ksi (210,000 MPa) and the material density is 0.288 lb/in^3 (7,971.810 kg/m^3). The yield stress of steel is taken as 58.0 ksi (400 MPa).

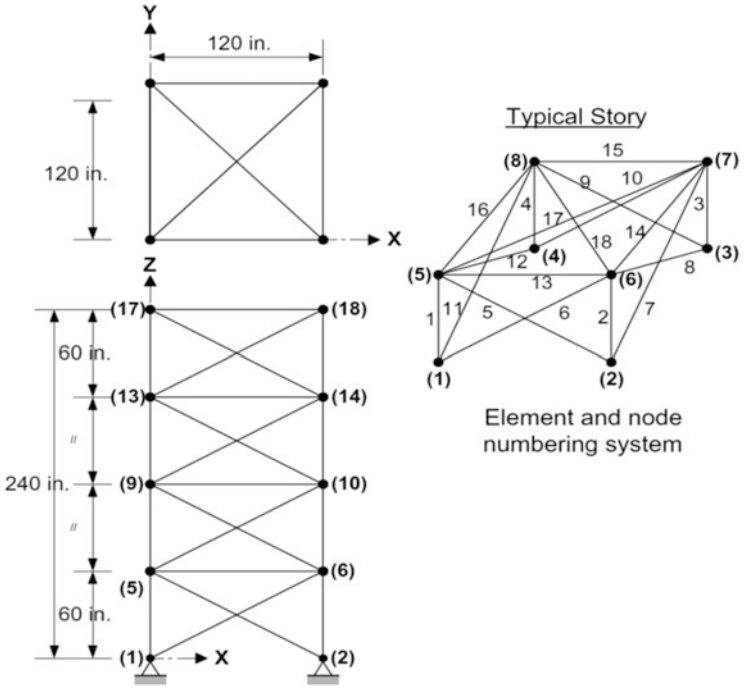

Fig. 4.14 Schematic of a 72-bar spatial truss

Table 4.17 Loading conditions for the 72-bar spatial truss

	Case 1			Case 2		
Node	P_X kips (kN)	P_y kips (kN)	P_z kips (kN)	P_X kips (kN)	P_y kips (kN)	P_z kips (kN)
17	5.0 (22.25)	5.0 (22.25)	−5.0 (−22.25)	0.0	0.0	−5.0 (−22.25)
18	0.0	0.0	0.0	0.0	0.0	−5.0 (−22.25)
19	0.0	0.0	0.0	0.0	0.0	−5.0 (−22.25)
20	0.0	0.0	0.0	0.0	0.0	−5.0 (−22.25)

The allowable tensile and compressive stresses are used according to the AISC-ASD code [25], as follows:

$$\begin{cases} \sigma_i^+ = 0.6F_y & for\ \sigma_i \geq 0 \\ \sigma_i^- & for\ \sigma_i < 0 \end{cases} \tag{4.47}$$

where σ_i^- is calculated according to the slenderness ratio

Table 4.18 Optimal design comparison for the 72-bar truss (Case 1)

		Wu and Chow [30]	Lee and Geem [27]	Li et al. [28]			Kaveh and Talatahari [31]	Present work [2]	
Element group		GA	HS	PSO	PSOPC	HPSO	DHPSACO	MCSS	IMCSS
A1	A1–A4	1.5	1.9	2.6	3	2.1	1.9	1.8	2
A2	A5–A12	0.7	0.5	1.5	1.4	0.6	0.5	0.5	0.5
A3	A13–A16	0.1	0.1	0.3	0.2	0.1	0.1	0.1	0.1
A4	A17–A18	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
A5	A19–A22	1.3	1.4	2.1	2.7	1.4	1.3	1.3	1.3
A6	A23–A30	0.5	0.6	1.5	1.9	0.5	0.5	0.5	0.5
A7	A31–A34	0.2	0.1	0.6	0.7	0.1	0.1	0.1	0.1
A8	A35–A36	0.1	0.1	0.3	0.8	0.1	0.1	0.1	0.1
A9	A37–A40	0.5	0.6	2.2	1.4	0.5	0.6	0.7	0.5
A10	A41–A48	0.5	0.5	1.9	1.2	0.5	0.5	0.6	0.5
A11	A49–A52	0.1	0.1	0.2	0.8	0.1	0.1	0.1	0.1
A12	A53–A54	0.2	0.1	0.9	0.1	0.1	0.1	0.1	0.1
A13	A55–A58	0.2	0.2	0.4	0.4	0.2	0.2	0.2	0.2
A14	A59–A66	0.5	0.5	1.9	1.9	0.5	0.6	0.6	0.6
A15	A67–A70	0.5	0.4	0.7	0.9	0.3	0.4	0.4	0.4
A16	A71–A72	0.7	0.6	1.6	1.3	0.7	0.6	0.4	0.6
Weight (kg)		**400.6**	**387.94**	**1,089.88**	**1,069.79**	**388.94**	**385.54**	**389.49**	**385.54**
No. of analyses		N/A	N/A	N/A	150,000	50,000	5,330	5,400	3,625

$$\sigma_i^- = \begin{cases} \left[\left(1 - \dfrac{\lambda_i^2}{2C_c^2}\right)F_y\right] \Big/ \left(\dfrac{5}{3} + \dfrac{3\lambda_i}{8C_c} - \dfrac{\lambda_i^3}{8C_c^3}\right) & \text{for } \lambda_i < C_c \\ \dfrac{12\pi^2 E}{23\lambda_i^2} & \text{for } \lambda_i \geq C_c \end{cases} \tag{4.48}$$

where E is the modulus of elasticity, F_y is the yield stress of steel, C_c is the slenderness ratio (λ_i) dividing the elastic and inelastic buckling regions ($C_c = \sqrt{2\pi^2 E/F_y}$), λ_i is the slenderness ratio ($\lambda i = kL_i/r_i$), k is the effective length factor, L_i is the member length and r_i is the radius of gyration. The radius of gyration (r_i) can be expressed in terms of cross-sectional areas, i.e., $r_i = aA_i^b$. Here, a and b are the constants depending on the types of sections adopted for the members such as pipes, angles, and tees. In this paper, pipe sections (a = 0.4993 and b = 0.6777) were adopted for bars [36].

All members of the dome are categorized into seven groups, as shown in Fig. 4.16. The dome is considered to be subjected to vertical loading at all the unsupported joints. These were taken as −13.49 kips (60 kN) at node 1, −6.744

Table 4.19 Optimal design comparison for the 72-bar truss (Case 2)

		Wu and Chow [30]	Li et al. [28]			Kaveh and Talatahari			Kaveh and Bakhshpoori [33]	Present work [2]	
Element group		GA	PSO	PSOPC	HPSO	DHPSACO [31]	CSS [23]	ICA [32]	CS	MCSS	IMCSS
1	A1–A4	0.196	7.22	4.49	4.97	1.8	1.99	1.99	1.8	1.8	1.8
2	A5–A12	0.602	1.8	1.457	1.228	0.442	0.442	0.442	0.563	0.563	0.563
3	A13–A16	0.307	1.13	0.111	0.111	0.141	0.111	0.111	0.111	0.111	0.111
4	A17–A18	0.766	0.2	0.111	0.111	0.111	0.111	0.141	0.111	0.111	0.111
5	A19–A22	0.391	3.09	2.62	2.88	1.228	0.994	1.228	1.266	1.457	1.228
6	A23–A30	0.391	0.79	1.13	1.457	0.563	0.563	0.602	0.563	0.563	0.563
7	A31–A34	0.141	0.56	0.196	0.141	0.111	0.111	0.111	0.111	0.111	0.111
8	A35–A36	0.111	0.79	0.111	0.111	0.111	0.111	0.141	0.111	0.111	0.111
9	A37–A40	1.8	3.09	1.266	1.563	0.563	0.563	0.563	0.563	0.563	0.391
10	A41–A48	0.602	1.23	1.457	1.228	0.563	0.563	0.563	0.442	0.442	0.563
11	A49–A52	0.141	0.11	0.111	0.111	0.111	0.111	0.111	0.111	0.111	0.111
12	A53–A54	0.307	0.56	0.111	0.196	0.25	0.111	0.111	0.111	0.111	0.111
13	A55–A58	1.563	0.99	0.442	0.391	0.196	0.196	0.196	0.196	0.196	0.196
14	A59–A66	0.766	1.62	1.457	1.457	0.563	0.563	0.563	0.602	0.563	0.563
15	A67–A70	0.141	1.56	1.228	0.766	0.442	0.442	0.307	0.391	0.307	0.307
16	A71–A72	0.111	1.27	1.457	1.563	0.563	0.766	0.602	0.563	0.766	0.563
Weight (lb)		**427.20**	**1,209**	**941.82**	**933.09**	**393.38**	**393.06**	**392.84**	**389.87**	**393.13**	**389.6**
No. of analyses		N/A	N/A	150,000	50,000	5,330	7,000	4,500	4,840	4,775	4,325

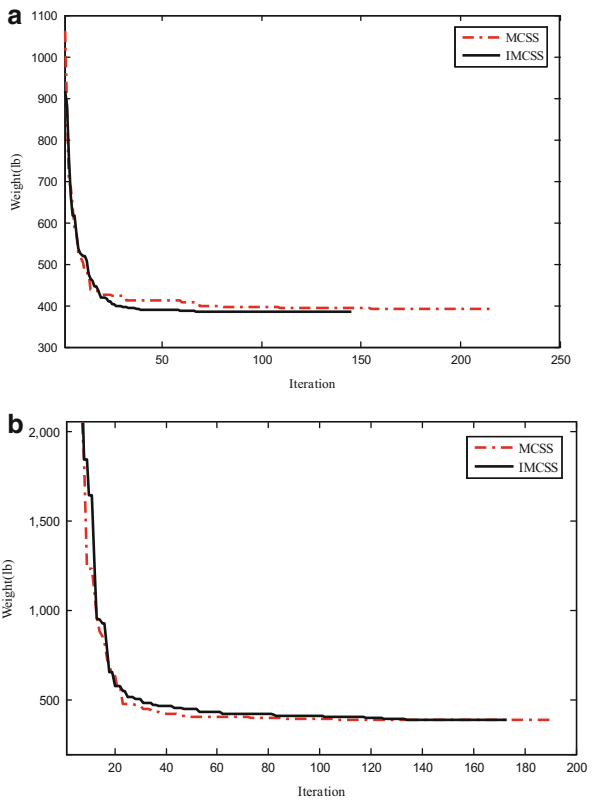

Fig. 4.15 Convergence history for the 72-bar truss structure using MCSS, IMCSS [2]. (**a**) Case 1 and (**b**) Case 2

kips (30 kN) at nodes 2–14, and −2.248 kips (10 kN) at the rest of the nodes. The minimum cross-sectional area of all members is 0.775 in^2 (2 cm^2).

In this example, two cases of constraints are considered:

Case 1, with stress constraints and no displacement constraints, and Case 2, with stress constraints and displacement limitations of ±0.1969 in (5 mm) imposed on all nodes in x- and y-directions. For two cases, the maximum cross-sectional area is 5.0 in^2 (32.26 cm^2).

Figure 4.17 shows the convergence history for all cases and Table 4.20 gives the best solution vectors and weights for both cases.

In Case 1, the best weights of MCSS and IMCSS are 19,607.39 lb and 19,476.92 lb, respectively, while for the Ray, HPSACO and PSOPC are 19,476.19 lb, 19,491.30 lb and 19,618.7 lb. The MCSS and IMCSS find the best

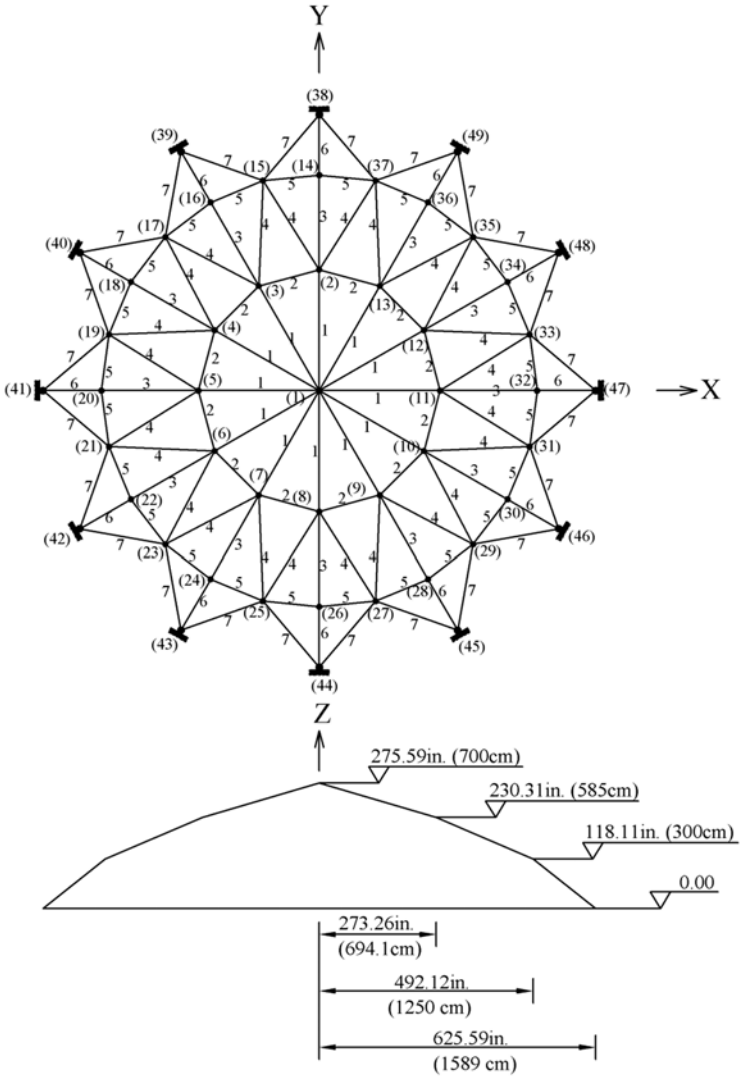

Fig. 4.16 Schematic of a 120-bar dome shaped truss

solutions in 314 iterations (7,850 analyses) and 299 iterations (7,475 analyses), respectively, but for Ray and HPSACO algorithms, it takes 19,950 and 10,025 analyses to reach the best solutions, respectively.

In Case 2, the MCSS and IMCSS algorithms need 386 iterations (9,650 analyses) and 324 iterations (8,100 analyses) to find the best solutions, respectively, while for Ray and HPSACO algorithms 19,950 and 10,075 analyses is required. The best weights obtained from MCSS and IMCSS algorithms are 19,928 lb and

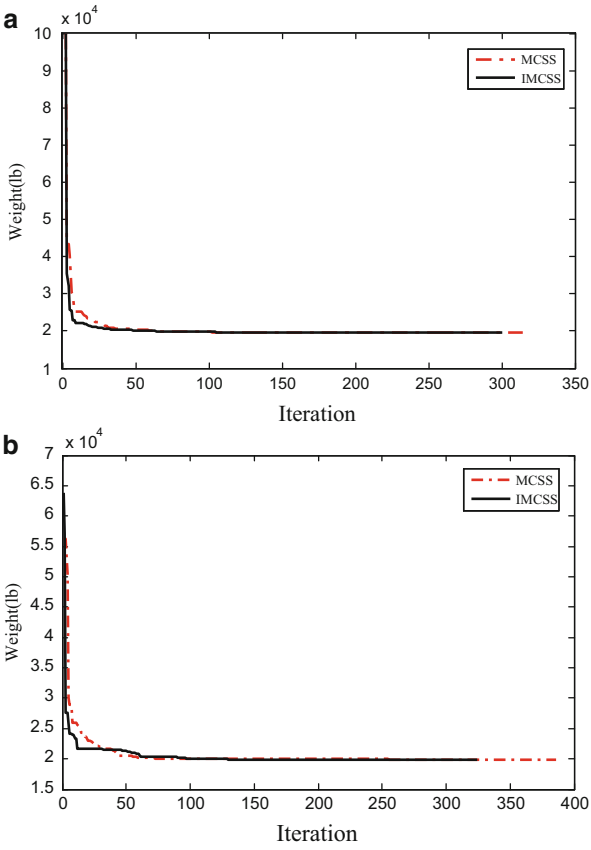

Fig. 4.17 Comparison of the convergence rates between the MCSS and IMCSS for the 120-bar dome truss structure [2], (**a**) Case 1 and (**b**) Case 2

19,796.71 lb, respectively, but from the Ray, HPSACO and PSOPC are 20,071.9 lb, 20,078 and 20,681.7 lb, respectively.

Some design examples as benchmark problems are optimized using the IMCSS algorithm for both continuous and discrete design variables. The aim of this study is to find the best merit function, i.e. considering both penalty and cost functions. In comparison the results with those of the previous studies for all examples, the IMCSS has the better merit function than all of previous algorithms, however for few examples the best weight obtained from IMCSS algorithm is not the best in the results. Also, the results demonstrate the effectiveness of improvement process for MCSS algorithm to achieve a better convergence and find better solutions especially in final iterations of the improved algorithm.

Table 4.20 Optimal design comparison for the 120-bar dome truss (two cases) optimal cross-sectional areas (in^2)

	Case 1						
Element group	Lee and Geem [27]			Kaveh and Talatahari [29]	Kaveh and Khayatazad [37]	Present work [2]	
	HS	PSO	PSOPC	HPSACO	Ray	MCSS	IMCSS
1	3.295	3.147	3.235	3.311	3.128	3.1108	3.1208
2	3.396	6.376	3.37	3.438	3.357	3.3903	3.3566
3	3.874	5.957	4.116	4.147	4.114	4.106	4.111
4	2.571	4.806	2.784	2.831	2.783	2.7757	2.7811
5	1.15	0.775	0.777	0.775	0.775	0.9674	0.8055
6	3.331	13.798	3.343	3.474	3.302	3.2981	3.3001
7	2.784	2.452	2.454	2.551	2.453	2.4417	2.4451
Weight (lb)	**19,707.77**	**32,432.9**	**19,618.7**	**19,491.3**	**19,476.19**	**19,607.39**	**19,476.92**
No. of analyses	35,000	N/A	125,000	10,025	19,950	7,850	7,475
	Case 2						
Element group	Lee and Geem [27]			Kaveh and Talatahari [29]	Kaveh and Khayatazad [37]	Present work	
	HS	PSO	PSOPC	HPSACO	Ray	MCSS	IMCSS
1	3.296	15.978	3.083	3.779	3.084	3.309	3.3187
2	2.789	9.599	3.639	3.377	3.360	2.6316	2.4746
3	3.872	7.467	4.095	4.125	4.093	4.2768	4.2882
4	2.57	2.79	2.765	2.734	2.762	2.7918	2.8103
5	1.149	4.324	1.776	1.609	1.593	0.9108	0.7753
6	3.331	3.294	3.779	3.533	3.294	3.5168	3.523
7	2.781	2.479	2.438	2.539	2.434	2.3769	2.3826
Weight (lb)	**19,893.34**	**41,052.7**	**20,681.7**	**20,078**	**20,071.9**	**19,928**	**19,796.71**
No. of analyses	35,000	N/A	125,000	10,075	19,950	9,650	8,100

References

1. Kaveh A, Motie Share MA, Moslehi M (2013) A new meta-heuristic algorithm for optimization: magnetic charged system search. Acta Mech 224(1):85–107
2. Kaveh A, Jafarvand A, Mirzaei B (2014) An improved magnetic charged system search for optimization of truss structures with continuous and discrete variables. Asian J Civil Eng 15 (1):95–105
3. Kaveh A, Talatahari S (2010) A novel heuristic optimization method: charged system search. Acta Mech 213(3–4):267–286
4. Halliday D, Resnick R, Walker J (2008) Fundamentals of physics, 8th edn. Wiley, New York
5. Tsoulos IG (2008) Modifications of real code genetic algorithm for global optimization. Appl Math Comput 203:598–607

6. Kaveh A, Talatahari S (2010) Optimal design of truss structures via the charged system search algorithm. Struct Multidisp Optim 37(6):893–911
7. Hines W, Montgomery D (1990) Probability and statistics in engineering and management science, 3rd edn. Wiley, New York
8. Suganthan PN, Hansen N, Liang, JJ, Deb K, Chen Y.-P, Auger A, Tiwari S (2005) Problem definitions and evaluation criteria for CEC 2005 Special Session on real-parameter optimization. Technical Report, Nanyang Technological University, Singapore and KanGAL report number 2005005
9. Garcia S, Molina D, Lozano M, Herrera F (2009) A study on the use of non-parametric tests for analyzing the evolutionary algorithms behavior: a case study on the CEC 2005 Special Session on Real Parameter Optimization. J Heurist 15:617–644
10. Belegundu AD (1982) A study of mathematical programming methods for structural optimization. Ph.D. thesis, Department of Civil and Environmental Engineering, University of Iowa, Iowa, USA
11. Arora JS (2000) Introduction to optimum design. McGraw-Hill, New York (1989)
12. Coello CAC (2000) Use of a self-adaptive penalty approach for engineering optimization problems. Comput Ind 41:113–127
13. Coello CAC, Montes EM (2002) Constraint-handling in genetic algorithms through the use of dominance-based tournament selection. Adv Eng Inform 16:193–203
14. He Q, Wang L (2007) An effective co-evolutionary particle swarm optimization for constrained engineering design problems. Eng Appl Artif Intell 20:89–99
15. Montes EM, Coello CAC (2008) An empirical study about the usefulness of evolution strategies to solve constrained optimization problems. Int J Gen Syst 37(4):443–473
16. Kaveh A, Talatahari S (2010) An improved ant colony optimization for constrained engineering design problems. Eng Comput 27(1):155–182
17. Ragsdell KM, Phillips DT (1976) Optimal design of a class of welded structures using geometric programming. ASME J Eng Ind B 98(3):1021–1025
18. Deb K (1991) Optimal design of a welded beam via genetic algorithms. AIAA J 29 (11):2013–2015
19. Sandgren E (1988) Nonlinear integer and discrete programming in mechanical design. In: Proceedings of the ASME design technology conference, Kissimine, FL, pp 95–105
20. Kannan BK, Kramer SN (1994) An augmented Lagrange multiplier based method for mixed integer discrete continuous optimization and its applications to mechanical design. Trans ASME J Mech Des 116:318–320
21. Deb K, Gene AS (1997) A robust optimal design technique for mechanical component design. In: Dasgupta D, Michalewicz Z (eds) Evolutionary algorithms in engineering applications. Springer, Berlin, pp 497–514
22. Mahdavi M, Fesanghary M, Damangir E (2007) An improved harmony search algorithm for solving optimization problems. Appl Math Comput 188:1567–1579
23. Kaveh A, Talatahari S (2010) A charged system search with a fly to boundary method for discrete optimum design of truss structures. Asian J Civil Eng 11(3):277–293
24. Kaveh A, Farahmand Azar B, Talatahari S (2008) Ant colony optimization for design of space trusses. Int J Space Struct 23(3):167–181
25. American Institute of Steel Construction (AISC) (1989) Manual of steel construction-allowable stress design, 9th edn. AISC, IL, Chicago
26. Camp C, Pezeshk S, Cao G (1998) Optimized design of two dimensional structures using a genetic algorithm. J Struct Eng ASCE 124(5):551–559
27. Lee KS, Geem ZW (2004) A new structural optimization method based on the harmony search algorithm. Comput Struct 82:781–798
28. Li LJ, Huang ZB, Liu F, Wu QH (2007) A heuristic particle swarm optimizer for optimization of pin connected structures. Comput Struct 85:340–349
29. Kaveh A, Talatahari S (2009) Particle swarm optimizer, ant colony strategy and harmony search scheme hybridized for optimization of truss structures. Comput Struct 87:267–283

30. Wu SJ, Chow PT (1995) Steady-state genetic algorithms for discrete optimization of trusses. Comput Struct 56(6):979–991
31. Kaveh A, Talatahari S (2009) A particle swarm ant colony optimization for truss structures with discrete variables. J Construct Steel Res 65(8–9):1558–1568
32. Kaveh A, Talatahari S (2010) Optimum design of skeletal structures using imperialist competitive algorithm. Comput Struct 88:1220–1229
33. Kaveh A, Bakhshpoori T (2013) Optimum design of space trusses using cuckoo search algorithm with lévy flights. Iranian J Sci Technol: Trans Civil Eng 37:C1.1–15
34. Soh CK, Yang J (1996) Fuzzy controlled genetic algorithm search for shape optimization. J Comput Civil Eng ASCE 10(2):143–150
35. Keleşoğlu O, Ülker M (2005) Fuzzy optimization geometrical nonlinear space truss design. Turkish J Eng Environ Sci 29:321–329
36. Saka MP (1990) Optimum design of pin-jointed steel structures with practical applications. J Struct Eng ASCE 116:2599–620
37. Kaveh A, Khayatazad M (2013) Ray optimization for size and shape optimization of truss structures. Comput Struct 117:82–94

Chapter 5
Field of Forces Optimization

5.1 Introduction

Although different metaheuristic algorithms have some differences in approaches to determine the optimum solution, however their general performance is approximately the same. They start the optimization with random solutions; and the subsequent solutions are based on randomization and some other rules. With progressing the optimization process, the power of rules increases, and the power of randomization decreases. It seems that these rules can be modeled by a familiar concept of physics as well-known as the *fields of forces* (FOF). FOF is a concept which is utilized in physics to explain the reason of the operation of the universe. The virtual FOF model is approximately simulated by using the concepts of real world fields such as gravitational, magnetical or electrical fields, Kaveh and Talatahari [1].

This chapter utilizes the concept of the FOF model to enhance the performance of the CSS algorithm. To reach such an improved algorithm, the definition of the iteration for the FOF model is altered. Though this change is only performed for the CSS algorithm, however it can be easily utilized for all the above mentioned metaheuristic. It seems the enhanced method opens a new horizon for the concept of time or iteration for the metaheuristics.

In order to investigate the efficiently of the enhanced CSS algorithm, it is used to the optimum configuration design of the structures. The aim of the structural configuration optimization is to obtain optimum locations of the structural joints and suitable cross sections for the structural elements, such that the weight of the structure becomes a minimum. In this type of optimization problems usually a large numbers of design variables are encountered, corresponding to a design space of large dimension, Kaveh et al. [2]. In addition, there are many constraints such as member stresses, buckling stresses and joint displacements, and many local optimums which increase the complexity and difficulty of the problem. Therefore, the configuration optimization is found to be a good field to examine the performance of the new algorithm.

A. Kaveh, *Advances in Metaheuristic Algorithms for Optimal Design of Structures*,
DOI 10.1007/978-3-319-05549-7_5, © Springer International Publishing Switzerland 2014

The remaining sections are organized as follows: In Sect. 5.2, statement of the configuration optimization design of structures is formulated. Fundamental concepts of the fields of forces from physics are presented in Sect. 5.3. The necessary definitions for a FOF-based model are presented in Sect. 5.4. Section 5.5 describes the FOF-based methods as a unified general framework of metaheuristics. An enhanced CSS algorithm is provided in Sect. 5.6. Various examples are studied in Sect. 5.7 and conclusions are derived in Sect. 5.8.

5.2 Formulation of the Configuration Optimization Problems

The goal of configuration optimization is to find the optimal shape of the structure for a given topology. Therefore, decision variables of the problem include the coordinates of certain nodes of the truss (**G**) in addition to the sizing variables for its different members (**A**). The problem can be expressed as follows:

$$\begin{aligned} \text{minimize} \quad & W(\mathbf{A},\mathbf{G}) = \sum_{i=1}^{n} \gamma_i \cdot A_i \cdot L_i \\ \text{subject to :} \quad & \delta_{\min} \le \delta_i \le \delta_{\max} \qquad i = 1,2,\ldots,m \\ & \sigma_{\min} \le \sigma_i \le \sigma_{\max} \qquad i = 1,2,\ldots,n \\ & \sigma_i^b \le \sigma_i \le 0 \qquad i = 1,2,\ldots,ns \\ & A_{i,\min} \le A_i \le A_{i,\max} \qquad i = 1,2,\ldots,ng \\ & G_{i,\min} \le G_i \le G_{i,\max} \qquad i = 1,2,\ldots,m \end{aligned} \tag{5.1}$$

where $W(\mathbf{A},\mathbf{G})$ is the weight of the structure; n is the number of members making up the structure; m denotes the number of nodes; ns is the number of compression elements; ng is the number of groups (number of design variables); γ_i represents the material density of member i; L_i is the length of member i; A_i is the cross-sectional area of member i chosen between $A_{\min}$ and $A_{\max}$; G_i denotes the location of the joints; *min* and *max* are the lower and upper bounds, respectively; σ_i and δ_i are the stress and nodal deflection, respectively; σ_i^b represents the allowable buckling stress in member i when it is in compression.

5.3 Fundamental Concepts of the Fields of Forces

In physics, a field is a physical quantity associated to each point of *space-time*, Gribbin [3]. Space-time is a mathematical model that combines space and time into a single construct and is usually interpreted with space being three-dimensional and time playing the role of the fourth dimension. Particles in the space-time exert field forces which dictate the motion of particles because of carrying the energy. There

are many types of the fields in physics such as temperature fields, air pressure fields, Newtonian gravitational fields, electric fields and magnetic fields, etc.

In physics, it is known that the force field between two charges, two magnetic monopoles, or two masses all follow an inverse square law as

$$\begin{aligned} F_{ij} &= G\frac{m_i m_j}{r_{ij}^2} \\ F_{ij} &= k_e\frac{q_i q_j}{r_{ij}^2} \\ F_{ij} &= U\frac{M_i M_j}{r_{ij}^2} \end{aligned} \tag{5.2}$$

where G, k_e and U are constants; r_{ij} is the distance between two objects; m is the mass of the object; q is the magnitude of charge on the particle and M is the magnetic monopoles strength. According to (5.2), the force between two particles is inversely proportional to the square of the separation distance between them, and proportional to the product of the related magnitudes. Also, the force is directed along the line joining the particles. The magnitude of the field is obtained for particle i, by substituting a unit particle instead of m_j, q_j or M_j in the (5.2) as

$$\begin{aligned} E_{ij} &= G\frac{m_i}{r_{ij}^2} \\ E_{ij} &= K_e\frac{q_i}{r_{ij}^2} \\ E_{ij} &= U\frac{M_i}{r_{ij}^2} \end{aligned} \tag{5.3}$$

As the second example, let us consider an insulating solid sphere of radius a, which has a uniform volume charge density and carries a total charge of q_i. The electric field E_{ij} at a point inside the space can be obtained using the Gauss's law as

$$E_{ij} = k_e\frac{q_i}{a^3}r_{ij} \tag{5.4}$$

The magnitude of the electric field at a point outside the sphere is as defined by (5.3).

The magnitude of the field at a point due to a group of objects is obtained by using the superposition principle as

$$E_j = \sum_{i=1, i\neq j}^{N} E_{ij} \tag{5.5}$$

where N is the total number of objects. In a vector form, it can be expressed as the following

$$\mathbf{E}_j = \sum_{i=1, i\neq j}^{N} E_{ij} \frac{\mathbf{r}_{ij}}{||\mathbf{r}_{ij}||} \tag{5.6}$$

where E_{ij} for the electric fields is given as

$$E_{ij} = \begin{cases} \frac{k_e q_i}{a^3} r_{ij} & \text{if } r_{ij} < a \\ \frac{k_e q_i}{r_{ij}^2} & \text{if } r_{ij} \geq a \end{cases} \tag{5.7}$$

5.4 Necessary Definitions for a FOF-Based Model

Here, some principal and definitions of the FOF-based models are presented as follows:

- *Probe*: Each agent in the optimization algorithm is treated as a particle or probe which can only move in the predefined search space and its location is determined in the search space in the current time and sometime in the previous times. The location of probes is a vector of numbers in which each number represents a dimension of the search-time and the value of the number indicates the value of that parameter.
- *Space-time*: The term of the space-time is used for the search space at a determined time. The dimension of the space-time is equal to the number of the design variables in addition to the time.
- *Time*: In the optimization problem, the *iteration* term is used for the time and thus it can be assumed that the time changes discretely. This means that the time domain is an integer domain and the change of the space-time is performed considering this property.
- *Sources of fields*: In a FOF-based model, there are some sources of fields which can create a virtual field of force and attract the probes toward themselves; however their powers are limited. The sources cannot be located out of the space-time.
- *Effective material*: The power of the field sources is limited by the amount of the effective material. The effective material can be modeled on the amount of the mass for a particle in Newtonian gravitational fields, or the magnitude of the charge in the electric fields. The magnitude of the effective materials may be altered during the optimization process based on the value (or fitness) of the objective.
- *Uniform field*: The points of space-time under the effect of the uniform field can be selected with a uniform probability. In the start of the algorithms, the initial solutions are obtained randomly. This model can be utilized in this condition.
- *Additional instrument*: The field-based model can utilize randomization as an additional instrument. This will change some required values, such as the

location of probes or location or the magnitude of effective material of sources, in a random manner.

5.5 A FOF-Based General Method

Based on the definitions presented in the previous section, here a unified approach is developed which directly utilizes the FOF concept. Before describing the properties of the new algorithm, a pseudo-code as a general form of the FOF-based algorithms is provided as follows:

Step 1: Initialization. For initialization of the algorithm, we have

- The assumptions and definitions are as presented in Sect. 5.4.
- The primary location of the agents must be determined (often obtained randomly using uniform fields).
- The location and the amounts of the effective material for the sources must also be determined.

Step 2: Solution construction.

- In this step, each agent moves toward the space-time and finds a place using affected fields of forces created by the sources. The rules of moving is dependent on the type of the algorithm, however, all algorithms use the abilities of the randomization in this stage.

Step 3: Source updating.

- The amounts of the effective material for the sources must be updated; and/or
- The new locations of the sources must be obtained.

Step 4: Terminating criterion control.

- Steps 2 and 3 are repeated until a terminating criterion is satisfied. Though the order of steps 2 and 3 may be changed, this cannot make a problem in generality of this pseudo-code.

Figure 5.1 summarizes the flowchart of the general FOF-based model.

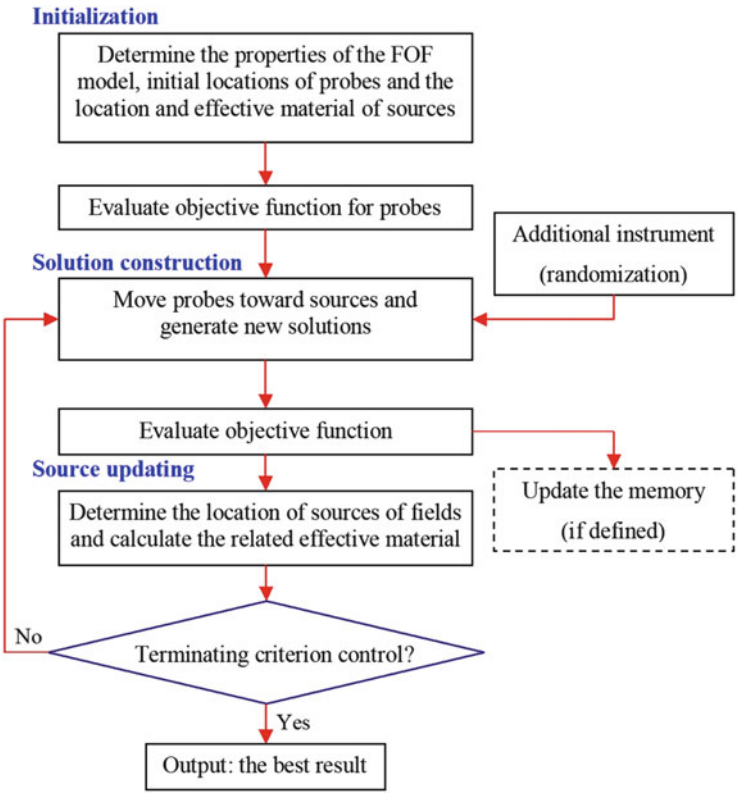

Fig. 5.1 Flowchart of the FOF-based model [1]

5.6 An Enhanced Charged System Search Algorithm for Configuration Optimization

5.6.1 Review of the Charged System Search Algorithm

The Charged System Search (CSS) algorithm is proposed by Kaveh and Talatahari [4] and utilized for size optimization of the structures (Kaveh and Talatahari [5, 6]. The pseudo-code for the CSS algorithm is summarized as follows:

Step 1: Initialization.

- The initial positions of CPs are determined randomly in the search space and the initial velocities of charged particles are assumed to be zero. A memory, called Charged Memory (CM) is considered to save the best results.

Step 2: Solution construction.

Forces determination. Each CP is a source of the field. Based on the field of CPs, the force vectors for all CPs are calculated as

$$\mathbf{F}_j = q_j \sum_{i,i\neq j} \left(\frac{q_i}{a^3} r_{ij} \cdot i_1 + \frac{q_i}{r_{ij}^2} \cdot i_2 \right) p_{ij}\left(\mathbf{X}_i - \mathbf{X}_j\right) \quad \begin{cases} j = 1, 2, \ldots, N \\ i_1 = 1, i_2 = 0 \Leftrightarrow r_{ij} < a \\ i_1 = 0, i_2 = 1 \Leftrightarrow r_{ij} \geq a \end{cases} \tag{5.8}$$

where $\mathbf{F}_j$ is the resultant force acting on the jth CP. $\mathbf{X}_i$ and $\mathbf{X}_j$ are the positions of the ith and jth CPs, respectively. q_i is the effective martial of the ith CP and is defined considering the quality of its solution. The separation distance r_{ij} between two charged particles is defined as follows

$$r_{ij} = \frac{\left|\left|\mathbf{X}_i - \mathbf{X}_j\right|\right|}{\left|\left|\left(\mathbf{X}_i + \mathbf{X}_j\right)/2 - \mathbf{X}_{best}\right|\right| + \varepsilon} \tag{5.9}$$

where $\mathbf{X}_{best}$ is the position of the best current CP, and ε is a small positive number. In (5.8), p_{ij} is the probability of moving each CP toward the others and is equal to

$$p_{ij} = \begin{cases} 1 & \dfrac{fit(i) - fitbest}{fit(j) - fit(i)} > rand \text{ or } fit(j) > fit\left(i\right) \\ 0 & \text{else} \end{cases} \tag{5.10}$$

New position creation. Each CP moves to the new position and find the velocities as

$$\mathbf{X}_{j,new} = rand_{j1} \cdot k_a \cdot \frac{\mathbf{F}_j}{m_j} \cdot \Delta t^2 + rand_{j2} \cdot k_v \cdot \mathbf{V}_{j,old} \cdot \Delta t + \mathbf{X}_{j,old} \tag{5.11}$$

$$\mathbf{V}_{j,new} = \frac{\mathbf{X}_{j,new} - \mathbf{X}_{j,old}}{\Delta t} \tag{5.12}$$

where k_a is the acceleration coefficient; k_v is the velocity coefficient to control the influence of the previous velocity; and $rand_{j1}$ and $rand_{j2}$ are two random numbers uniformly distributed in the range of (0, 1). Then, the related objective functions for the agents are calculated.

Step 3: CM (sources) updating.

- If some new CP vectors are better than the worst ones in the CM, in terms of their objective function values, the better vectors are included in the CM and the worst ones are excluded from the CM.

Step 4: Terminating criterion control.

- Steps 2 and 3 are reiterated until a terminating criterion is satisfied.

5.6.2 An Enhanced Charged System Search Algorithm

One of the assumptions, we described in Sect. 5.4 for establishing a FOF-based model of metaheuristics is that the time alters discretely. This means that all alterations in space-time are performed when all agents have created their solutions. For example, in the CSS algorithm, when the calculations of the amount of forces are completed for all CPs, the new locations of agents are determined (step 2). Also CM updating is fulfilled after moving all CPs to their new locations. All these conform to discrete time concept. In the optimization problems, this is known as iteration. In other words, the modification of the space-time for the multi-agent algorithms is often performed when an iteration is completed and the new iteration is not started yet. Here, we ignore this assumption for the CSS algorithm and therefore an enhanced CSS is presented. In the enhanced CSS, time changes continuously and after creating just one solution, all updating processes are performed. Using this enhanced CSS, the new position of each agent can affect on the moving process of the subsequent CPs while in the standard CSS unless an iteration is completed, the new positions are not utilized. Based on this inference, the enhanced CSS is as follows:

Step 1: Initialization.

- This step is similar to the one defined previously. The initial positions and velocities of CPs as well as the CM are initialized. A number associated to each CP is considered.

Step 2: Solution construction.

- Forces determination. The force vector for the jth CP is calculated as (5.8).
- New position creation. Each CP moves to the new position as defined in (5.11) and (5.12). It should be noted that in order to determine the location of each CP using (5.11), the recent location of the previous agents is utilized instead of the previous ones and this leads to the use of the pervious information directly after their generation. After moving the CP to its new position, the objective function is evaluated.

Step 3: CM (sources) updating.

- If the new CP vector is better than the worst one in the CM, it is included in the CM.

Step 4: Terminating criterion control.

- Steps 2 and 3 are repeated until a terminating criterion is satisfied.

5.7 Design Examples

This section presents some numerical design examples to illustrate the efficiency of the new algorithm. The two first examples chosen from literature are known as the benchmark examples in the field of the configuration optimization problem containing an 18-bar planar truss and a 25-bar space truss. Since the largeness of the examples does not make much difference on the search space, the number of CPs is set to 20 for all the studied examples. The result of the enhanced CSS is obtained and compared to some other numerical methods. The last example is solved by the primary and enhanced CSS to identify the superiority of the new approach. The algorithms are coded in Matlab and a direct stiffness method is utilized to analysis the structures.

5.7.1 An 18-Bar Planar Truss

The initial configuration of an 18-bar cantilever planar truss is shown in Fig. 5.2 which has been previously analyzed by many authors to obtain the optimal design. The material density is 2,768 kg/m^3 (0.1 lb/in^3) and the modulus of elasticity is 68,950 MPa (10,000 ksi). The members are subjected to stress limitations of $\pm$137.9 MPa ($\pm$20 ksi). Also, an Euler bucking compressive stress limitation is imposed for truss member i, according to

$$\sigma_i^b = \frac{-kEA_i}{L_i^2} \tag{5.13}$$

where E is the modulus of elasticity; and k is a constant determined from the cross-sectional geometry and here it is equal to 4.

Vertical downward loads of –89 N (–20 kips) at nodes 1, 2, 4, 6 and 8 are considered. The cross-sectional areas of the members are linked into four groups, as follows:

1. A_1, A_4, A_8, A_{12}, A_{16};
2. A_2, A_6, A_{10}, A_{14}, A_{18};
3. A_3, A_7, A_{11}, A_{15}; and
4. A_5, A_9, A_{13}, A_{17}.

The lower nodes, 3, 5, 7, and 9, are allowed to move in any direction in the x-y plane. Thus, there are 12 design variables which include four sizing and eight coordinate variables. Side constraints for geometry variable are as follows:

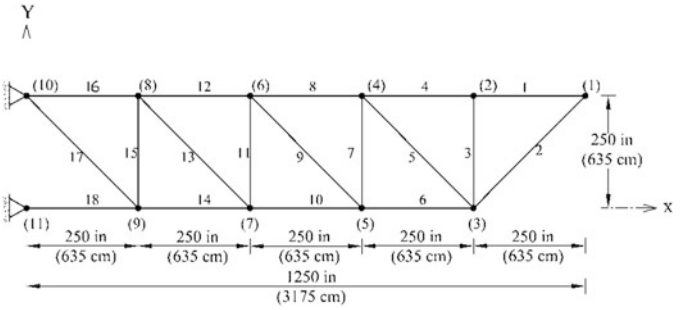

Fig. 5.2 The initial geometry of the 18-bar planar truss [1]

$$-571.5 \text{ cm } (-225 \text{ in}) \leq y_3, y_5, y_7, y_9 \leq 622.3 \text{ cm } (245 \text{ in});$$
$$1,968.5 \text{ cm } (775 \text{ in}) \leq x_3 \leq 3,111.5 \text{ cm } (1,225 \text{ in});$$
$$1,333.5 \text{ cm } (525 \text{ in}) \leq x_5 \leq 2,476.5 \text{ cm } (975 \text{ in});$$
$$698.5 \text{ cm } (275 \text{ in}) \leq x_7 \leq 1,841.5 \text{ cm } (725 \text{ in});$$
$$63.5 \text{ cm } (25 \text{ in}) \leq x_9 \leq 1,206.5 \text{ cm } (475 \text{ in}).$$

In this example, the continuous size variables are used and the allowable bounds on the cross-sectional areas are 22.58−129.03 cm^2 (3.5–20 in^2).

Table 5.1 presents the best solution vectors from the CSS and other methods. Imai and Schmit [7] and Felix [8] used the mathematical methods to find optimum results which are equal to 20,763.8 and 25,412.7 N, respectively. GA-based approaches (many authors including Rahami et al. [9]) are also used to solve this example. The weights are 20,251.9, 20,158.9, 20,536.5, 20,105.9 and 20,067.7 N, respectively. Zheng et al. [10] used a GP algorithm and find a truss with the weight of 21,377.7 N. The HS result (Lee and Geem [11]) is equal to 20,086.4 N. The best CSS design results in a truss weighing 20,048.7 N, which is the best among the other approaches. Among the GA-based methods, the result of the algorithm proposed by Rahami et al. [9] is the best and obtained after 8,000 structural analyses and the HS algorithm (Lee and Geem [11]) converges to the optimum point after 24,805 analyses while CSS needs 4,000 FEM analyses to reach the result. Figure 5.3 shows the convergence history for the CSS results and Fig. 5.4 displays optimal geometry of the 18-bar truss obtained by the CSS algorithm. Also, a comparison between the allowable and existing stress values in elements for the CSS result is shown in Fig. 5.5. In this figure, the dashed bold lines indicates the limits of the stress constraints and it is equal to 20.00 ksi when the type of stress is tension and the limit for elements in compression are obtained by using (5.13). The maximum tension stress is equal to 20.00 ksi in the 16th element and the maximum compression stress is −17.0152 ksi in the last element while the allowable buckling stress equals to −17.0154 ksi.

Table 5.1 Performance comparison for the 18-bar truss

Design variables	Imai and Schmit [7]	Rahami et al. [8]	Zheng et al. [9]	Lee and Geem [10]	Kaveh and Talatahari [1]
A_1	11.24	12.554	12.040	12.65	12.476
A_2	15.68	18.029	17.847	7.22	17.831
A_3	7.93	5.114	7.969	6.17	5.277
A_5	6.49	3.571	4.726	3.55	3.726
x_3	891.10	912.969	944.882	903.10	911.698
y_3	143.60	188.067	150.000	174.30	185.788
x_5	608.20	646.450	664.961	630.30	643.917
y_5	105.40	150.617	122.441	136.30	147.640
x_7	381.70	416.624	414.961	402.10	414.18
y_7	57.10	102.526	77.559	90.50	98.507
x_9	181.00	204.282	192.520	195.30	202.444
y_9	−3.20	32.653	17.323	30.60	30.557
Best Weight (N)	20,763.8	20,067.7	21,377.7	20,086.4	20,048.7

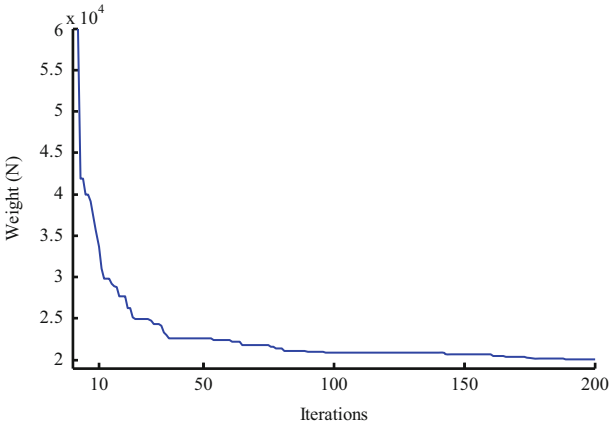

Fig. 5.3 Convergence history of the 18-bar truss for the CSS algorithm [1]

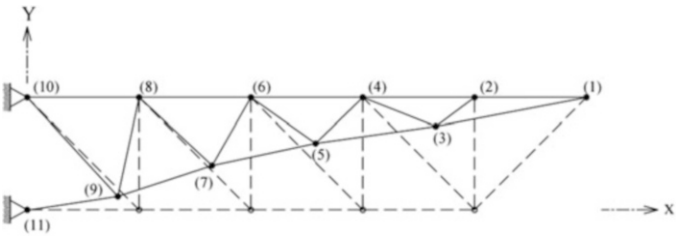

Fig. 5.4 Optimal geometry of the 18-bar truss obtained by the CSS algorithm

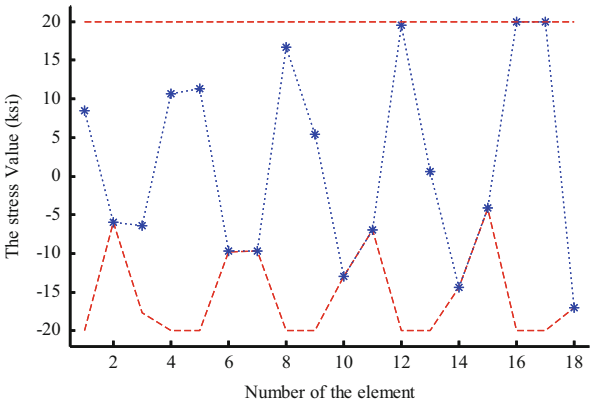

Fig. 5.5 Comparison of the allowable and existing stress values for the 18-bar truss using the CSS algorithm [1]

5.7.1.1 A 25-Bar Spatial Truss

Figure 5.6 shows the initial topology of a 25-bar spatial truss structure. This example has been frequently studied in sizing and configuration optimization using mathematical approaches. The material density is 2,768 kg/m^3 (0.1 lb/in^3) and the modulus of elasticity is 68,950 MPa (10,000 ksi). Two cases are considered:

Case 1: The cross-sectional are continuous and the bounds on the member cross-sectional areas are 0.065–6.45 cm^2 (0.01–1.0 in^2). The load condition for this case is indicated in Table 5.2. All members are constrained to 275.6 MPa (40 ksi) in both tension and compression. In addition, all members stresses are constrained to the Euler buckling stress, as given by (5.13) with the buckling constant $k = 39.274$ corresponding to tubular members with a nominal diameter-to-thickness ratio 100. Case 2: For the second case the discrete set of cross sections is considered. The list of the available profiles are as: {0.645I (I = 1,. . ., 26), 18.064, 19.355, 20.645, 21.935} cm^2 or {0.1I (I = 1,. . ., 26), 2.8, 3.0, 3.2, 3.4} in^2 which has thirty discrete values. Table 5.3 presents the load condition for this case. The constraints are the nodal displacements (no more than 0.89 cm or 0.35 in) in all directions of the coordinate system for the nodes and the stress constraint (no more than ±275.6 MPa or ±40 ksi) for all members.

The structure was required to be doubly symmetric about x- and y- axe; this condition grouped the truss members as follows: (1) A_1; (2) A_2–A_5; (3) A_6–A_9; (4) A_{10}–A_{11}; (5) A_{12}–A_{13}; (6) A_{14}–A_{17}; (7) A_{18}–A_{21}; and (8) A_{22}–A_{23}. For the configuration optimization, the geometric variables are selected as coordinates x_4, y_4, z_4, x_8, and y_8, with symmetry required in x–z and y–z planes. The side constraints for the geometric variables in the second case are as follows:

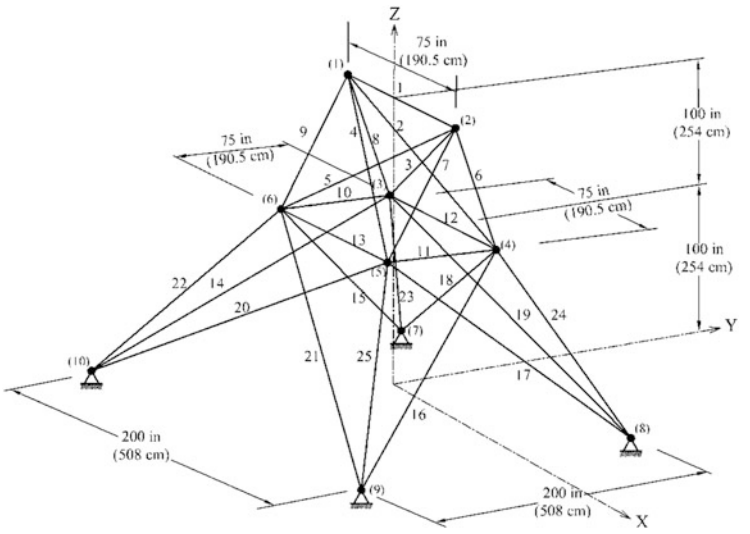

Fig. 5.6 The initial geometry of the 25-bar spatial truss

Table 5.2 Loading conditions for the 25-bar spatial truss (Case 1)

	Case 1			Case 2		
Node	P_X *kips(kN)*	P_Y *kips(kN)*	P_Z *kips(kN)*	P_X *kips(kN)*	P_Y *kips(kN)*	P_Z *kips(kN)*
1	0.0	20.0 (89)	−5.0 (22.25)	1.0 (4.45)	10.0 (44.5)	−5.0 (22.25)
2	0.0	−20.0 (89)	−5.0 (22.25)	0.0	10.0 (44.5)	−5.0 (22.25)
3	0.0	0.0	0.0	0.5 (2.22)	0.0	0.0
6	0.0	0.0	0.0	0.5 (2.22)	0.0	0.0

Table 5.3 Loading conditions for the 25-bar spatial truss (Case 2)

Node	P_X *kips(kN)*	P_Y *kips(kN)*	P_Z *kips(kN)*
1	1.0 (4.45)	−10.0 (44.5)	−10.0 (44.5)
2	0.0	−10.0 (44.5)	−10.0 (44.5)
3	0.5 (2.22)	0.0	0.0
6	0.6 (2.67)	0.0	0.0

$$
\begin{aligned}
&50.8\ \text{cm}\ (20\ \text{in}) \le x_4 \le 152.4\ \text{cm}\ (60\ \text{in});\\
&101.6\ \text{cm}\ (40\ \text{in}) \le y_4 \le 203.2\ \text{cm}\ (80\ \text{in});\\
&228.6\ \text{cm}\ (90\ \text{in}) \le z_4 \le 330.2\ \text{cm}\ (130\ \text{in});\\
&101.6\ \text{cm}\ (40\ \text{in}) \le x_8 \le 203.2\ \text{cm}\ (80\ \text{in}); \quad \text{and}\\
&254\ \text{cm}\ (100\ \text{in}) \le y_8 \le 355.6\ \text{cm}\ (140\ \text{in}).
\end{aligned}
$$

Considering Case 1, this example was solved by different methods. Vanderplaats and Moses [12] and Felix [7] used mathematical methods and Yang [13], Soh and Yang [14] and Yang and Soh [15] utilized GA-based methods. In addition, Zheng et al. [10] used a GP algorithm to solve this example. The corresponding weight of

Table 5.4 Performance comparison for the 25-bar truss (Case 1)

Design variables	Vanderplaats and Moses [12]	Felix [8]	Soh and Yang [14]	Zheng et al. [10]	Kaveh and Talatahari [1]
A_1	0.08	0.07	0.58	0.58	0.11
A_2	2.67	3.14	2.84	4.97	2.33
A_3	5.43	5.39	5.81	4.39	5.64
A_4	0.21	0.16	0.32	0.52	0.25
A_5	0.65	0.79	0.71	0.065	0.70
A_6	0.78	0.54	1.36	0.1	0.80
A_7	4.77	4.50	4.52	3.1	5.34
A_8	3.57	3.54	3.61	3.1	3.69
x_4	54.6	60.20	55.8	32.0	51.57
y_4	122.7	125.2	110.7	222.0	96.00
z_4	254.8	248.2	246.0	254.0	262.77
x_8	54.1	69.9	35.9	159.0	45.65
y_8	244.7	244.9	206.1	254.0	198.71
Best Weight (N)	593.8	571.6	590.9	583.8	567.5

these methods are 593.8, 571.6, 610.3, 590.9, 584.0, 583.8 N, respectively while it is 567.5 for the solution vector of the CSS algorithm. Table 5.4 presents some of the best results of these algorithms. CSS need 4,000 analyses to reach the optimum result as shown in Fig. 5.7.

For Case 2, Wu and Chow [16], Kaveh and Kalatjari [17] and Rahami et al. [9] used GA-based algorithms. Lee and Geem [11] used a harmony search algorithm. The result of the CSS algorithm is 528.58 N which is 14.6 %, 4.35 %, 1.08 % and 4.12 % less than the previous studies, respectively. Table 5.5 summarizes the design vectors as well as the weight of the results obtained by different algorithms. Maximum displacement for the design of the CSS algorithm is 0.888 cm which is less than its maximum limit. Also, its maximum stress value is equal −126.4 MPa and so it can be seen that the displacement constraint is dominant in this case. The optimum configurations for two cases are shown in Fig. 5.8.

In addition to previous cases, when the range of cross-sectional areas varies from 0.01 to 3.4 in^2 (0.6452 cm^2 to 21.94 cm^2) and only size optimization is considered, a statistical study on the results of different algorithms is performed. The detailed information for constraint conditions is presented in Kaveh and Talatahari [4]. Table 5.6 compares the performance of the presented algorithm and other metaheuristic algorithms. Obviously, the enhanced CSS performs better than other algorithms when the best weight, the average weight or the standard deviation are compared.

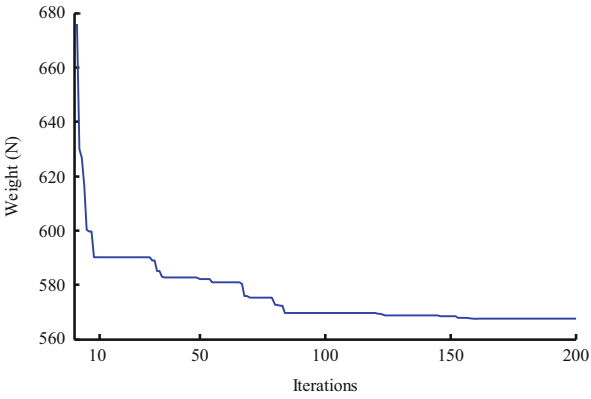

Fig. 5.7 Convergence history of the 25-bar spatial truss for the CSS algorithm [1]

Table 5.5 Performance comparison for the 25-bar truss (Case 2)

Design variables	Wu and Chow [16]	Kaveh and Kalatjari [17]	Rahami et al. [9]	Lee and Geem [11]	Kaveh and Talatahari [1]
A_1	0.1	0.1	0.1	0.2	0.1
A_2	0.2	0.1	0.1	0.1	0.1
A_3	1.1	1.1	1.1	0.9	0.9
A_4	0.2	0.1	0.1	0.1	0.1
A_5	0.3	0.1	0.1	0.1	0.1
A_6	0.1	0.1	0.1	0.1	0.1
A_7	0.2	0.1	0.2	0.2	0.1
A_8	0.9	1.0	0.8	1.0	1.0
x_4	41.07	36.23	33.049	31.88	36.762
y_4	53.47	58.56	53.566	83.57	56.920
z_4	124.6	115.59	129.909	126.35	124.863
x_8	50.8	46.46	43.783	40.43	49.767
y_8	131.48	127.95	136.838	130.64	136.757
Best Weight (N)	605.85	551.58	534.30	550.55	528.58

5.7.1.2 A 120-Bar Dome Truss

The design of a 120-bar dome truss, shown in Fig. 5.9, is considered as the last example to compare the practical capability of the standard and enhanced CSS algorithms. This dome is utilized in literature to find size optimum design, however here the aim is to obtain the optimal sizing and configuration variables. The modulus of elasticity is 210,000 MPa (30,450 ksi), and the material density is 7971.810 kg/m^3 (0.288 lb/in^3). The yield stress of steel is taken as 400 MPa (58.0 ksi). The dome is considered to be subjected to vertical loading at all the unsupported joints. These loads are taken as −60 kN (−13.49 kips) at node 1, −30 kN (−6.744 kips) at nodes 2 through 14, and −10 kN (−2.248 kips) at the rest of the nodes. The minimum cross-sectional area of all members is 2 cm^2

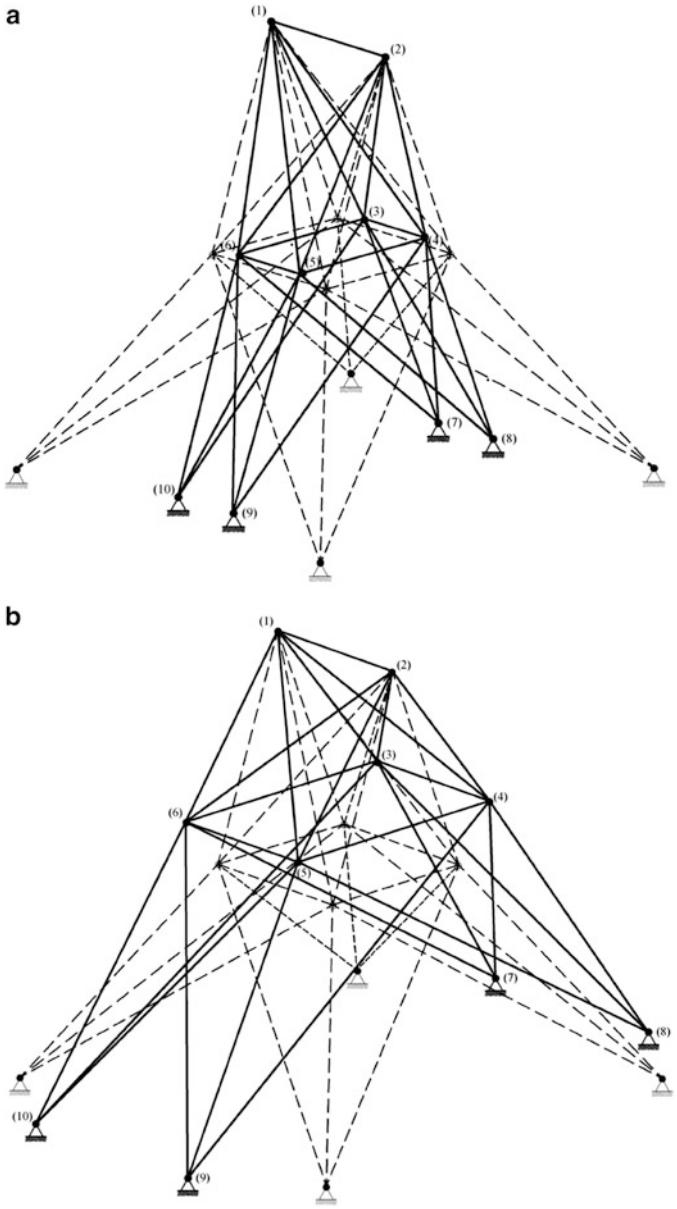

Fig. 5.8 Optimal geometry of the 25-bar truss obtained by the CSS algorithm [1]: (**a**) Case 1; (**b**) Case 2

(0.775 in^2) and the maximum cross-sectional area is taken as 129.03 cm^2 (20.0 in^2). Due to the symmetry of the structure, the geometric variables are selected as the height of the rings and the crown (three geometric variables). The geometric

Table 5.6 Performance comparison for the 25-bar spatial truss (pure size optimization)

	Rajeev and Krishnamoorthy	Schutte and Groenwold	Lee and Geem	Kaveh and Talatahari				
	GA [17]	PSO [18]	HS [10]	PSACO [19]	HPSACO [19]	HBB–BC [20]	CSS [4]	Kaveh and Talatahari [1]
Best Weight(lb)	546	545.21	544.38	545.04	544.99	545.16	545.10	544.92
Average Weight (lb)	N/A	546.84	N/A	N/A	545.52	545.66	545.58	545.42
Std Dev(lb)	N/A	1.478	N/A	N/A	0.315	0.367	0.412	0.375

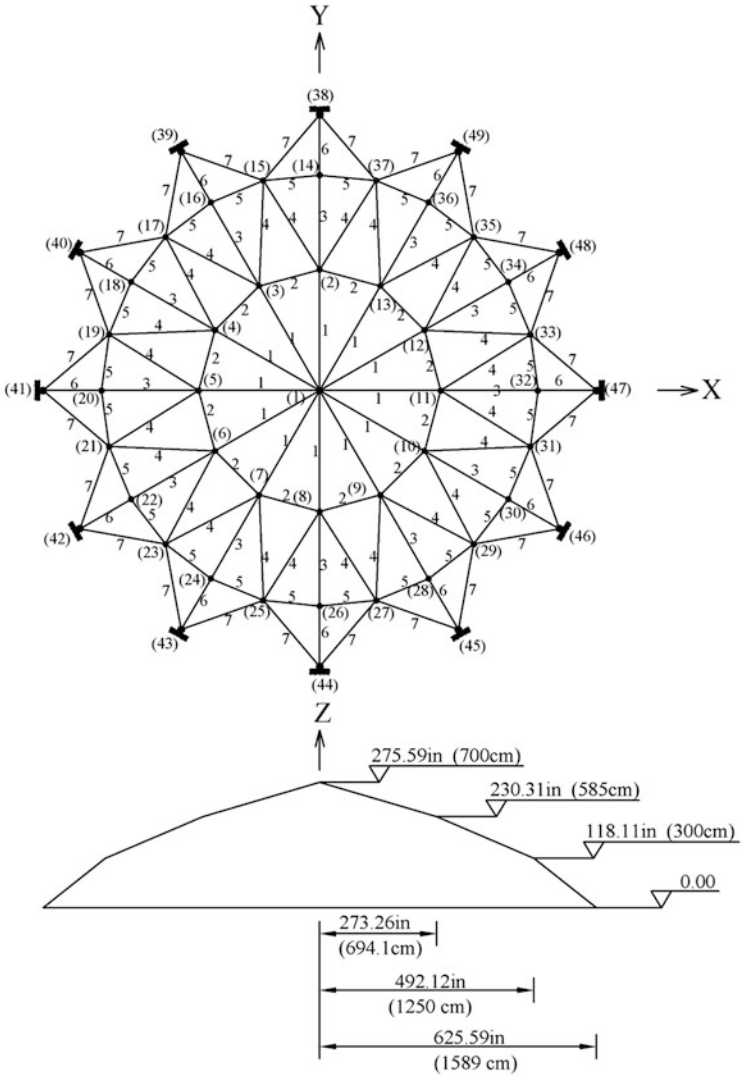

Fig. 5.9 The initial geometry of the 120-bar dome shaped truss

variables are allowed to move 0.50 m, based on their initial value. The stress constraints of the structural members are calculated as per AISC [21] specifications. Besides, the displacements of all nodes in any direction are limited to a maximum value of 5 mm (0.1969 in).

Table 5.7 compares the result of the standard CSS and enhanced CSS. When pure size optimization is considered, the enhanced CSS can find a better result in a less number of analyses. The enhanced CSS needs 4,000 analyses to find the optimum result while it is 7,000 for the standard CSS as reported by the authors

Table 5.7 Performance comparison for the 120-bar truss

	Pure size optimization		Configuration optimization	
Design variables	Standard CSS (Kaveh and Talatahari [5])	Kaveh and Talatahari [1]	Standard CSS	Kaveh and Talatahari [1]
A_1	3.027	3.032	3.103	3.235
A_2	14.606	15.335	7.328	4.875
A_3	5.044	4.767	4.350	4.303
A_4	3.139	3.030	2.731	2.764
A_5	8.543	8.252	1.719	2.438
A_6	3.367	3.723	3.739	3.637
A_7	2.497	2.502	2.452	2.505
z_1	–	–	286.505	259.569
z_2	–	–	209.295	207.017
z_3	–	–	107.271	106.556
Best weight (N)	147,912	147,537	100,984	98,815
Average weight(N)	151,865	149,862	102,723	101,156
Std Dev(N)	2,963	2,456	3,365	2,884

(Kaveh and Talatahari [5]). Figure 5.10 shows the convergence history for these two CSS-based algorithms. For the configuration optimization, the enhanced CSS algorithm can find a design with weight of 98,815 N while it is 100,984 N for the standard CSS. Adding three geometry design variables to the problem saves the structural martial more than 33 %.

5.8 Discussion

A model is developed to improve the performance of metaheuristics utilizing the concept of the virtual fields of forces. This general framework of the FOF-based algorithm contains four steps:

- Initialization, where considers some probes and sources and determines the first location of the probes (initial solutions) using a uniform field. Also, the amounts of the effective material for the sources based on the utilized approach are determined.
- Solution construction, in which each probe moves toward the sources and finds a place as a new solution.
- Source updating, where the location of the sources and/or the amounts of the effective material are updated to direct the search process toward an optimum point.
- Terminating criterion control, which determines the end time of the search process.

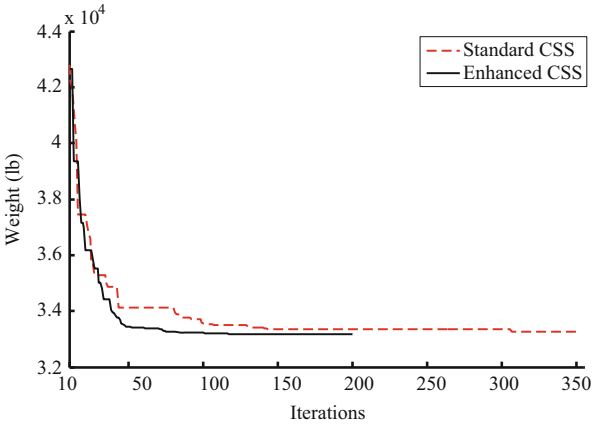

Fig. 5.10 Convergence history of the 120-bar dome shaped truss for the CSS-based algorithms [1]

Using this model, an enhanced CSS algorithm is developed. Although the CSS algorithm uses the concept of the FOF model directly, however considering the continuous space-time for this algorithm improves its efficiency. In this algorithm, time changes continuously and after creating just one solution, all updating processes are performed. Using the enhanced CSS, the new position of each agent can affect the moving process of the subsequent CPs, while in the standard CSS until the completion the iteration, the new positions are not utilized and this change improves the performance of the algorithm.

In this chapter, the enhanced CSS is utilized to determine the optimum configuration design of the truss structures. For this purpose, three examples are considered and the results are compared to the results of different algorithms and the standard CSS. The results indicate the efficiency of the enhanced CSS for determining the optimum design of structures.

It can be postulated that further improvement of metaheuristics can be achieved by changing the definition and/or application of some concepts used in the FOF model. Considering the continuous space-time is the first result of such an improvement. Utilizing what is defined as the continuous space-time for other metaheuristic algorithms opens a new horizon for the concept of the time or iteration for metaheuristics, and it is expected this continuous space-time can be expended for other algorithms. As a future work, one can change the manner of using other concepts of the FOF model to reach to some better optimization methods similar to the enhanced CSS which alters the way of using the space-time term.

References

1. Kaveh A, Talatahari S (2011) An enhanced charged system search for configuration optimization using the concept of fields of forces. Struct Multidiscip Optim 43(3):339–351
2. Kaveh A, Farahmand Azar B, Talatahari S (2008) Ant colony optimization for design of space trusses. Int J Space Struct 23(3):167–181

3. Gribbin J (1998) Particle physics from A to Z. Weidenfeld & Nicolson, London
4. Kaveh A, Talatahari S (2010) A novel heuristic optimization method: charged system search. Acta Mech 213(3–4):267–286
5. Kaveh A, Talatahari S (2010) Optimal design of skeletal structures via the charged system search algorithm. Struct Multidiscip Optim 41(6):893–911
6. Kaveh A, Talatahari S (2010) Charged system search for optimum grillage systems design using the LRFD-AISC code. J Constr Steel Res 66(6):767–771
7. Imai K, Schmit LA (1981) Configuration optimisation of trusses. J Struct Div ASCE 107:745–756
8. Felix JE (1981) Shape optimization of trusses subjected to strength, displacement, and frequency constraints. M.Sc. thesis, Naval Postgraduate School
9. Rahami H, Kaveh A, Gholipoura Y (2008) Sizing, geometry and topology optimization of trusses via force method and genetic algorithm. Eng Struct 30:2360–2369
10. Zheng QZ, Querin OM, Barton DC (2006) Geometry and sizing optimization of discrete structure using the genetic programming method. Struct Multidiscip Optim 231:452–461
11. Lee KS, Geem ZW (2004) A new structural optimization method based on the harmony search algorithm. Comput Struct 82:781–798
12. Vanderplaats GN, Moses F (1972) Automated design of trusses for optimum geometry. J Struct Div, ASCE 98:671–690
13. Yang JP (1996) Development of genetic algorithm-based approach for structural optimization. Ph.D. Thesis, Nanyang Technology University, Singapore
14. Soh CK, Yang JP (1996) Fuzzy controlled genetic algorithm for shape optimization. J Comput Civil Eng, ASCE 10(2):143–150
15. Yang JP, Soh CK (1997) Structural optimization by genetic algorithms with tournament selection, J Comput Civil Eng. ASCE 11(3):195–200
16. Wu SJ, Chow PT (1995) Integrated discrete and configuration optimization of trusses using genetic algorithms. Comput Struct 55(4):695–702
17. Kaveh A, Kalatjari V (2004) Size/geometry optimization of trusses by the force method and genetic algorithm. Z Angew Math Mech 84(5):347–357
18. Rajeev S, Krishnamoorthy CS (1997) Genetic algorithms based methodologies for design optimisation of trusses. J Struct Eng, ASCE 123:350–358
19. Schutte JF, Groenwold AA (2003) Sizing design of truss structures using particle swarms. Struct Multidiscip Optim 25:261–269
20. Kaveh A, Talatahari S (2009) Hybrid algorithm of harmony search, particle swarm and ant colony for structural design optimization, Chapter 5 of a book entitled: Harmony search algorithms for structural design, Springer-Verlag Berlin Heidelberg
21. American Institute of Steel Construction (AISC) (1989) Manual of steel construction-allowable stress design, 9th edn. AISC, IL, Chicago
22. Kaveh A, Talatahari S (2009) Size optimization of space trusses using big bang–big crunch algorithm. Comput Struct 87(17–18):1129–1140

Chapter 6
Dolphin Echolocation Optimization

6.1 Introduction

Nature has provided inspiration for most of the man-made technologies. Scientists believe that dolphins are the second to humans in smartness and intelligence. Echolocation is the biological sonar used by dolphins and several kinds of other animals for navigation and hunting in various environments. This ability of dolphins is mimicked in this chapter to develop a new optimization method. There are different metaheuristic optimization methods, but in most of these algorithms parameter tuning takes a considerable time of the user, persuading the scientists to develop ideas to improve these methods. Studies have shown that metaheuristic algorithms have certain governing rules and knowing these rules helps to get better results. Dolphin Echolocation takes advantages of these rules and outperforms many existing optimization methods, while it has few parameters to be set. The new approach leads to excellent results with low computational efforts [1].

Dolphin echolocation is a new optimization method which is presented in this chapter. This method mimics strategies used by dolphins for their hunting process. Dolphins produce a kind of voice called *sonar* to locate the target, doing this dolphin change sonar to modify the target and its location. Dolphin echolocation is depicted in Fig. 6.1. This fact is mimicked here as the main feature of the new optimization method.

6.2 Dolphin Echolocation in Nature

The term "echolocation" was initiated by Griffin [2] to describe the ability of flying bats to locate obstacles and preys by listening to echoes returning from high-frequency clicks that they emitted. Echolocating animals include some mammals and a few birds. The best studied echolocation in marine mammals is that of the bottlenose dolphins, Au [3].

A. Kaveh, *Advances in Metaheuristic Algorithms for Optimal Design of Structures*, 157
DOI 10.1007/978-3-319-05549-7_6, © Springer International Publishing Switzerland 2014

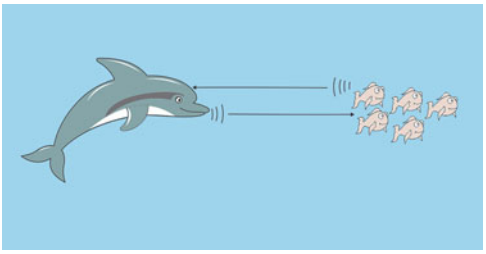

Fig. 6.1 A real dolphin catching its prey [1]

A dolphin is able to generate sounds in the form of clicks. Frequency of these clicks is higher than that of the sounds used for communication and differs between species. When the sound strikes an object, some of the energy of the sound-wave is reflected back towards the dolphin. As soon as an echo is received, the dolphin generates another click. The time lapse between click and echo enables the dolphin to evaluate the distance from the object; the varying strength of the signal as it is received on the two sides of the dolphin's head enabling him to evaluate the direction. By continuously emitting clicks and receiving echoes in this way, the dolphin can track objects and home in on them, May [4]. The clicks are directional and are for echolocation, often occurring in a short series called a click train. The click rate increases when approaching an object of interest [3].

Though bats also use echolocation, however, they differ from dolphins in their sonar system. Bats use their sonar system at short ranges of up to approximately 3–4 m, whereas dolphins can detect their targets at ranges varying from a few tens of meters to over a hundred meters. Many bats hunt for insects that dart rapidly to-and-fro, making it very different from the escape behavior of a fish chased by dolphin. The speed of sound in air is about one fifth of that of water, thus the information transfer rate during sonar transmission for bats is much shorter than that of the dolphins. These and many other differences in environment and prey require totally different types of sonar system, which naturally makes a direct comparison difficult [3, 5].

6.3 Dolphin Echolocation Optimization

6.3.1 Introduction to Dolphin Echolocation

Regarding an optimization problem, it can be understood that echolocation is similar to optimization in some aspects; the process of foraging preys using echolocation in dolphins is similar to finding the optimum answer of a problem.

As mentioned in the previous part, dolphins initially search all around the search space to find the prey. As soon as a dolphin approaches the target, the animal restricts its search, and incrementally increases its clicks in order to concentrate on the location.

The method simulates dolphin echolocation by limiting its exploration proportional to the distance from the target. For making the relationship much clear, consider an optimization problem. Two stages can be identified: in the first stage the algorithm explores all around the search space to perform a global search, therefore it should look for unexplored regions. This task is carried out by exploring some random locations in the search space, and in the second stage it concentrates on investigation around better results achieved from the previous stage. These are obvious inherent characteristics of all metaheuristic algorithms. An efficient method is presented in [6] for controlling the value of the randomly created answers in order to set the ratio of the results to be achieved in phase 1 to phase 2.

By using Dolphin Echolocation (DE) algorithm, the user would be able to change the ratio of answers produced in phase 1 to the answers produces in phase 2 according to a predefined curve. In other words, global search, changes to a local one gradually in a user defined style.

The user defines a curve on which the optimization convergence should be performed, and then the algorithm sets its parameters in order to be able to follow the curve. The method works with the likelihood of occurrence of the best answer in comparison to the others. In other words, for each variable there are different alternatives in the feasible region, in each loop the algorithm defines the possibility of choosing the best so far achieved alternative according to the user determined convergence curve. By using this curve, the convergence criterion is dictated to the algorithm, and then the convergence of the algorithm becomes less parameter dependent. The curve can be any smooth ascending curve but there are some recommendations for it, which will be discussed later.

Previously it has been shown that there is a unified method for parameter selection in metha-heuristics [6]. In the latter paper, an index called the convergence factor was presented. A *Convergence Factor* (*CF*) is defined as the average possibility of the elitist answer. As an example, if the aim is to devote some steel profiles to a structure that has four elements, then in the first step, frequency of modal profile of each element should be defined. *CF* is the mean of these frequencies. Table 6.1 illustrates an example of calculating the *CF* for a structure containing four elements.

6.3.2 Dolphin Echolocation Algorithm

Before starting optimization, search space should be sorted using the following rule:

Search Space Ordering For each variable to be optimized during the process, sort alternatives of the search space in an ascending or descending order. If alternatives include more than one characteristic, perform ordering according to the most important one. Using this method, for variable j, vector A_j of length LA_j is created which contains all possible alternatives for the j^{th} variable putting these vectors next

Table 6.1 An example for calculation of the CF [6]

	Element 1	Element 2	Element 3	Element 4
Answer 1	5	41	22	15
Answer 2	3	36	22	17
Answer 3	4	39	25	16
Answer 4	3	42	22	17
Answer 5	3	41	22	19
Modal answer	3	41	22	17
Frequency of the modal answer	3	2	4	2
Proportion of the modal answer among all answers	60 %	40 %	80 %	40 %
CF	55 %			

to each other , as the columns of a matrix, the $Matrix\ Alternatives_{MA*NV}$ is created, in which MA is $\max(LA_j)_{j=1:NV}$; with NV being the number of variables.

Moreover, a curve according to which the convergence factor should change during the optimization process, should be assigned. Here, the change of CF is considered to be according to the following curve:

$$PP\,(Loop_i) = PP_1 + (1 - PP_1)\frac{Loop_i^{Power} - 1}{(Loops\,Number)^{Power} - 1} \tag{6.1}$$

PP: Predefined probability.

PP_1: Convergence factor of the first loop in which the answers are selected randomly.

$Loop_i$: Number of the current loop.

$Power$: Degree of the curve. As it can be seen, the curve in (6.1) is a polynominal of $Power$ degree.

$Loops\ Number$: Number of loops in which the algorithm should reach to the convergence point. This number should be chosen by the user according to the computational effort that can be afforded for the algorithm.

Figure 6.2 shows the variation of PP by the changes of the $Power$, using the proposed formula, Eq. (6.1).

The flowchart of the algorithm is shown in Fig. 6.3. The main steps of Dolphin Echolocation (DE) for discrete optimization are as follows:

1. Initiate NL locations for a dolphin randomly.
   This step contains creating L_{NL*NV} matrix, in which NL is the number of locations and NV is the number of variables (or dimension of each location).
2. Calculate the PP of the loop using (6.1).
3. Calculate the fitness of each location.
   Fitness should be defined in a manner that the better answers get higher values. In other words the optimization goal should be to maximize the fitness.

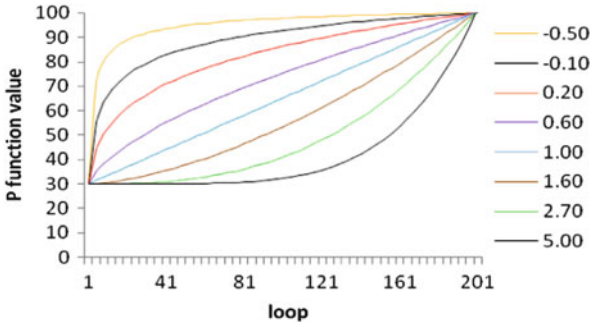

Fig. 6.2 Sample convergence curves, using (6.1) for different values for power [6]

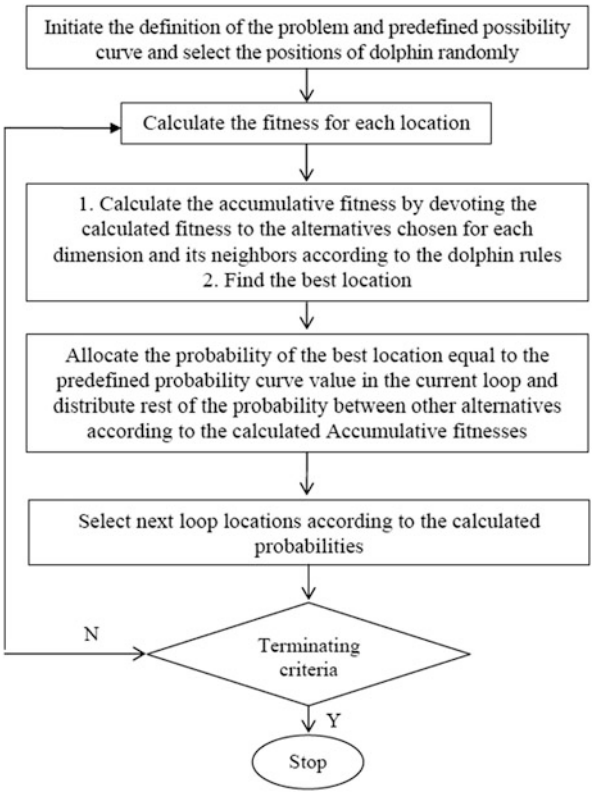

Fig. 6.3 The flowchart of the DE algorithm [1]

4. Calculate the accumulative fitness according to dolphin rules as follows:

   (a) for $i = 1$ to the number of locations
   for $j = 1$ to the number of variables
   find the position of $L(i,j)$ in j^{th} column of the Alternatives matrix and name it as A.
   for $k = -R_e$ to R_e

$$AF_{(A+k)_j} = \frac{1}{R_e} * \left(R_e - |k|\right) Fitness\,(i) + AF_{(A+k)j} \tag{6.2}$$

end
end
end
Where

$AF_{(A+k)j}$ is the accumulative fitness of the $(A + k)$th alternative (numbering of the alternatives is identical to the ordering of the Alternative matrix) to be chosen for the jth variable; R_e: is the effective radius in which accumulative fitness of the alternative A's neighbors are affected from its fitness. This radius is recommended to be not more than 1/4 of the search space; *Fittness* (i) is the fitness of location i.

It should be added that for alternatives close to edges (where $A + k$ is not a valid; $A + k < 0$ or $A + \mathrm{k} > \mathrm{L}A_j$), the AF is calculated using a reflective characteristic. In this case, if the distance of an alternative to the edge is less than R_e, it is assumed that the same alternative exists where picture of the mentioned alternative can be seen, if a mirror is placed on the edge.

(b) In order to distribute the possibility much evenly in the search space, a small value of is added to all the arrays as $AF = AF + \varepsilon$. Here, ε should be chosen according to the way the fitness is defined. It is better to be less than the minimum value achieved for the fitness.

(c) Find the best location of this loop and name it "The best location". Find the alternatives allocated to the variables of the best location, and let their AF be equal to zero.

In other words:
for $j = 1$: Number of variables
for $i = 1$: Number of alternatives
if $i =$ The best location(j)

$$AF_{ij} = 0 \tag{6.3}$$

end
end
end

5. for variable j(j = 1,…, NV), calculate the probability of choosing alternative i(i = 1,…, ALj), according to the following relationship:

$$P_{ij} = \frac{AF_{ij}}{\sum_{i=1}^{\mathrm{LAj}} AF_{ij}} \tag{6.4}$$

6. Assign a probability equal to PP to all alternatives chosen for all variables of the

best location and devote rest of the probability to the other alternatives according to the following formula:

for $j = 1$: Number of variables

for $i = 1$: Number of alternatives

if $i =$ The best location(j)

$$P_{ij} = PP \tag{6.5}$$

Else

$$P_{ij} = (1 - PP)P_{ij} \tag{6.6}$$

end

end

end

Calculate the next step locations according to the probabilities assigned to each alternative.

Repeat Steps 2 to 6 as many times as the *Loops Number*.

Parameters of the Algorithm Input parameters for the algorithm are as follows:

(a) Loops Number
For an optimization algorithm it is beneficial for the user to be able to dictate the algorithm to work according to the affordable computational cost. The answers may obviously be dependent on the selected number of loops and will improve by an increase in the loops number. However, the point is that one may not achieve results as bad as those of other optimization algorithms gained in less loops, because in this case although the algorithm quit its job much sooner than expected, the answer is good because of convergence criteria being reached. The number of loops can be selected by sensitivity analysis when high accuracy is required, however, in structural optimization of normal buildings, the loops number is recommended to be more than 50.

(b) Convergence Curve Formula
This is another important parameter to be selected for the algorithm. The curve should reach to the final point of 100 % smoothly. If the curve satisfies the above mentioned criteria the algorithm will perform the job properly, but it is recommended to start with a linear curve and try the curves that spend more time (more loops) in high values of the PP. For example, if one is using proposed curves of this chapter, it is recommended to start with $Power = 1$ which usually gives good results and it is better to try some cases of the $Power < 1$ to check if it improves the results.

(c) Effective Radius (R_e)
This parameter is better to be chosen according to the size of search space. It is recommended to be selected less than ¼ of the size of the search space.

(d) ε
This parameter is better to be less than any possible fitness.
(e) Number of Locations (*NL*)
This parameter is the same as the population size in GA or number of ants in ACO. It should be chosen in a reasonable way.

An Illustrative Numerical Example As an example consider the following simple mathematical function optimization problem:

$$\min\left(h = \sum_{i=1}^{N} x_i^2\right), x_i \in Z, \ -20 \leq x_i \leq 20 \tag{6.7}$$

Considering $N = 4$, dolphin echolocation algorithm suggests the following steps:

Before starting the optimization process for the changes of CF, a curve should be selected using (6.1), utilizing Power = 1, Loops number = 8, and $PP_1 = 0.1$, as follow:

$$PP = 0.1 + 0.9\left(\frac{Loop_i - 1}{7}\right) = 0.1 + 0.9(Loop_i - 1) \tag{6.8}$$

It should be noted that the PP_1 is better to be considered as the *CF* of the randomly selected generation of the first loop, which is equal to 0.11 for this example.

Dolphin Echolocation steps to solve the problem are as follows:

1. Create the initial locations randomly, which includes the generating *NL* vectors consisting of *N* integer numbers between −20 and 20. For example, considering *NL* and *N* equal to 30 and 4, 30 vectors of length 4 should be selected randomly. One possible answer for the *i*th location can be $L_i = \{-10, 4, -7, 18\}$.
2. Calculate the *PP* of the loop using (6.8).
3. Calculate fitness for each location. In this example as the objective function is defined by (6.7), for the considered location (L_i), $h = (-10)^2 + 4^2 + (-7)^2 + 18^2 = 489$. As in DE, the fitness is used to calculate the probability. Better fitnesses should have higher possibilities, then we can use $Fitness = 1/h$. It should be added that, for this special case, as h can be equal to zero, small value of 1 is added to the h in order to prevent the error of dividing by zero. Then the $Fitness = 1/(h + 1)$, and for the considered location $Fitness(L_i) = 1/(489 + 1) = 0.00204$.
4. Calculate the Accumulative fitness, using (6.2). As discussed before the alternatives should be sorted in an ascending order. The $Alternatives_{MA*NV}$ (*MA* is the number of alternatives, and *NV* is the number of optimization variables) is allocated to the possible alternatives for variables. For this example, the Alternatives matrix is:

$$Alternatives = \begin{bmatrix} -20 & -20 & -20 & -20 \\ -19 & -19 & -19 & -19 \\ . & . & . & . \\ . & . & . & . \\ . & . & . & . \\ 19 & 19 & 19 & 19 \\ 20 & 20 & 20 & 20 \end{bmatrix} \tag{6.9}$$

Then for sample location, L_i, considering $R_e = 10$, Eq. (6.2) becomes:
for $i = L_i$
for $j = 1$ to 4
find the position of $L(i,j)$ in the jth column of the Alternatives matrix and name it as A.
for $k = -10$ to 10

$$AF_{(A+k)j} = \frac{1}{10} * \left(10 - |k|\right) Fitness\left(L_i\right) + AF_{(A+k)j} \tag{6.10}$$

end
end
end
Equation (6.10) can also be stated as:
for $j = \{1, 2, 3, 4\}$

$$L(i,j) = \{-10, 4, -7, 18\}, \text{ then } A = \{11, 25, 14, 39\}$$

for $k = -10$ to 10

$$\begin{aligned} AF_{(11+k)1} &= \frac{1}{10} * \left(10 - |k|\right) Fitness\left(L_i\right) + AF_{(11+k)1} \\ AF_{(25+k)2} &= \frac{1}{10} * \left(10 - |k|\right) Fitness\left(L_i\right) + AF_{(25+k)2} \\ AF_{(14+k)3} &= \frac{1}{10} * \left(10 - |k|\right) Fitness\left(L_i\right) + AF_{(14+k)3} \\ AF_{(39+k)4} &= \frac{1}{10} * \left(10 - |k|\right) Fitness\left(L_i\right) + AF_{(39+k)4} \end{aligned} \tag{6.11}$$

end
end
Considering ε as the worth possible fitness, it will be $\varepsilon = 1/(4 * 20^2)$ and then $AF = AF + 0.000625$.
In these equations, it can be seen that for example for $j = 2$ (the second variable), for calculating the accumulative fitness, search space should be

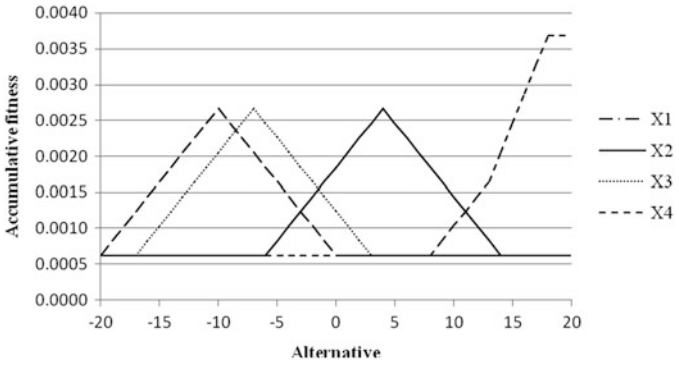

Fig. 6.4 Accumulative fitness resulted from sample location of the mathematical example [1]

divided into two regions: affected region (in effective radius) and not affected region. Choosing R_e equal to 10, alternatives with absolute distance to 4 (alternative 4 is chosen for the second variable) more than 10 ($x < -6$ and $x > 14$) are considered not affected. Also in the affected area the accumulative fitness resulted from this sample location changes linearly in a way that its maximum appears in $x = 4$. The accumulative fitness to be added for this alternative is:

$$AF_{(x+25)2} = AF_{(x+25)2} + \begin{cases} 0 & x < -6 \\ \dfrac{Fitness\,(Li)}{10}(x+6) & -6 < x \leq 4 \\ \dfrac{Fitness\,(Li)}{10}(14-x) & 4 < x \leq 14 \\ 0 & x > 14 \end{cases} \tag{6.12}$$

$$AF = AF + 0.000625$$

Figure 6.4 shows the result of performing the explained process for all 4 variables of this location.

Performing Step 4 for all the randomly selected answers, the final Accumulative fitness of the first loop is achieved.

5. For variable $j(j = 1,\ldots, 4)$, calculate the probability of choosing alternative $i(i = 1,\ldots, 40)$, according to the following relationship:

$$P_{ij} = \frac{AF_{ij}}{\sum_{i=1}^{40} AF_{ij}} \tag{6.13}$$

and consequently the probability will be according to Figs. 6.5 and 6.6.

6. Figure 6.5 demonstrates the accumulative fitness of variables X1, X2, X3 and X4. The best location of the first loop is achieved by setting variables as: X1 = − 11, X2 = 3, X3 = X4 = 4. On the other hand, according to (6.8),

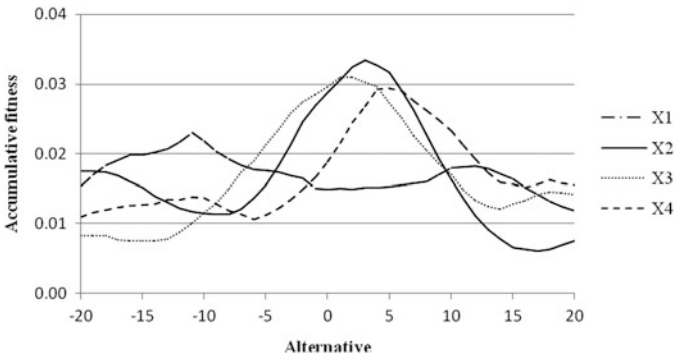

Fig. 6.5 Accumulative fitness of all four variables in the first loop of DE in mathematical example [1]

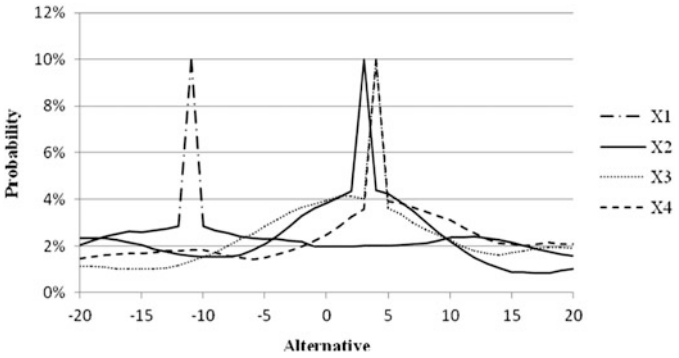

Fig. 6.6 Probability curve of all four variables in the first loop of DE in mathematical example [1]

PP for the first loop is equal to 10 %, as a result all variables in their best placement is equal to 10 % probability of the other alternatives is defined distributing remaining value of probability equal to 90 % to the other alternatives, using the following formula:

$$P_{ij} = (1 - 0.1)P_{ij} = 0.9P_{ij} \tag{6.14}$$

Since the number of loops is equal to 8, Steps 2 to 6 should be repeated 8 times.

Figures 6.7, 6.8, 6.9, and 6.10 show the accumulative fitness and the probability of alternatives in loops 4 and 8, respectively. It can be seen from these figures that the probability changes in a way that in 8 loops DE reaches the best answer.

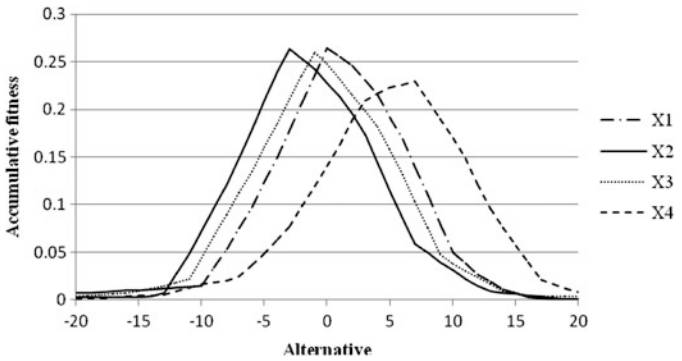

Fig. 6.7 Accumulative fitness of all four variables in the fourth loop of DE in of mathematical example [1]

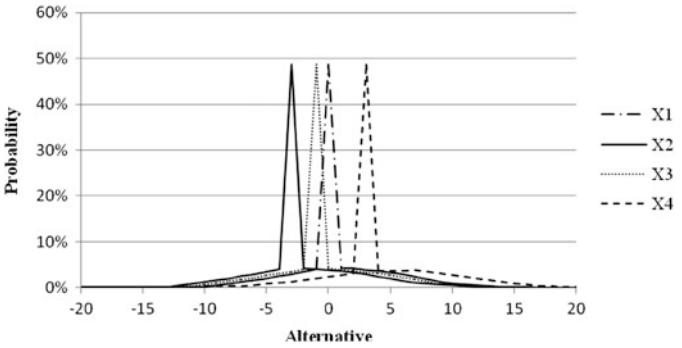

Fig. 6.8 Probability curve of all four variables in the fourth loop of DE in mathematical example

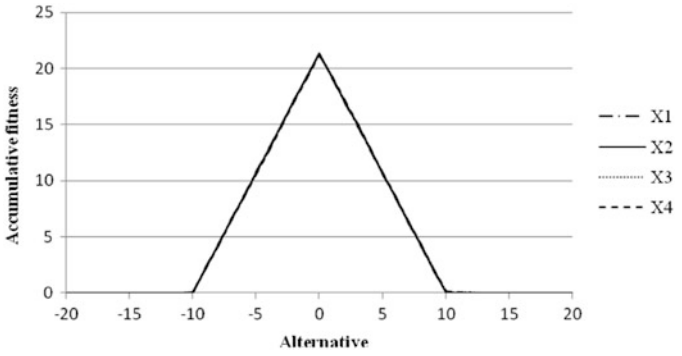

Fig. 6.9 Accumulative fitness of all four variables in the eighth loop of DE in of mathematical example [1]

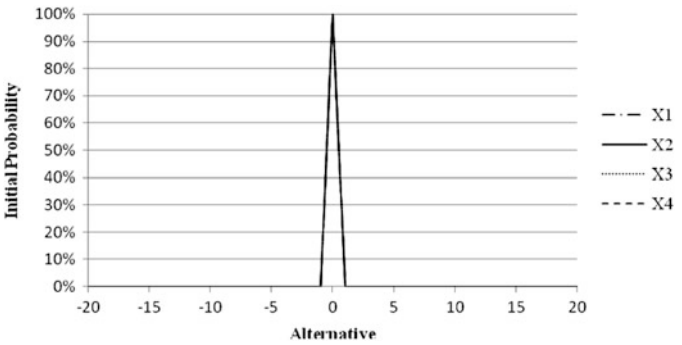

Fig. 6.10 Probability curve of all four variables in the eighth loop of DE in mathematical example [1]

Comparison Between the Dolphin Echolocation and Bat Inspired Algorithm Bat inspired algorithm can be considered as a balanced combination of the standard particle swarm optimization and the intensive local search controlled by the loudness and pulse rate [7]. In this algorithm loudness and pulse frequency are echolocation parameters that gradually restrict the search according to pulse emission and loudness rules. This is while, in dolphin echolocation algorithm there is no movement to the best answer. DE algorithm works with possibilities.

6.4 Structural Optimization

In this study the structural optimization goal is to minimize the weight of the structure that is formulated as follows:

Minimize:

$$W = \rho \sum_{i=1}^{M} A_i L_i \tag{6.15}$$

Subjected to:

$$\begin{aligned} &KU - P = 0 \\ &g_1 \geq 0, g_2 \geq 0, \ldots, g_n \geq 0 \end{aligned} \tag{6.16}$$

Where $g_1, g_2, \ldots, g_n$ are constraint functions depending on the element being used in each problem and K, U and P are the stiffness matrix, nodal displacement and nodal force vectors, respectively. In this study, different constraints are implemented for structural design including drift, displacement and strength. Constraints are clarified in numerical examples.

Furthermore, such a constrained formulation is treated in an unconstrained form, using a penalized fitness function as:

$$F = F_0 - W * \left(1 + K_p.V\right) \tag{6.17}$$

Where F_0 is a constant taken as zero for the class of considered examples. K_p is the penalty coefficient, and V denotes the total constraints' violation considering all the load combinations.

6.5 Numerical Examples

In this section three trusses and two frames are optimized using the present algorithm and the results are compared to those of some other existing approaches. The algorithms are coded in Matlab and structures are analyzed using the direct stiffness method.

6.5.1 Truss Structures

In the following three trusses are optimized and the results of the present algorithm are compared to those of different algorithm.

6.5.1.1 A 25-Bar Spatial Truss

The 25-bar spatial truss structure shown in Fig. 6.11 has been studied in [8–11]. The material density is 0.1 lb/in^3 (2,767.990 kg/m^3) and the modulus of elasticity is 10,000 ksi (68,950 MPa). The stress limitations of the members are ±40 kpsi (±275.80 MPa). All nodes in three directions are subjected to displacement limitations of ±0.35 inch (in) (±8.89 mm) imposed on every node in each direction. The structure includes 25 members, which are divided into eight groups, as follows: (1) A_1, (2) A_2–A_5, (3) A_6–A_9, (4) A_{10}–A_{11}, (5) A_{12}–A_{13}, (6) A_{14}–A_{17}, (7) A_{18}–A_{21} and (8) A_{22}–A_{25}. Two optimization cases are implemented.

Case 1: The discrete variables are selected from the set $D = \{0.01, 0.4, 0.8, 1.2, 1.6, 2.0, 2.4, 2.8, 3.2, 3.6, 4.0, 4.4, 4.8, 5.2, 5.6, 6.0\}$ (in^2) or {0.065, 2.58, 5.16, 7.74, 10.32, 12.90, 15.48, 18.06, 20.65, 23.22, 25.81, 28.39, 30.97, 33.55, 36.13, 38.71} (cm^2).
Case 2: The discrete variables are selected from the [12], listed in Table 6.2. The loads for both cases are shown in Table 6.3.

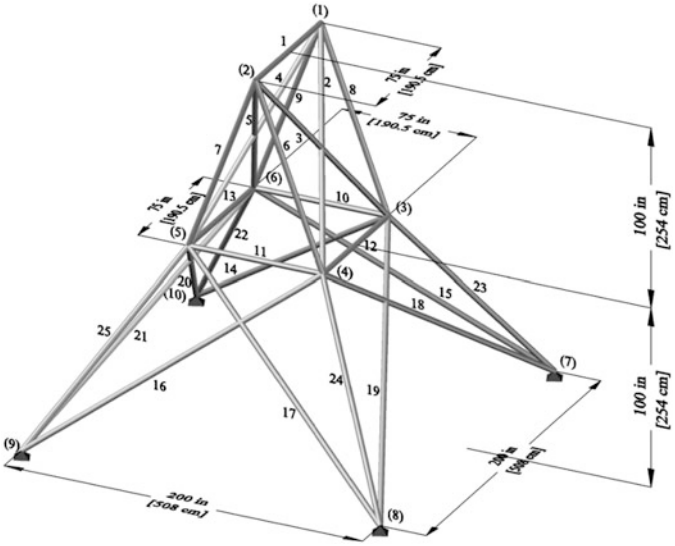

Fig. 6.11 Schematic of a 25-bar spatial truss

For solving this problem by the use of DE, Loops number is set to 80. Convergence curve is according to (6.1) considering $PP_1 = 0.15$ and $Power = 1$. R_e and ε are equal to 5 and 1, respectively.

According to Tables 6.4 and 6.5 and Fig. 6.12, DE achieves the best answer in approximately 50 loops in Case 1 and near 80 loops in Case 2, while HPSACO reaches to the same result in around 100 loops. It should be mentioned that Kaveh and Talatahari [11] show that the HPSACO itself has better convergence rate in comparison to GA, PSO, PSOPC and HPSO.

In addition, Fig. 6.13 shows the convergence factor history. It can be seen that the algorithm follows the predefined linear curve as expected.

6.5.1.2 A 72-Bar Spatial Truss

For the 72-bar spatial truss structure shown in Fig. 6.14, the material density is 0.1 lb/in^3 (2,767:990 kg/m^3) and the modulus of elasticity is 10,000 ksi (68,950 MPa). The members are subjected to the stress limits of ±25 ksi (±172.375 MPa). The nodes are subjected to the displacement limits of ±0.25 in (±0.635 cm).

The 72 structural members of this spatial truss are sorted into 16 groups using symmetry: (1) A_1–A_4, (2) A_5–A_{12}, (3) A_{13}–A_{16}, (4) A_{17}–A_{18}, (5) A_{19}–A_{22}, (6) A_{23}–A_{30}, (7) A_{31}–A_{34}, (8) A_{35}–A_{36}, (9) A_{37}–A_{40}, (10) A_{41}–A_{48}, (11) A_{49}–A_{52}, (12) A_{53}–A_{54}, (13) A_{55}–A_{58}, (14) A_{59}–A_{66} (15), A_{67}–A_{70}, and (16) A_{71}–A_{72}.

Two optimization cases are implemented.

Table 6.2 The available cross-section areas of the AISC code

No.	in^2	mm^2	No.	in^2	mm^2
1	0.111	(71.613)	33	3.840	(2,477.414)
2	0.141	(90.968)	34	3.870	(2,496.769)
3	0.196	(126.451)	35	3.880	(2,503.221)
4	0.250	(161.290)	36	4.180	(2,696.769)
5	0.307	(198.064)	37	4.220	(2,722.575)
6	0.391	(252.258)	38	4.490	(2,896.768)
7	0.442	(285.161)	39	4.590	(2,961.284)
8	0.563	(363.225)	40	4.800	(3,096.768)
9	0.602	(388.386)	41	4.970	(3,206.445)
10	0.766	(494.193)	42	5.120	(3,303.219)
11	0.785	(506.451)	43	5.740	(3,703.218)
12	0.994	(641.289)	44	7.220	(4,658.055)
13	1.000	(645.160)	45	7.970	(5,141.925)
14	1.228	(792.256)	46	8.530	(5,503.215)
15	1.266	(816.773)	47	9.300	(5,999.988)
16	1.457	(939.998)	48	10.850	(6,999.986)
17	1.563	(1,008.385)	49	11.500	(7,419.430)
18	1.620	(1,045.159)	50	13.500	(8,709.660)
19	1.800	(1,161.288)	51	13.900	(8,967.724)
20	1.990	(1,283.868)	52	14.200	(9,161.272)
21	2.130	(1,374.191)	53	15.500	(9,999.980)
22	2.380	(1,535.481)	54	16.000	(10,322.560)
23	2.620	(1,690.319)	55	16.900	(10,903.204)
24	2.630	(1,696.771)	56	18.800	(12,129.008)
25	2.880	(1,858.061)	57	19.900	(12,838.684)
26	2.930	(1,890.319)	58	22.000	(14,193.520)
27	3.090	(1,993.544)	59	22.900	(14,774.164)
28	1.130	(729.031)	60	24.500	(15,806.420)
29	3.380	(2,180.641)	61	26.500	(17,096.740)
30	3.470	(2,238.705)	62	28.000	(18,064.480)
31	3.550	(2,290.318)	63	30.000	(19,354.800)
32	3.630	(2,341.931)	64	33.500	(21,612.860)

Table 6.3 Loading conditions for the 25-bar spatial truss

	Case 1			Case 2		
Node	P_X kips (kN)	P_Y kips (kN)	P_Z kips (kN)	P_X kips (kN)	P_Y kips (kN)	P_Z kips (kN)
1	0.0	20.0 (89)	−5.0 (22.25)	1.0 (4.45)	10.0 (44.5)	−5.0 (22.25)
2	0.0	−20.0 (89)	−5.0 (22.25)	0.0	10.0 (44.5)	−5.0 (22.25)
3	0.0	0.0	0.0	0.5 (2.22)	0.0	0.0
6	0.0	0.0	0.0	0.5 (2.22)	0.0	0.0

Case 1: The discrete variables are selected from the set $D = \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2.0, 2.1, 2.2, 2.3, 2.4, 2.5, 2.6, 2.7, 2.8, 2.9, 3.0, 3.1, 3.2\}$ (in^2) or {0.65, 1.29, 1.94, 2.58, 3.23, 3.87, 4.52, 5.16, 5.81, 6.45, 7.10, 7.74, 8.39, 9.03, 9.68, 10.32, 10.97, 12.26, 12.90, 13.55, 14.19, 14.84, 15.48, 16.13, 16.77, 17.42, 18.06, 18.71, 19.36, 20.00, 20.65} (cm^2).

Table 6.4 Optimal design comparison for the 25-bar spatial truss (Case 1)

Element group		Optimal cross-sectional areas (in^2)								
		Wu and Chow [8]	Lee and Geem [9]	Li et al. [10]			Kaveh and Talatahari [11] HPSACO		Present work [1]	
		GA	HS	PSO	PSOPC	HPSO	in^2	cm^2	in^2	cm^2
1	A_1	0.40	0.01	0.01	0.01	0.01	0.01	0.07	0.01	0.07
2	A_2–A_5	2.00	2.00	2.00	2.00	2.00	1.60	10.32	1.60	10.32
3	A_6–A_9	3.60	3.60	3.60	3.60	3.60	3.20	20.65	3.20	20.65
4	A_{10}–A_{11}	0.01	0.01	0.01	0.01	0.01	0.01	0.07	0.01	0.07
5	A_{12}–A_{13}	0.01	0.01	0.40	0.01	0.01	0.01	0.07	0.01	0.07
6	A_{14}–A_{17}	0.80	0.80	0.80	0.80	0.80	0.80	5.16	0.80	5.16
7	A_{18}–A_{21}	2.00	1.60	1.60	1.60	1.60	2.00	12.90	2.00	12.90
8	A_{22}–A_{25}	2.40	2.40	2.40	2.40	2.40	2.40	15.48	2.40	15.48
Weight (lb)		563.52	560.59	566.44	560.59	560.59	551.6	250.2 kg	551.6	250.2 kg

Table 6.5 Optimal design comparison for the 25-bar spatial truss (Case 2)

Element group		Optimal cross-sectional areas (in^2)							
		Wu and Chow [8]	Li et al. [10]			Kaveh and Talatahar [11] HPSACO		Present work [1]	
		GA	PSO	PSOPC	HPSO	in^2	cm^2	in^3	cm^3
1	A_1	0.31	1.00	0.11	0.11	0.11	0.72	0.11	0.72
2	A_2–A_5	1.99	2.62	1.56	2.13	2.13	13.74	2.13	13.74
3	A_6–A_9	3.13	2.62	3.38	2.88	2.88	18.58	2.88	18.58
4	A_{10}–A_{11}	0.11	0.25	0.11	0.11	0.11	0.72	0.11	0.72
5	A_{12}–A_{13}	0.14	0.31	0.11	0.11	0.11	0.72	0.11	0.72
6	A_{14}–A_{17}	0.77	0.60	0.77	0.77	0.77	4.94	0.77	4.94
7	A_{18}–A_{21}	1.62	1.46	1.99	1.62	1.62	10.45	1.62	10.45
8	A_{22}–A_{25}	2.62	2.88	2.38	2.62	2.62	16.90	2.62	16.90
Weight (lb)		556.43	567.49	567.49	551.14	551.1	249.99	551.1	249.99

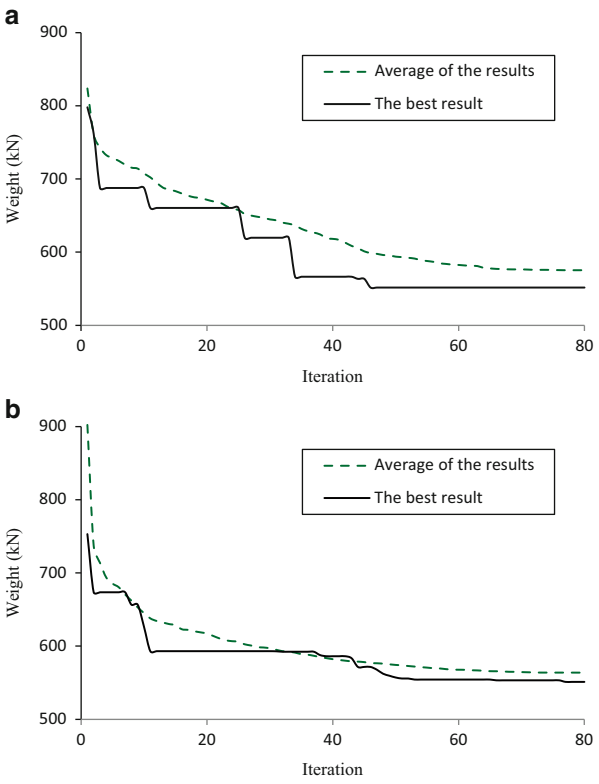

Fig. 6.12 The optimum answer convergence history for the 25-bar truss using DE [1]. (**a**) Case 1 and (**b**) Case 2

Case 2: The discrete variables are selected from Table 6.2.

Table 6.6 lists the values and directions of the two load cases applied to the 72-bar spatial truss.

The problem has been solved by GA [8, 9] and DHPSO [11].

Solving the problem using DE, the Loops number is set to 200. Convergence curve is according to (6.1) considering $PP_1 = 0.15$ and $Power = 1$. R_e and ε are equal to 5 and 1, respectively.

It can be seen from Table 6.7 that in Case 1 the best answer is achieved using DE that is better than GA and HS and although it is the same as DHPACO, but the penalty of the optimum answer is less than that of the DHPACO.

Moreover Table 6.8 shows that in Case 2, the DE achieves better results in comparison to the previously published works. Figure 6.15 shows that the DE can converge to the best answer in 200 loops, then it has higher convergence rate compared to the other algorithms.

In addition, Fig. 6.16 shows the convergence factor history. It can be seen that the algorithm follows the predefined linear curve as expected.

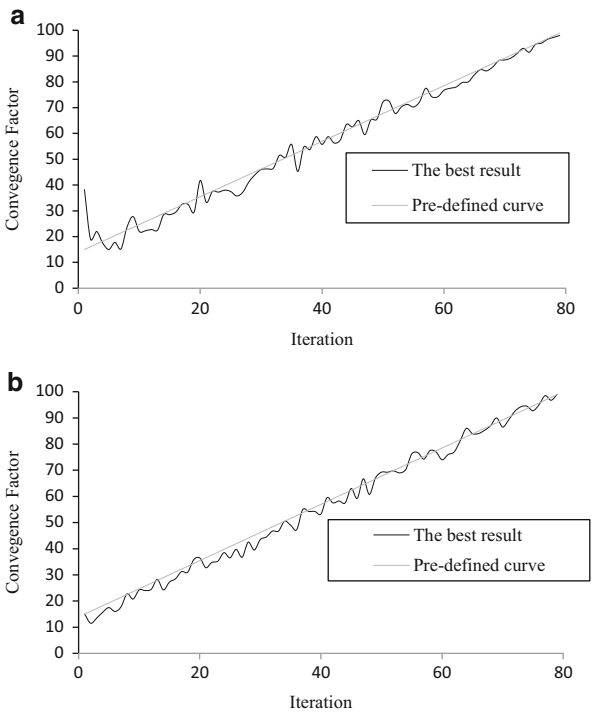

Fig. 6.13 The optimum answer and the average answers' convergence factor history for the 25-bar truss structure using the DE [1]. (**a**) Case 1 and (**b**) Case 2

Figure 6.17 shows the allowable and existing displacements for the nodes of the 72-bar truss structure using the DE.

6.5.1.3 A 582-Bar Tower Truss

The 582-bar tower truss shown in Fig. 6.18, is chosen from [13]. The symmetry of the tower about x-axis and y-axis is considered to group the 582 members into 32 independent size variables.

A single load case is considered consisting of the lateral loads of 5.0 kN (1.12 kips) applied in both x- and y-directions and a vertical load of 30 kN (6.74 kips) applied in the z-direction at all nodes of the tower. A discrete set of 140 economical standard steel sections selected from W-shape profile list based on area and radii of gyration properties is used to size the variables [13]. The lower and upper bounds on size variables are taken as 6.16 in^2 (39.74 cm^2) and 215.0 in^2 (1,387.09 cm^2), respectively. The stress limitations of the members are imposed according to the provisions of ASD-AISC [12] as follows:

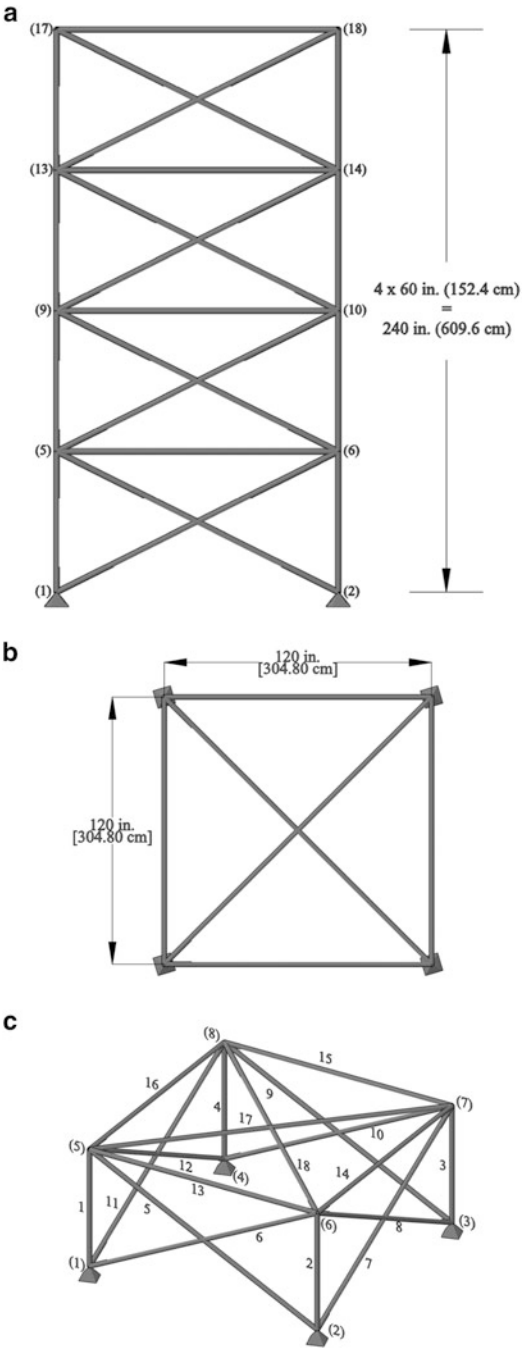

Fig. 6.14 Schematic of a 72-bar spatial truss [1]. (**a**) Front view, (**b**) top view, and (**c**) Element and node numbering system of a typical story

Table 6.6 Loading conditions for the 72-bar spatial truss

	Case 1			Case 2		
Node	P_x kips (kN)	P_Y kips (kN)	P_Z kips (kN)	P_X kips (kN)	P_Y kips (kN)	P_Z kips (kN)
17	5.0 (22.25)	5.0 (22.25)	−5.0 (22.25)	0	0	−5.0 (22.25)
18	0.0	0.0	0.0	0.0	0.0	−5.0 (22.25)
19	0.0	0.0	0.0	0.0	0.0	−5.0 (22.25)
20	0.0	0.0	0.0	0.0	0.0	−5.0 (22.25)

Table 6.7 Optimal design comparison for the 72-bar spatial truss (Case 1)

		Optimal cross-sectional areas (in^2)					
		Wu and Chow [8]	Lee and Geem [9]	Kaveh and Talatahari [11]		Present work [1]	
Element group		GA	HS	DHPSACO		DE	
		in^2	in^2	in^2	cm^2	in^2	cm^2
1	A_1–A_4	1.5	1.9	1.9	12.26	2.0	12.90
2	A_5–A_{12}	0.7	0.5	0.5	3.23	0.5	3.23
3	A_{13}–A_{16}	0.1	0.1	0.1	0.65	0.1	0.65
4	A_{17}–A_{18}	0.1	0.1	0.1	0.65	0.1	0.65
5	A_{19}–A_{22}	1.3	1.4	1.3	8.39	1.3	8.39
6	A_{23}–A_{30}	0.5	0.6	0.5	3.23	0.5	3.23
7	A_{31}–A_{34}	0.2	0.1	0.1	0.65	0.1	0.65
8	A_{35}–A_{36}	0.1	0.1	0.1	0.65	0.1	0.65
9	A_{37}–A_{40}	0.5	0.6	0.6	3.87	0.5	3.23
10	A_{41}–A_{48}	0.5	0.5	0.5	3.23	0.5	3.23
11	A_{49}–A_{52}	0.1	0.1	0.1	0.65	0.1	0.65
12	A_{53}–A_{54}	0.2	0.1	0.1	0.65	0.1	0.65
13	A_{55}–A_{58}	0.2	0.2	0.2	1.29	0.2	1.29
14	A_{59}–A_{66}	0.5	0.5	0.6	3.87	0.6	3.87
15	A_{67}–A_{70}	0.5	0.4	0.4	2.58	0.4	2.58
16	A_{71}–A_{72}	0.7	0.6	0.6	3.87	0.6	3.87
Weight (lb)		400.66	387.94	385.54	174.9 kg	385.54	174.9 kg

$$\begin{cases} \sigma_i^+ = 0.6Fy & for \quad \sigma_i \geq 0 \\ \sigma_i^- & for \quad \sigma_i < 0 \end{cases} \tag{6.18}$$

Where σ_i^- is calculated according to the slenderness ratio

$$\sigma_i^- = \begin{cases} \left[\left(1 - \dfrac{\lambda_i^2}{2C_i^2}\right)F_y\right] \Big/ \left(\dfrac{5}{3} + \dfrac{3\lambda_i}{8C_C} - \dfrac{\lambda_i^3}{8C_C^3}\right) & for \quad \lambda_i < C_C \\ \dfrac{12\pi^2 E}{23\lambda_i^2} \quad for \quad \lambda_i \geq C_C \end{cases} \tag{6.19}$$

Table 6.8 Optimal design comparison for the 72-bar spatial truss (Case 2)

		Optimal cross-sectional areas (in^2)				
		Wu and Chow [8]	Kaveh and Talatahari [11]		Present work [1]	
		GA	DHPSACO		DE	
Element group		in^2	in^2	cm^2	in^2	cm^2
1	A_1–A_4	0.196	1.800	11.610	2.130	13.742
2	A_5–A_{12}	0.602	0.442	2.850	0.442	2.852
3	A_{13}–A_{16}	0.307	0.141	0.910	0.111	0.716
4	A_{17}–A_{18}	0.766	0.111	0.720	0.111	0.716
5	A_{19}–A_{22}	0.391	1.228	7.920	1.457	9.400
6	A_{23}–A_{30}	0.391	0.563	3.630	0.563	3.632
7	A_{31}–A_{34}	0.141	0.111	0.720	0.111	0.716
8	A_{35}–A_{36}	0.111	0.111	0.720	0.111	0.716
9	A_{37}–A_{40}	1.800	0.563	3.630	0.442	2.852
10	A_{41}–A_{48}	0.602	0.563	3.630	0.563	3.632
11	A_{49}–A_{52}	0.141	0.111	0.720	0.111	0.716
12	A_{53}–A_{54}	0.307	0.250	1.610	0.111	0.716
13	A_{55}–A_{58}	1.563	0.196	1.270	0.196	1.265
14	A_{59}–A_{66}	0.766	0.563	3.630	0.563	3.632
15	A_{67}–A_{70}	0.141	0.442	2.850	0.307	1.981
16	A_{71}–A_{72}	0.111	0.563	3.630	0.563	3.632
Weight (lb)		427.203	393.380	178.4 kg	391.329	177.47 kg

Where E = the modulus of elasticity; F_y = the yield stress of A36 steel; $C_C = \sqrt{2\pi^2 E/F_y}$; λ_i = the slenderness ratio (kL_i/r_i); k = the effective length factor; L_i = the member length; and r_i = the radius of gyration. The other constraint is the limitation of the nodal displacements (no more than 8.0 cm or 3.15 in for each direction). In addition, the maximum slenderness ratio is limited to 300 for the tension members, and this limit is recommended to be 200 for the compression members according to the ASD-AISC [26] design code provisions.

The problem was solved later by Kaveh and Talatahari [14] and Sonmez [15]. Two cases for analyzing are used according to [15], as follows:

Case 1: All members are selected from a set of 140 W-shaped profiles according to [13] and the maximum number of evaluations is set to 50,000. For the DE, 25,000 evaluations are considered for this case to demonstrate the efficiency of the algorithm.

Case 2: There is no difference between Case 1 and Case 2, but in the number of evaluations which is set to 100,000. For the DE, 50,000 evaluations are considered for this case to demonstrate efficiency of the algorithm.

Convergence curve is according to (6.1) considering $PP_1 = 15\%$ and $Power = 0.2$. R_e and ε are equal to 10 and 1, respectively.

Results can be seen in Table 6.9, which shows that in Case 1, the DE outperforms the HPSACO, ABC and PSO by 5.7 % , 2.3 % and 1 %, respectively, and in

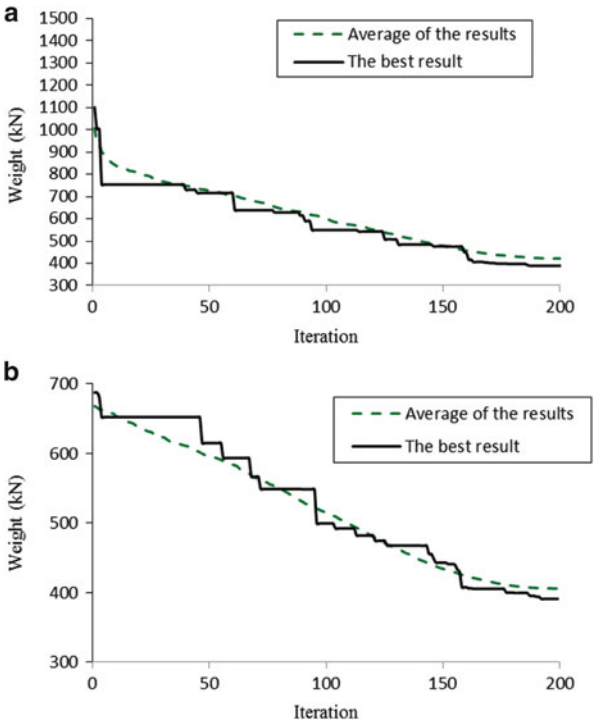

Fig. 6.15 The optimum answer and average answers' convergence history for the 72-bar truss using the DE [1]. (**a**) Case 1 and (**b**) Case 2

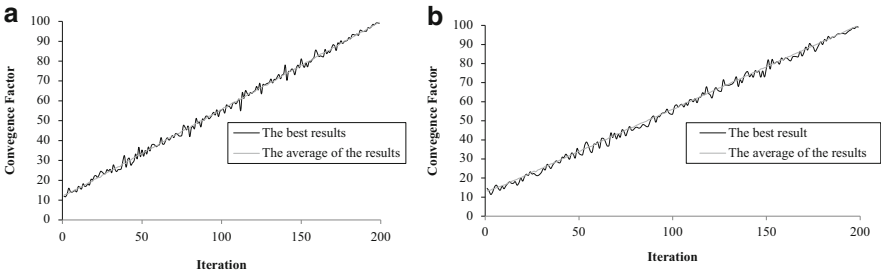

Fig. 6.16 The optimum answer and the average answers' convergence factor history for the 72-bar truss structure using the DE [1]. (**a**) Case 1 and (**b**) Case 2

Case 2, the DE results is 1.6 % better than those of ABC algorithm. In addition comparing the results with those presented in [13], it can be seen that the optimum answer of the DE in Case 1 is 1.1 %, 1.3 %, 2.2 %, 2.7 %, 4.7 % and 6.7 % lighter than those of the ESs, SA, TS, ACO, HS and SGA.

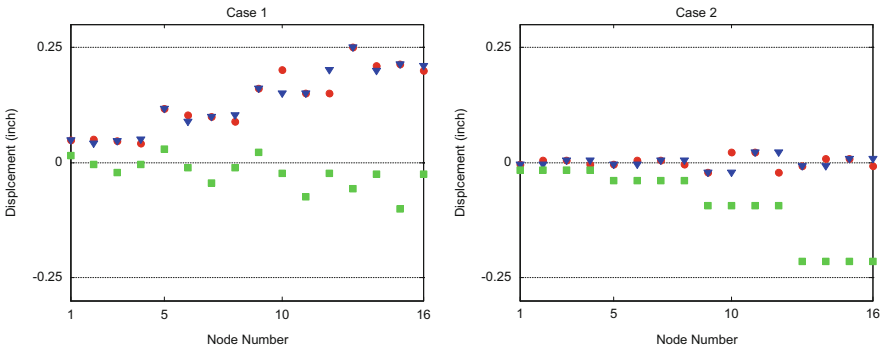

Fig. 6.17 Comparison of the allowable and existing displacements for the nodes of the 72-bar truss structure using the DE [1]

Figure 6.19 shows the comparison of the allowable and existing constrains for the 582-bar truss using the DE. The maximum values for displacement in x, y and z directions are 3.148 in (7.995 cm), 2.986 in (7.584 cm) and 0.931 in (2.365 cm), respectively. The maximum stress ratio is 96.60 %. It can be seen that some displacements and stresses are near the boundary conditions. It should be mentioned that there is a small difference between analysis results of Sap2000 (Hasançebi et al. [13]), C# programming language code (Sonmez [15]) and Matlab code (present study). Then checking the results of each code with another one may show a violation of constraints. Figure 6.19 shows according to the finite element program coded in Matlab, there is no penalty for the best answer.

Figure 6.20 shows the convergence history of the best answer and average results for the DE, and Fig. 6.21 illustrates the convergence factor history. It can be seen that the algorithm follows the predefined linear curve as expected.

6.5.2 *Frame Structures*

The displacement and AISC combined strength constraints are the performance constraints of the frame as follows:

(a) Maximum lateral displacement:

$$\frac{\Delta_T}{H} < R \tag{6.20}$$

Where Δ_T is the maximum lateral displacement of the structure (the roof lateral displacement), H is the height of the structure, and R is the maximum drift index.

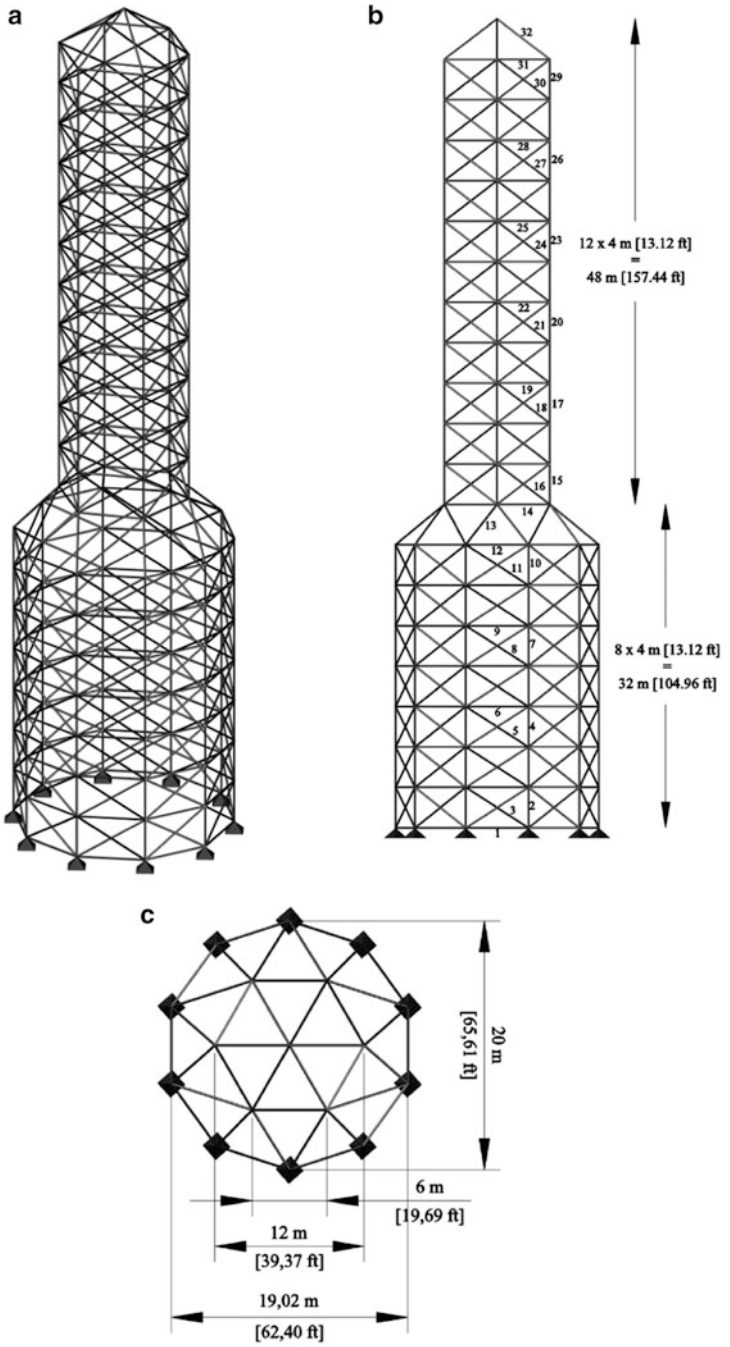

Fig. 6.18 Schematic of a 582-bar tower truss. (**a**) 3D view, (**b**) side view, and (**c**) top view

Table 6.9 Optimal design comparison for the 582-bar spatial truss

	Optimal cross-section					
	Case 1				Case 2	
	Hasançebi et al. [13]	Sonmez [15]	Kaveh and Talatahari [14]	Present work [1]	Sonmez [15]	Present work [1]
	(PSO)	(ABC)	(DHPSACO)	(DE)	(ABC)	(DE)
Element group	Ready section	Ready section	Ready section	Ready section	Ready section	Ready section
1	W8X21	W8X22	W8X24	W8X21	W8X22	W8X21
2	W12X79	W12X97	W12X72	W12X96	W10X78	W27X94
3	W8X24	W8X25	W8X28	W8X24	W8X25	W8X24
4	W10X60	W12X59	W12X58	W12X58	W14X62	W12X58
5	W8X24	W8X24	W8X24	W8X24	W8X24	W8X24
6	W8X21	W8X21	W8X24	W8X21	W8X21	W8X21
7	W14X48	W12X46	W10X49	W12X45	W12X51	W12X50
8	W8X24	W8X24	W8X24	W8X24	W8X24	W8X24
9	W8X21	W8X21	W8X24	W8X21	W8X21	W8X21
10	W10X45	W12X46	W12X40	W12X45	W10X50	W12X45
11	W8X24	W8X22	W12X30	W8X21	W8X25	W8X21
12	W10X68	W12X66	W12X72	W12X65	W10X69	W12X72
13	W14X74	W10X77	W18X76	W10X77	W18X77	W14X74
14	W14X48	W10X49	W10X49	W10X49	W14X49	W12X50
15	W18X76	W14X83	W14X82	W14X82	W10X78	W10X68
16	W8X31	W8X32	W8X31	W8X31	W8X32	W8X31
17	W16X67	W12X53	W14X61	W10X60	W21X62	W14X61
18	W8X24	W8X24	W8X24	W8X24	W8X24	W8X24
19	W8X21	W8X21	W8X21	W8X21	W8X21	W8X21
20	W8X40	W16X36	W12X40	W12X45	W14X43	W14X43
21	W8X24	W8X24	W8X24	W8X21	W8X24	W8X21
22	W8X21	W10X22	W14X22	W8X21	W8X21	W8X21
23	W10X22	W10X22	W8X31	W10X22	W8X24	W6X25
24	W8X24	W6X25	W8X28	W8X21	W8X24	W8X21
25	W8X21	W8X21	W8X21	W8X21	W8X21	W8X21
26	W8X21	W8X21	W8X21	W8X21	W8X21	W8X21
27	W8X24	W8X24	W8X24	W8X21	W8X24	W8X21
28	W8X21	W8X21	W8X28	W8X21	W8X21	W8X21
29	W8X24	W8X22	W16X36	W8X21	W8X21	W8X21
30	W8X21	W10X23	W8X24	W8X21	W8X21	W8X21
31	W8X21	W8X25	W8X21	W8X21	W8X24	W8X21
32	W8X24	W6X26	W8X24	W8X21	W8X24	W8X21
Best (lb)	363,795.7	368,484.1	380,982.7	360,367.8	365,906.3	360,143.3
Average (lb)	365,124.9	370,178.6	–	364,404.7	366,088.4	362,207.1
Worst (lb)	370,159.1	373,530.3	–	371,922.1	369,162.2	367,512.2
Evaluations (#)	50,000	50,000	8,500	25,000	100,000	50,000
Differences compared to DE	0.95 %	2.25 %	5.72 %		1.60 %	

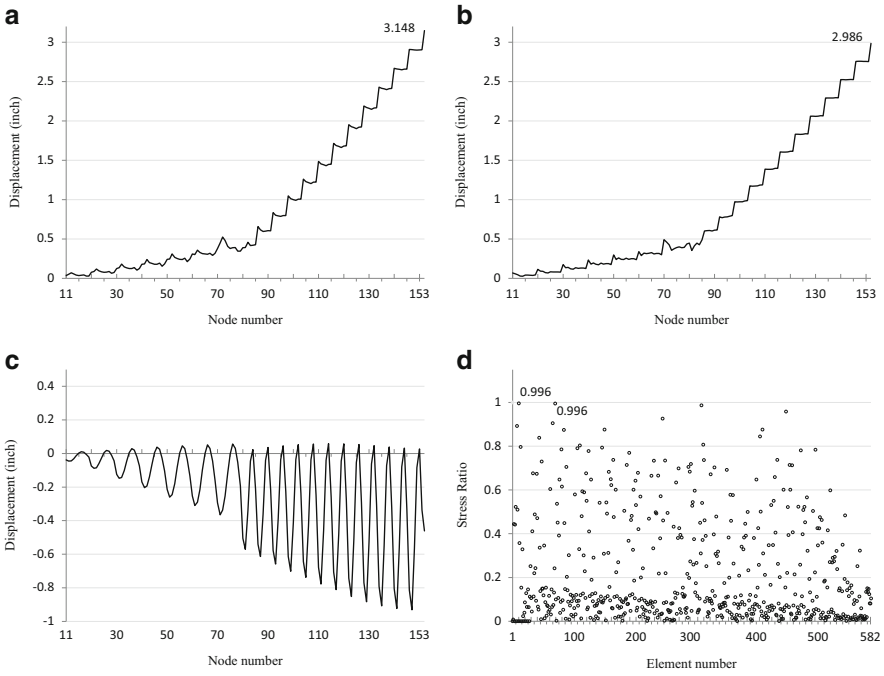

Fig. 6.19 Comparison of the allowable and existing constrains for the 582-bar truss, Case 2 using DE [1]. (**a**) Displacement in the x-direction. (**b**) Displacement in y-direction. (**c**) Displacement in the z-direction. (**d**) Stress ratios

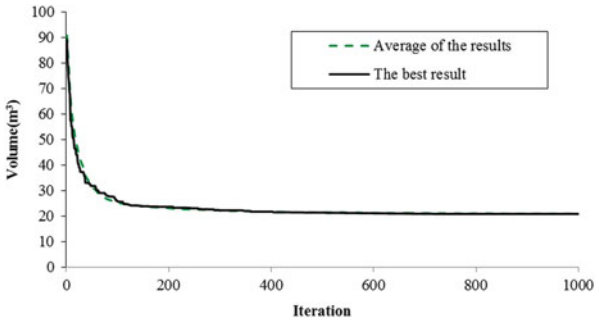

Fig. 6.20 Convergence history of optimum result and average results for the 582-bar tower truss, Case 2, using DE [1]

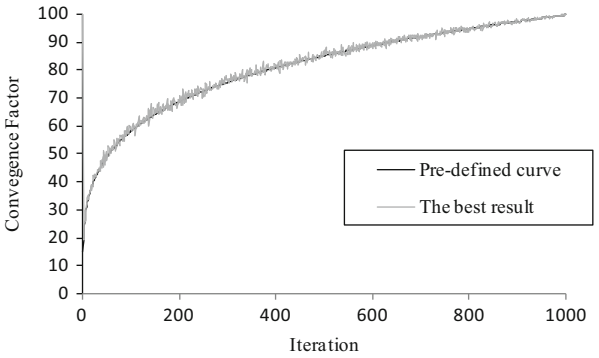

Fig. 6.21 Convergence factor history for the 582-bar tower truss [1], Case 2 using DE

(b) The inter-story displacements:

$$\frac{d_j}{h_j} < R_I, \quad j = 1, 2, \ldots, ns \tag{6.21}$$

d_j is the inter-story drift which is used to give the relative displacement of each roof in comparison to its following floor; h_j is the story height of j^{th} floor; ns is the total number of stories; R_I is the inter-story drift index which is equal to 1/300 according to the ANSI/AISC 360-05 (2005) [16].

(c) Element forces:

$$\begin{aligned} &\frac{P_u}{2\phi_C P_n} + \frac{M_u}{\phi_b M_n} < 1 \quad for \quad \frac{P_u}{\phi_C P_n} < 0.2 \\ &\frac{P_u}{\phi_C P_n} + \frac{8}{9}\frac{M_u}{\phi_b M_n} < 1 \quad for \quad \frac{P_u}{\phi_C P_n} \geq 0.2 \end{aligned} \tag{6.22}$$

Where P_u is the required strength (tension or compression); P_n is the nominal axial strength (tension or compression); ϕ_c is the axial resistance factor ($\phi_c = 0.9$ for tension, $\phi_c = 0.85$ for compression); M_u is required flexural strength; M_n is nominal flexural strength; and ϕ_b is the flexural resistance factor ($\phi_b = 0.9$).

6.5.2.1 A 3-Bay 15-Story Planar Frame

Figure 6.22 shows the configuration and applied loads of a 3-bay 15-story frame structure chosen from [14]. This frame consists of 64 joints and 105 members. The sway of the top story is limited to 23.5 cm. The material has a modulus of elasticity

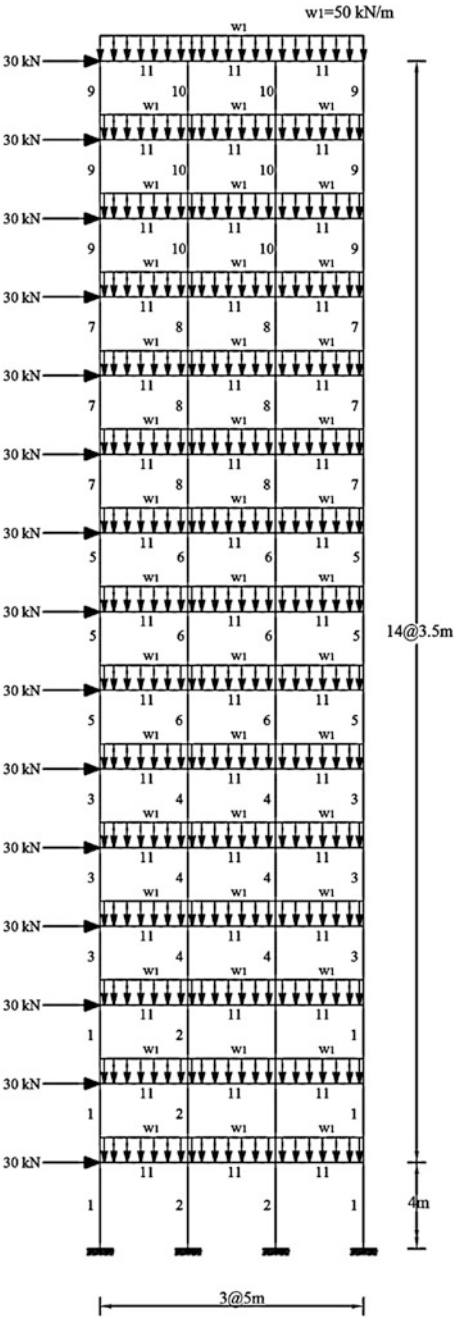

Fig. 6.22 Schematic of a 3-bay 15-story planar frame

Table 6.10 Optimal design comparison for the 3-bay 15-story planar frame

	Optimal W-shaped sections					
	Kaveh and Talatahari					
Element group	PSO [14]	PSOPC [14]	HPSACO [14]	ICA [17]	CSS [7]	Present work [1]
1	W33X118	W27X129	W21X111	W24X117	W21X147	W12X87
2	W33X263	W24X131	W18X158	W21X147	W18X143	W36X182
3	W24X76	W24X103	W10X88	W27X84	W12X87	W21X93
4	W36X256	W33X141	W30X116	W27X114	W30X108	W18X106
5	W21X73	W24X104	W21X83	W14X74	W18X76	W18X65
6	W18X86	W10X88	W24X103	W18X86	W24X103	W14X90
7	W18X65	W14X74	W21X55	W12X96	W21X68	W10X45
8	W21X68	W27X94	W27X114	W24X68	W14X61	W12X65
9	W18X60	W21X57	W10X33	W10X39	W18X35	W6X25
10	W18X65	W18X71	W18X46	W12X40	W10X33	W10X45
11	W21X44	W21X44	W21X44	W21X44	W21X44	W21X44
Weight (kN)	496.68	452.34	426.36	417.466	412.62	395.35
Differences compared to DE	26 %	14 %	8 %	6 %	4 %	

equal to $E = 200$ GPa and a yield stress of $F_y = 248.2$ MPa. The effective length factors of the members are calculated as $K_x \geq 0$ for a sway-permitted frame and the out-of-plane effective length factor is specified as $K_y = 1.0$. Each column is considered as non-braced along its length, and the unbraced length for each beam member is specified as one-fifth of the span length.

For solving this problem by DE, the Loops number is set to 100. The convergence curve is according to (6.1) considering $PP_1 = 0.15$ and $Power = 1$. R_e and ε are equal to 5 and 1, respectively.

Results of the present study and those of [7, 14, 17] are provided in Table 6.10. It can be seen that the DE achieves results that are 26 %, 14 %, 8 %, 6 % and 4 % lighter than the PSO, PSOPC, HPSACO, ICA and CSS, respectively.

Convergence history is depicted in Fig. 6.23. It can be seen that the present algorithm leads to the best answer in 100 loops which is less than that of the CSS (250 loops).

The maximum value of displacement is 14.27 cm which is less than the allowable limit (23.5 cm).

Figure 6.24 shows the inter-story drifts, the maximum value of which is 1.15 cm. This is less than the allowable value (1.17 cm). It can be recognized that by reducing the weight of structure its stiffness is reduced, then the inter-story drifts are closer to the maximum allowable value.

In Fig. 6.25 the stress ratios of the elements are shown. The maximum stress ratio is 99.69 %. One can see that similar to the inter-story limitation, stress ratios are closer to the limit line.

Figure 6.26 shows the *CF* changes during optimization. It is clear that the *CF* changes around predefined line.

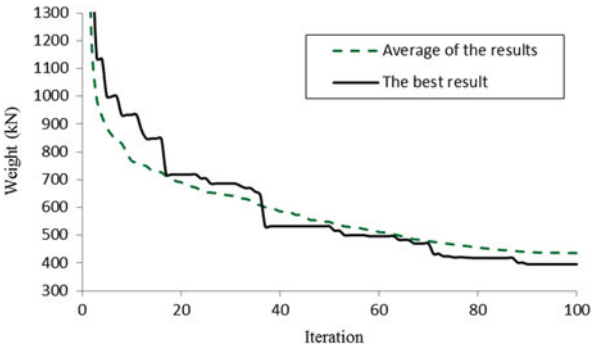

Fig. 6.23 The optimum answer and average answer with the convergence history for the 3-bay 15-story frame using the DE [1]

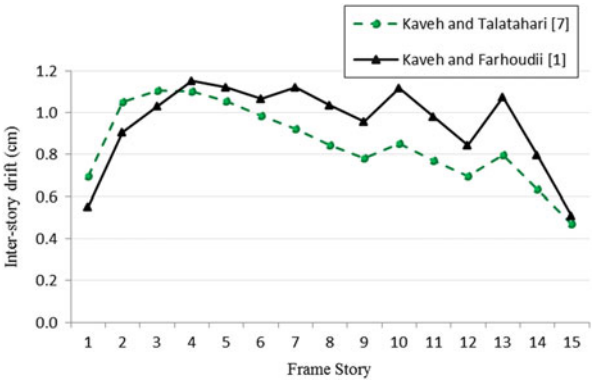

Fig. 6.24 Comparison of the allowable and the existing inter-story drift for the 3-bay 15-story planar frame [1]

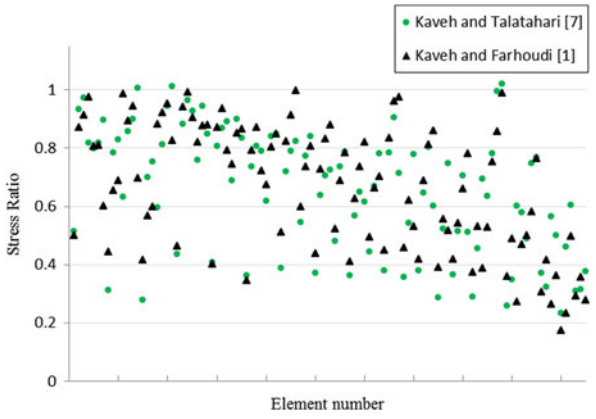

Fig. 6.25 Comparison of the allowable and the existing stress ratios for the 3-bay 15-story planar frame [1]

Fig. 6.26 The optimum answer and the average answer with the convergence factor history for the 3-bay 15-story planar frame using the DE [1]

6.5.2.2 A 3-bay 24-story planar frame

Figure 6.27 shows the topology and the service loading conditions for a 3-bay 24-story frame consisting of 100 joints and 168 members which is chosen from Camp et al. [18]. The frame is designed following the LRFD specification and uses an inter-story drift displacement constraint. The material properties are a modulus of elasticity equal to $E = 205$ GPa and a yield stress of $F_y = 230.3$ MPa.

The effective length factors of the members are calculated as $K_x \geq 0$ for the sway-permitted frame and the out-of-plane effective length factor is specified as $K_y = 1.0$. All columns and beams are considered non-braced along their lengths. Fabrication conditions are imposed on the construction of the 168-element frame requiring that the same beam section be used in the first and third bay on all the floors except the roof beams, resulting in four beam groups.

Beginning at the foundation, the exterior columns are combined into one group and the interior columns are combined together in another group over three consecutive stories. The grouping results in 16 column sections and 4 beam sections for a total of 20 design variables. In this example, each of the four beam element groups is chosen from all 267 W-shapes, while the 16 column element groups are limited to W14 sections (37 W-shapes).

For solving this problem by the DE, the Loops number is set to be equal to 200. The convergence curve is according to (6.1) considering $PP_1 = 0.15$ and $Power = 1$. R_e and ε are equal to 5 and 1, respectively.

Results of the present study and those of Camp et al. [18], Degertekin [19] and Kaveh and Talatahari [7, 17, 20] are provided in Table 6.11. It can be seen that the DE achieves results that are 7.5 %, 4.8 %, 6 %, 3.7 %, and 3.6 % lighter than those of the ACO, HS, IACO, ICA and CSS, respectively.

Convergence history is depicted in Fig. 6.28. It can be observed that DE leads to the best answer in 200 loops which is less than that of CSS being 275 loops.

The maximum value of displacement is 26.11 cm which is less than the allowable limit (29.20 cm).

Figure 6.29 shows the inter-story drifts with maximum value being 1.202 cm that is less than the allowable value (1.205 cm). It can be recognized that by reducing the

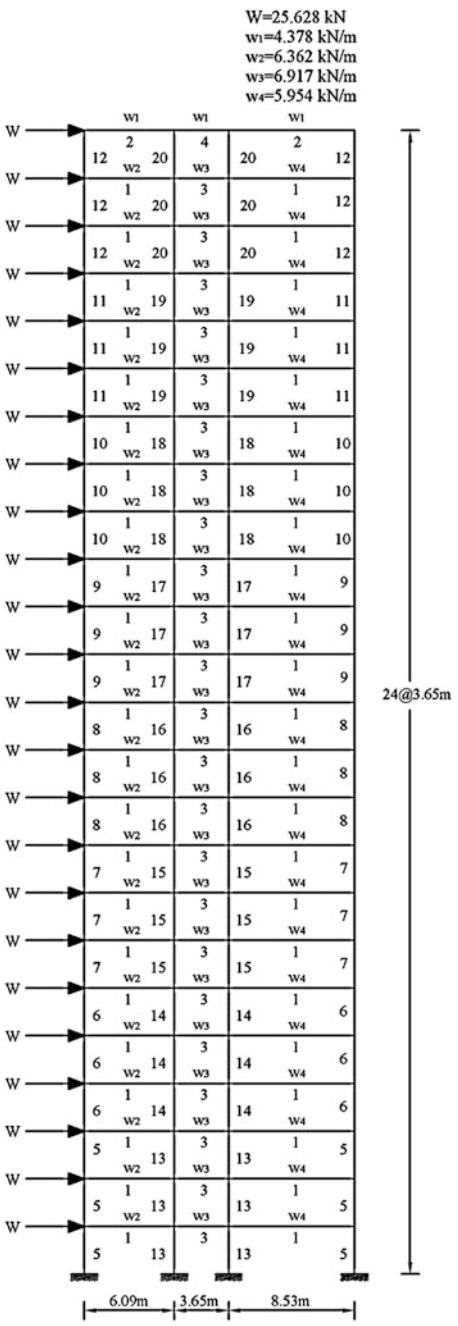

Fig. 6.27 Schematic of a 3-bay 24-story planar frame

Table 6.11 Optimal design comparison for the 3-bay 24-story planar frame

	Optimal W-shaped sections					
	Camp et al. [18]	Degertekin [19]	Kaveh and Talatahari			
Element group	ACO	HS	IACO [20]	ICA [17]	CSS [7]	Present work [1]
1	W30X90	W30X90	W30X99	W30X90		W30X90
2	W8X18	W10X22	W16X26	W21X50	W21X50	W6X20
3	W24X55	W18X40	W18X35	W24X55	W21X48	W21X44
4	W8X21	W12X16	W14X22	W8X28	W12X19	W6X9
5	W14X145	W14X176	W14X145	W14X109	W14X176	W14X159
6	W14X132	W14X176	W14X132	W14X159	W14X145	W14X145
7	W14X132	W14X132	W14X120	W14X120	W14X109	W14X132
8	W14X132	W14X109	W14X109	W14X90	W14X90	W14X99
9	W14X68	W14X82	W14X48	W14X74	W14X74	W14X68
10	W14X53	W14X74	W14X48	W14X68	W14X61	W14X61
11	W14X43	W14X34	W14X34	W14X30	W14X34	W14X43
12	W14X43	W14X22	W14X30	W14X38	W14X34	W14X22
13	W14X145	W14X145	W14X159	W14X159	W14X145	W14X109
14	W14X145	W14X132	W14X120	W14X132	W14X132	W14X109
15	W14X120	W14X109	W14X109	W14X99	W14X109	W14X90
16	W14X90	W14X82	W14X99	W14X82	W14X82	W14X82
17	W14X90	W14X61	W14X82	W14X68	W14X68	W14X74
18	W14X61	W14X48	W14X53	W14X48	W14X43	W14X43
19	W14X30	W14X30	W14X38	W14X34	W14X34	W14X30
20	W14X26	W14X22	W14X26	W14X22	W14X22	W14X26
Weight (kN)	980.63	956.13	967.33	946.25	945.02	912.26
Difference compared to DE	7.5 %	4.8 %	6.0 %	3.7 %	3.6 %	

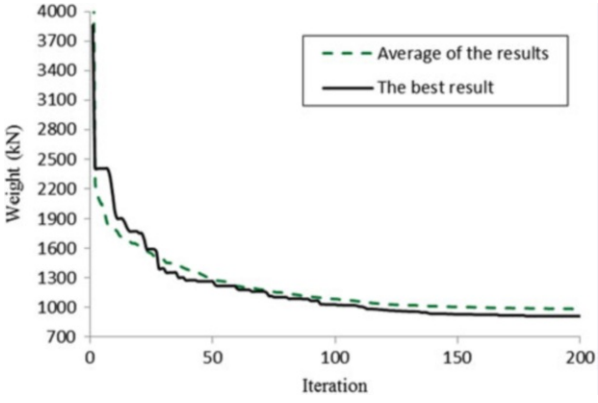

Fig. 6.28 The optimum answer and the average answer, with the convergence history for the 3-bay 24-story frame using the DE [1]

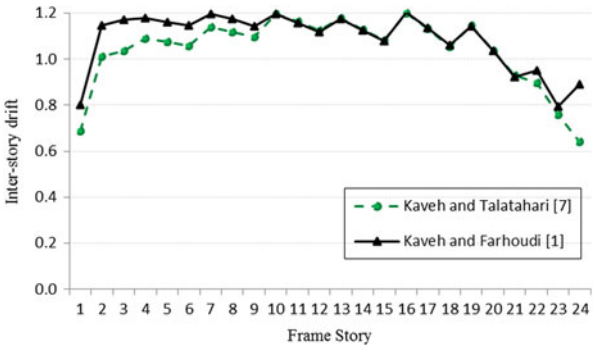

Fig. 6.29 Comparison of the allowable and the existing inter-story drift for the 3-bay 24-story planar frame [1]

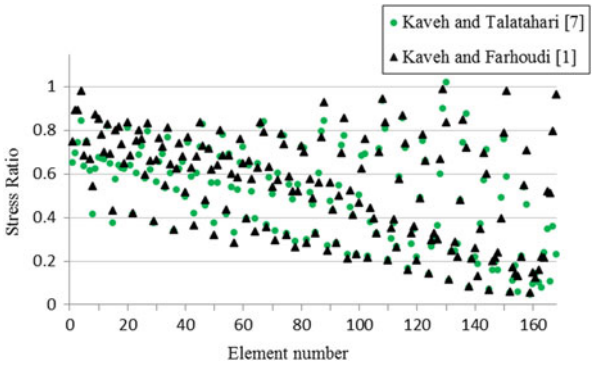

Fig. 6.30 Comparison of the allowable and existing stress ratio for the 3-bay 24-story planar frame [1]

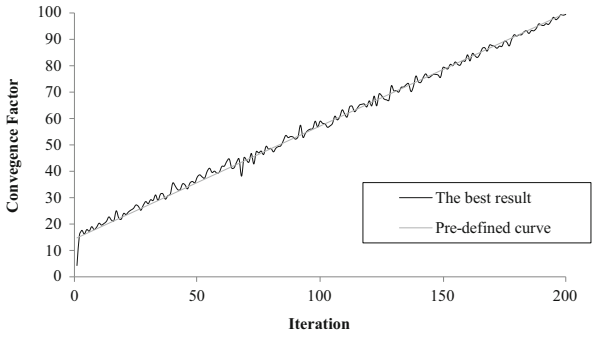

Fig. 6.31 The optimum answer and the average answer with the convergence factor history for the 3-bay 24-story planar frame using the DE [1]

weight of structure its stiffness is reduced and the inter-story drifts are quite close to the maximum allowable value.

In Fig. 6.30 the stress ratios of the elements are shown. One can see that similar to the inter-story limitation, the stress ratios are closer to the limitation line. The maximum stress ratio is 98.33 %.

Figure 6.31 shows the *CF* changes during the optimization process. It is clear that the *CF* changes around the predefined line.

6.5.2.3 Discussion

In this study a novel optimization method is developed based on dolphin echolocation. The new method has the advantage of working according to the computational effort that user can afford for his/her optimization. In this algorithm, the convergence factor defined by Kaveh and Farhoudi [6] is controlled in order to perform a suitable optimization.

For the examples optimized in this chapter, the DE achieves better results with higher convergence rates compared to other existing metaheuristic algorithms such as GA, ACO, PSO, BB–BC, HS, ESs, SGA, TS, ICA, IACO, PSOPC, HPSACO and CSS previously applied to these problems. The authors believe that the results achieved from metaheuristics are mostly dependent on the parameter tuning of the algorithms. It is also believed that by performing a limited number of numerical examples, one cannot correctly conclude the superiority of one method with respect to the others. Dolphin echolocation is an optimization algorithm that has the capability of adopting itself by the type of the problem in hand, having a reasonable convergence rate, and leading to an acceptable optimum answer in a number of loops specified by the user.

References

1. Kaveh A, Farhoudi N (2013) A new optimization method: dolphin echolocation. Adv Eng Softw 59:53–70
2. Griffin DR (1958) Listening in the dark: the acoustic orientation of bats and men. Yale University Press, New Haven, CT, p 413 [Biological Laboratories, Harvard University, Cambridge, MA]
3. Au WWL (1993) The sonar of dolphins. Springer, New York, NY
4. May J (1990) The Greenpeace book of dolphins. Greenpeace Communications Ltd, New York
5. Thomas JA, Moss CF, Vater M (2002) Echolocation in bats and dolphins. University of Chicago Press, Chicago
6. Kaveh A, Farhoudi N (2011) A unified approach to parameter selection in meta-heuristic algorithms for layout optimization. J Constr Steel Res 67:15453–15462
7. Kaveh A, Talatahari S (2012) Charged system search for optimal design of planar frame structures. Appl Soft Comput 12:382–393
8. Wu SJ, Chow PT (1995) Steady-state genetic algorithms for discrete optimization of trusses. Comput Struct 56:979–991
9. Lee KS, Geem ZW, Lee SH, Bae KW (2005) The harmony search heuristic algorithm for discrete structural optimization. Eng Optim 37:663–684
10. Li LJ, Huang ZB, Liu F (2009) A heuristic particle swarm optimization method for truss structures with discrete variables. Comput Struct 87:435–443
11. Kaveh A, Talatahari S (2009) A particle swarm ant colony optimization for truss structures with discrete variables. Comput Struct 87:1129–1140
12. Construction (AISC) (1989) Manual of steel construction allowable stress design, 9th edn. American Institute of Steel Construction, Chicago, IL
13. Hasançebi O, Çarbaş S, Doğan E, Erdal F, Saka MP (2009) Performance evaluation of metaheuristic search techniques in the optimum design of real size pin jointed structures. Comput Struct 87(5–6):284–302

14. Kaveh A, Talatahari S (2009) Hybrid algorithm of harmony search, particle swarm and ant colony for structural design optimization. Stud Comput Intell 239:159–198
15. Sonmez M (2011) Discrete optimum design of truss structures using artificial bee colony algorithm. Struct Multidiscip Optim 43:85–97
16. ANSI/AISC 360–05 (2005) Specification for structural steel buildings. American Institute of Steel Construction, Chicago,IL
17. Kaveh A, Talatahari S (2010) Optimum design of skeletal structures using imperialist competitive algorithm. Comput Struct 88:1220–1229
18. Camp CV, Bichon J, Stovall SP (2005) Design of steel frames using ant colony optimization. J Struct Eng ASCE 131(3):369–379
19. Degertekin SO (2008) Optimum design of steel frames using harmony search algorithm. Struct Multidiscip Optim 36:393–401
20. Kaveh A, Talatahari S (2010) An improved ant colony optimization for design of planar steel frames. Eng Struct 32:864–876

Chapter 7
Colliding Bodies Optimization

7.1 Introduction

This chapter presents a novel efficient metaheuristic optimization algorithm called Colliding Bodies Optimization (CBO), for optimization. This algorithm is based on one-dimensional collisions between bodies, with each agent solution being considered as the massed object or body. After a collision of two moving bodies having specified masses and velocities, these bodies are separated with new velocities. This collision causes the agents to move toward better positions in the search space. CBO utilizes simple formulation to find minimum or maximum of functions; also it is internally parameter independent [1].

This chapter consists of two parts. In the first part the main algorithm is developed and three well-studied engineering design problems and two structural design problems taken from the optimization literature are used to investigate the efficiency of the proposed approach [1]. In the second part, the CBO is applied to a number of continuous optimization benchmark problems. These examples include three well-known space trusses and two planar bridge structures [2].

7.2 Colliding Bodies Optimization

The main goal of this section is to introduce a simple optimization algorithm based on the collision between objects, which is called Colliding Bodies Optimization (CBO).

A. Kaveh, *Advances in Metaheuristic Algorithms for Optimal Design of Structures*,
DOI 10.1007/978-3-319-05549-7_7,

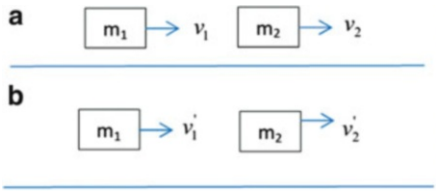

Fig. 7.1 The collision between two bodies. (**a**) before the collision (**b**) after the collision [1]

7.2.1 The Collision Between Two Bodies

Collisions between bodies are governed by the laws of momentum and energy. When a collision occurs in an isolated system (Fig. 7.1), the total momentum of the system of objects is conserved. Provided that there are no net external forces acting upon the objects, the momentum of all objects before the collision equals the momentum of all objects after the collision.

The conservation of the total momentum demands that the total momentum before the collision is the same as the total momentum after the collision, and can be expressed by the following equation:

$$m_1 v_1 + m_2 v_2 = m_1 v_1^{'} + m_2 v_2^{'} \tag{7.1}$$

Likewise, the conservation of the total kinetic energy is expressed as:

$$\frac{1}{2} m_1 {v_1}^2 + \frac{1}{2} m_2 {v_2}^2 = \frac{1}{2} m_1 v_1^{'} 2 + \frac{1}{2} m_2 v_2^{'} 2 + Q \tag{7.2}$$

Where v_1 is the initial velocity of the first object before impact, v_2 is the initial velocity of the second object before impact, $v_1^{'}$ is the final velocity of the first object after impact, $v_2^{'}$ is the final velocity of the second object after impact, m_1 is the mass of the first object, m_2 is the mass of the second object and Q is the loss of kinetic energy due to the impact [3].

The formulas for the velocities after a one-dimensional collision are:

$$v_1^{'} = \frac{(m_1 - \varepsilon m_2) v_1 + (m_2 + \varepsilon m_2) v_2}{m_1 + m_2} \tag{7.3}$$

$$v_2^{'} = \frac{(m_2 - \varepsilon m_1) v_2 + (m_1 + \varepsilon m_1) v_1}{m_1 + m_2} \tag{7.4}$$

Where ε is the Coefficient Of Restitution (COR) of the two colliding bodies, defined as the ratio of relative velocity of separation to relative velocity of approach:

$$\varepsilon = \frac{\left|v_2' - v_1'\right|}{\left|v_2 - v_1\right|} = \frac{v'}{v} \tag{7.5}$$

According to the coefficient of restitution, there are two special cases of any collision as follows:

1. A perfectly elastic collision is defined as the one in which there is no loss of kinetic energy in the collision ($Q = 0 \quad and \quad \varepsilon = 1$). In reality, any macroscopic collision between objects will convert some kinetic energy to internal energy and other forms of energy. In this case, after collision, the velocity of separation is high.
2. An inelastic collision is the one in which part of the kinetic energy is changed to some other form of energy in the collision. Momentum is conserved in inelastic collisions (as it is for elastic collisions), but one cannot track the kinetic energy through the collision since some of it will be converted to other forms of energy. In this case, coefficient of restitution does not equal to one ($Q \neq 0 \quad \& \quad \varepsilon \leq 1$). In this case, after collision the velocity of separation is low.

   For the most real objects, the value of ε is between 0 and 1.

7.2.2 The CBO Algorithm

7.2.2.1 Theory

The main objective of the present study is to formulate a new simple and efficient metaheuristic algorithm which is called Colliding Bodies Optimization (CBO). In CBO, each solution candidate X_i containing a number of variables (i.e. $X_i = \{X_{i,j}\}$) is considered as a colliding body (CB). The massed objects are composed of two main equal groups; i.e. stationary and moving objects, where the moving objects move to follow stationary objects and a collision occurs between pairs of objects. This is done for two purposes: (1) to improve the positions of moving objects; (2) to push stationary objects towards better positions. After the collision, new positions of colliding bodies are updated based on new velocity by using the collision laws as discussed in Sect. 7.2.

The CBO procedure can briefly be outlined as follows:

1. The initial positions of CBs are determined with random initialization of a population of individuals in the search space:

$$x_i^0 = x_{\min} + rand(x_{\max} - x_{\min}), \quad i = 1, 2, \ldots, n, \tag{7.6}$$

Where, x_i^0 determines the initial value vector of the i th CB. $x_{\min}$ and $x_{\max}$ are the

minimum and the maximum allowable values vectors of variables; *rand* is a random number in the interval [0,1]; and n is the number of CBs.

2. The magnitude of the body mass for each CB is defined as:

$$m_k = \frac{\frac{1}{fit(k)}}{\sum_{i=1}^{n} \frac{1}{fit(i)}}, \quad k = 1, 2, \ldots, n \tag{7.7}$$

Where $fit(i)$ represents the objective function value of the agent i; n is the population size. It seems that a CB with good values exerts a larger mass than the bad ones. Also, for maximization, the objective function $fit(i)$ will be replaced by $\frac{1}{fit(i)}$.

3. The arrangement of the CBs objective function values is performed in ascending order (Fig. 7.2a). The sorted CBs are equally divided into two groups:

- The lower half of CBs (stationary CBs); These CBs are good agents which are stationary and the velocity of these bodies before collision is zero. Thus:

$$v_i = 0, \quad i = 1, \ldots, \frac{n}{2} \tag{7.8}$$

- The upper half of CBs (moving CBs): These CBs move toward the lower half. Then, according to Fig. 7.2b, the better and worse CBs, i.e. agents with upper fitness value, of each group will collide together. The change of the body position represents the velocity of these bodies before collision as:

$$v_i = x_i - x_{i-\frac{n}{2}}, \quad i = \frac{n}{2} + 1, \ldots, n \tag{7.9}$$

Where, v_i and x_i are the velocity and position vector of the ith CB in this group, respectively; $x_{i-\frac{n}{2}}$ is the ith CB pair position of x_i in the previous group.

4. After the collision, the velocities of the colliding bodies in each group are evaluated utilizing Eqs. (7.3) and (7.4), and the velocity before collision. The velocity of each moving CBs after the collision is obtained by:

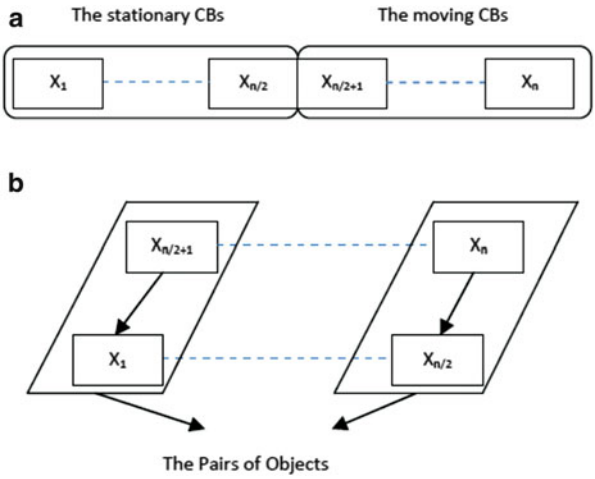

Fig. 7.2 (**a**) CBs sorted in increasing order; (**b**) colliding object pairs [1]

$$v_i^{'} = \frac{\left(m_i - \varepsilon m_{i-\frac{n}{2}}\right) v_i}{m_i + m_{i-\frac{n}{2}}}, \quad i = \frac{n}{2} + 1, \ldots, n \tag{7.10}$$

Where, v_i and $v_i^{'}$ are the velocity of the ith moving CB before and after the collision, respectively; m_i is mass of the i th CB; $m_{i-\frac{n}{2}}$ is mass of the i th CB pair. Also, the velocity of each stationary CB after the collision is:

$$v_i^{'} = \frac{\left(m_{i+\frac{n}{2}} + \varepsilon m_{i+\frac{n}{2}}\right) v_{i+\frac{n}{2}}}{m_i + m_{i+\frac{n}{2}}}, \quad i = 1, \ldots, \frac{n}{2} \tag{7.11}$$

Where, $v_{i+\frac{n}{2}}$ and $v_i^{'}$ are the velocity of the i th moving CB pair before and the i th stationary CB after the collision, respectively; m_i is mass of the i th CB; $m_{i+\frac{n}{2}}$ is mass of the i th moving CB pair; ε is the value of the COR parameter whose law of variation will be discussed in the next section.

5. New positions of CBs are evaluated using the generated velocities after the collision in position of stationary CBs.

   The new positions of each moving CB is:

$$x_i^{new} = x_{i-\frac{n}{2}} + rand \circ v_i^{'}, \quad i = \frac{n}{2} + 1, \ldots, n \tag{7.12}$$

Where, x_i^{new} and $v_i^{'}$ are the new position and the velocity after the collision of the i th moving CB, respectively; $x_{i-\frac{n}{2}}$ is the old position of i th stationary CB pair. Also, the new positions of stationary CBs are obtained by:

$$x_i^{new} = x_i + rand \circ v_i^{'}, \quad i = 1, \ldots, \frac{n}{2} \tag{7.13}$$

Where, x_i^{new}, x_i and $v_i^{'}$ are the new position, old position and the velocity after the collision of the i th stationary CB, respectively. *rand* is a random vector uniformly distributed in the range $(-1,1)$ and the sign "$\circ$" denotes an element-by-element multiplication.

6. The optimization is repeated from Step 2 until a termination criterion, such as maximum iteration number, is satisfied. It should be noted that, a body's status (stationary or moving body) and its numbering are changed in two subsequent iterations.

Apart from the efficiency of the CBO algorithm, which is illustrated in the next section through numerical examples, parameter independency is an important feature that makes CBO superior over other metaheuristic algorithms. Also, the formulation of CBO algorithm does not use the memory which saves the best-so-far solution (i.e. the best position of agents from the previous iterations).

The penalty function approach was used for constraint handling. The fit (i) function corresponds to the effective cost. If optimization constraints are satisfied, there is no penalty; otherwise the value of penalty is calculated as the ratio between the violation and the allowable limit.

7.2.2.2 The Coefficient of Restitution

The metaheuristic algorithms have two phases: exploration of the search space and exploitation of the best solutions found. In the metaheuristic algorithm it is very important to have a suitable balance between the exploration and exploitation. In the optimization process, the exploration should be decreased gradually while simultaneously exploitation should be increased.

In this paper, an index is introduced in terms of the coefficient of restitution (COR) to control exploration and exploitation rate. In fact, this index is defined as the ratio of the separation velocity of two agents after collision to approach velocity of two agents before collision. Efficiency of this index will be shown using one numerical example.

In this section, in order to have a general idea about the performance of COR in controlling local and global searches, a benchmark function (Aluffi-Pentiny) chosen from [4] is optimized using the CBO algorithm. Three variants of COR values are considered. Figure 7.3 is prepared to show the positions of the current CBs in 1st, 50th and 100th iteration for these cases. These three typical cases result in the following:

1. The perfectly elastic collision: In this case, COR is set equal to unity. It can be seen that in the final iterations, the CBs investigate the entire search space to discover a favorite space (global search).

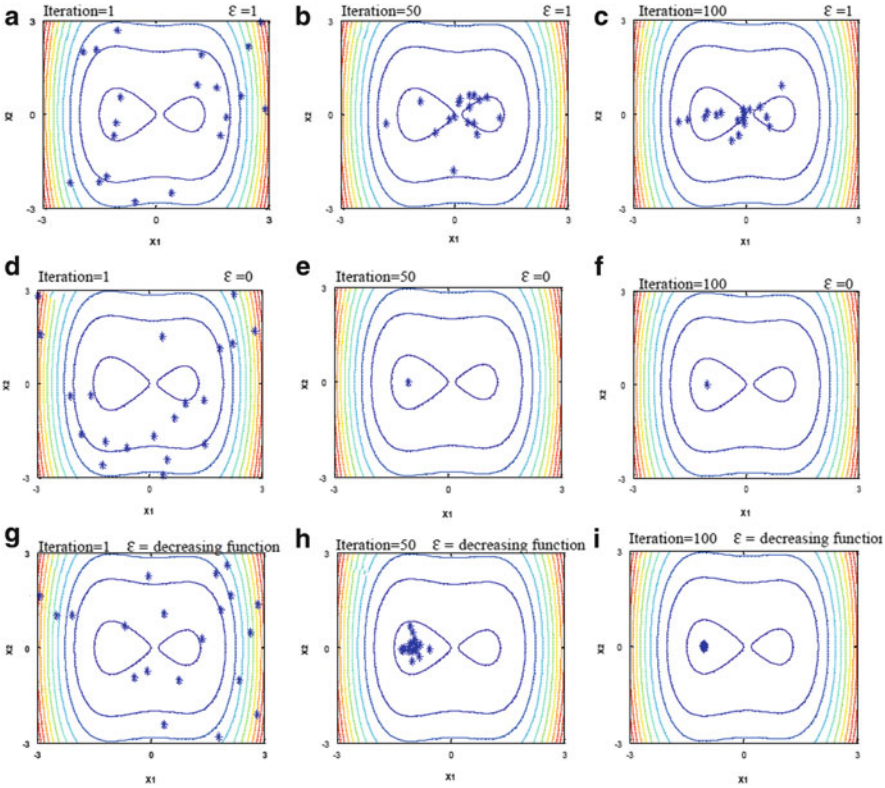

Fig. 7.3 Evolution of the positions of CBs during optimization history for different definitions of the coefficient of restitution (Aluffi-Pentiny benchmark function) [1]

2. The hypothetical collision: In this case, COR is set equal to zero. It can be seen that in the 50th iterations, the movements of the CBs are limited to very small space in order to provide exploitation (local search). Consequently, the CBs are gathered in a small region of the search space.
3. The inelastic collision: In this case, COR decreases linearly to zero and ε is defined as:

$$\varepsilon = 1 - \frac{iter}{iter_{\max}} \tag{7.14}$$

where *iter* is the actual iteration number and $iter_{max}$ is the maximum number of iterations. It can be seen that the CBs get closer by increasing iteration. In this way a good balance between the global and local search is achieved. Therefore, in the optimization process COR is considered such as the above equation.

7.2.3 Test Problems and Optimization Results

Three well-studied engineering design problems and two structural design problems taken from the optimization literature are used to investigate the efficiency of the proposed approach. These examples have been previously studied using a variety of other techniques, which are useful to show the validity and effectiveness of the proposed algorithm. In order to assess the effect of the initial population on the final result, these examples are independently optimized with different initial populations.

For engineering design examples, 30 independent runs were performed for CBO, considering 20 individuals and 200 iterations; the corresponding number of function evaluations is 4,000. The number of function evaluations set for the GA-based algorithm developed by Coello [5], the PSO-based method developed by He and Wang [6], the evolution strategies developed by Montes and Coello [7] is 900,000, 200,000 and 25,000, respectively. Similar to CBO, the number of function evaluation for the charged system search algorithm developed by Kaveh and Talatahari [8] is 4,000.

In the truss design problems, 20 independent runs were carried out, considering 40 individuals and 400 iterations: hence, the maximum number of structural analyses was 16,000. The CBO algorithm was coded in MATLAB. Structural analysis was performed with the direct stiffness method.

7.2.3.1 Example 1: Design of Welded Beam

As the first example, design optimization of the welded beam shown in Fig. 7.4 is carried out. The welded beam design problem was often utilized to evaluate performance of different optimization methods. The objective is to find the best set of design variables to minimize the total fabrication cost of the structure subject to shear stress (τ), bending stress (σ), buckling load (Pc), and end deflection (δ) constraints. Assuming $x_1 = h$, $x_2 = l$, $x_3 = t$, and $x_4 = b$ as the design variables, the mathematical formulation of the problem can be expressed as:

Find

$$\{x_1, x_2, x_3, x_4\} \tag{7.15}$$

To minimize

$$\cos t(x) = 1.10471 x_1^2 x_2 + 0.04811 x_3 x_4 \left(14 + x_2\right) \tag{7.16}$$

Subjected to

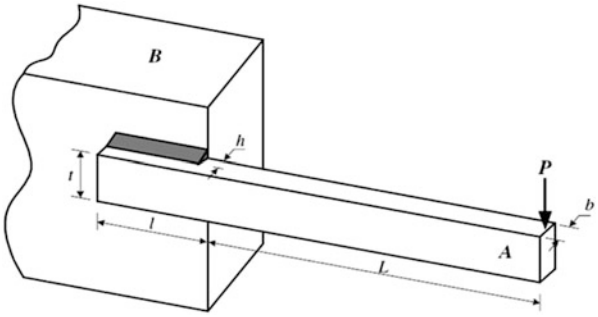

Fig. 7.4 Schematic of the welded beam structure with indication of design variables

$$
\begin{gathered}
g_1(x) = \tau(x) - \tau_{\max} \le 0 \\
g_2(x) = \sigma(x) - \sigma_{\max} \le 0 \\
g_3(x) = x_1 - x_4 \le 0 \\
g_4(x) = 0.10471x_1^2 + 0.04811x_3x_4(14 + x_2) - 5 \le 0 \\
g_5(x) = 0.125 - x_1 \le 0 \\
g_6(x) = \delta(x) - \delta_{\max} \le 0 \\
g_7(x) = p - p_c(x) \le 0
\end{gathered}
\tag{7.17}
$$

The bounds on the design variables are:

$$
0.1 \le x_1 \le 2, \quad 0.1 \le x_2 \le 10, \quad 0.1 \le x_3 \le 10, \quad 0.1 \le x_4 \le 2 \tag{7.18}
$$

Where

$$
\begin{gathered}
\tau(x) = \sqrt{(\tau')^2 + 2\tau'\tau''\frac{x_2}{2R} + (\tau'')^2} \\
\tau' = \frac{P}{\sqrt{2}x_1x_2} \quad \tau'' = \frac{MR}{J} \quad M = P\left(L + \frac{x_2}{2}\right) \quad R = \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2} \\
J = 2\left\{\sqrt{2}x_1x_2\left[\frac{x_2^2}{12} + \left(\frac{x_1 + x_3}{2}\right)^2\right]\right\} \quad \sigma(x) = \frac{6PL}{x_4x_3^2} \quad \delta(x) = \frac{4PL^3}{Ex_3^3x_4} \\
P_c(x) = \frac{4.013\sqrt{E(x_3^2x_4^6/36)}}{L^2}\left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right)
\end{gathered}
\tag{7.19}
$$

The constants in (7.17) and (7.19) are chosen as follows:

$P = 6{,}000$ lb, $L = 14$ in, $E = 30 \times 106$ psi, $G = 12 \times 106$ psi, $\tau_{\max} = 13{,}600$ psi, $\sigma_{\max} = 30{,}000$ psi, and $\delta_{\max} = 0.25$ in.

Radgsdell and Phillips [9] compared optimal results of different optimization methods which were mainly based on mathematical optimization algorithms. Deb [10], Coello [5] and Coello and Montes [11] solved this problem using GA-based

Table 7.1 Comparison of CBO optimized designs with literature for the welded beam problem

	Best solution found							
Design Variables	Ragsdell and Phillips [9]	Deb [10]	Coello [11]	Coello and Montes [11]	He and Wang [6]	Montes and Coello [7]	Kaveh and Talatahari [8]	Present work [1]
x_1 (h)	0.245500	0.248900	0.208800	0.205986	0.202369	0.202369	0.20582	0.205722
x_2 (l)	6.1960000	6.173000	3.420500	3.471328	3.544214	3.544214	3.468109	3.47041
x_3 (t)	8.273000	8.178900	8.997500	9.020224	9.04821	9.04821	9.038024	9.037276
x_4 (b)	0.245500	0.253300	0.210000	0.20648	0.205723	0.205723	0.205723	0.205735
f(x)	1.728024	2.433116	−1.748310	1.728226	1.728024	1.728024	1.724866	1.724663

Table 7.2 Statistical results from different optimization methods for the welded beam design problem

Methods	Best result	Average optimized cost	Worst result	Std dev
Ragsdell and Phillips [9]	2.385937	N/A	N/A	N/A
Deb [10]	2.433116	N/A	N/A	N/A
Coello [5]	1.748309	1.771973	1.785835	0.011220
Coello and Montes [11]	1.728226	1.792654	1.993408	0.074713
He and Wang [6]	1.728024	1.748831	1.782143	0.012926
Montes and Coello [7]	1.737300	1.813290	1.994651	0.070500
Kaveh and Talatahari [8]	1.724866	1.739654	1.759479	0.008064
Present work [1]	1.724662	1.725707	1.725059	0.0002437

methods. Also, He and Wang [6] used effective co-evolutionary particle swarm optimization, Montes and Coello [7] solved this problem utilizing evolution strategies, and Kaveh and Talatahari [8] employed charged system search.

Table 7.1 compares the optimized design and the corresponding cost obtained by CBO with those obtained by other metaheuristic algorithms documented in literature. It can be seen that the best solution obtained by CBO is better than those quoted for the other algorithms. The statistical data on 30 independent runs reported in Table 7.2 also demonstrate the better search ability of CBO with respect to the other algorithms: in fact the best, worst and average costs, and the standard deviation (S.D.) of the obtained solutions are better than literature. The lowest standard deviation achieved by CBO proves that the present algorithm is more robust than other metaheuristic methods.

7.2.3.2 Test Problem 2: Design of a Pressure Vessel

Design optimization of the cylindrical pressure vessel capped at both ends by hemispherical heads (Fig. 7.5) is considered as the second example. The objective of optimization is to minimize the total manufacturing cost of the vessel based on the combination of welding, material and forming costs. The vessel is designed for a working pressure of 3,000 psi and a minimum volume of 750 ft^3 regarding the provisions of *ASME* boiler and pressure vessel code. Here, the shell and head thicknesses should be multiples of 0.0625 in. The thickness of the shell and head is restricted to 2 in. The shell and head thicknesses must not be less than 1.1 in and 0.6 in respectively. The design variables of the problem are x_1 as the shell thickness (T_s), x_2 as the spherical head thickness (T_h), x_3 as the radius of cylindrical shell (R), and x_4 as the shell length (L). The problem formulation is as follows:

Find

$$\{x_1, x_2, x_3, x_4\} \tag{7.20}$$

To minimize

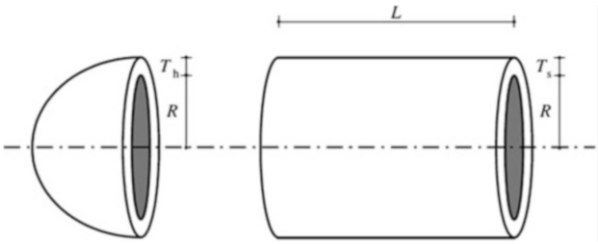

Fig. 7.5 Schematic of the spherical head and cylindrical wall of the pressure vessel with indication of design variables

$$\cos t(x) = 0.6224x_3x_1x_4 + 1.7781x_3^2x_2 + 3.1611x_1^2x_4 + 19.8621x_3x_1^2 \quad (7.21)$$

Subject to

$$\begin{gathered} g_1(x) = 0.0193x_3 - x_1 \leq 0 \\ g_2(x) = 0.00954x_3 - x_2 \leq 0 \\ g_3(x) = 750 \times 1728 - \pi x_3^2 x_4 - \frac{4}{3}\pi x_3^3 \leq 0 \\ g_4(x) = x_4 - 240 \leq 0 \end{gathered} \quad (7.22)$$

The bounds on the design variables are:

$$1.125 \leq x_1 \leq 2, \quad 0.625 \leq x_2 \leq 2, \quad 10 \leq x_3 \leq 240, \quad 10 \leq x_4 \leq 240 \quad (7.23)$$

It can be seen from Table 7.3 that the present algorithm found the best design overall which is about 3 % lighter than the best known design quoted in literature (5,889.911 vs. 6,059.088 of [8]). The statistical data reported in Table 7.4 indicate that the standard deviation of CBO optimized solutions is the third lowest among those quoted for the different algorithms compared in this test case. Statistical results given in Table 7.4 indicate that CBO is in general more robust than the other metaheuristic algorithms. However, the worst optimized design and standard deviation found by CBO are higher than for CSS.

7.2.3.3 Test Problem 3: Design of a Tension/Compression Spring

This problem was first described by Belegundu [15] and Arora [16]. It consists of minimizing the weight of a tension/compression spring subject to constraints on shear stress, surge frequency, and minimum deflection as shown in Fig. 7.6. The design variables are the wire diameter d $(= x_1)$; the mean coil diameter D $(= x_2)$, and the number of active coils N $(= x_3)$. The problem can be stated as follows:

Find

$$\{x_1, x_2, x_3\} \quad (7.24)$$

To minimize

Table 7.3 Comparison of CBO optimized designs with literature for the pressure vessel problem

Methods	X_1 (T_s)	X_2 (T_h)	X_3 (R)	X_4 (L)
Sandgren [12]	1.125000	0.625000	47.70000	117.7010
Kannan and Kramer [13]	1.125000	0.625000	58.29100	43.6900
Deb and Gene [14]	0.937500	0.500000	48.32900	112.6790
Coello [5]	0.812500	0.437500	40.32390	200.0000
Coello and Montes [11]	0.812500	0.437500	42.09739	176.6540
He and Wang [6]	0.812500	0.437500	42.09126	176.7465
Montes and Coello [7]	0.812500	0.437500	42.09808	176.6405
Kaveh and Talatahari [8]	0.812500	0.812500	0.812500	176.572656
Present work [1]	0.779946	0.385560	40.409065	198.76232

Table 7.4 Statistical results from different optimization methods for the pressure vessel problem

Methods	Best result	Average optimized cost	Worst result	Std Dev
Sandgren [12]	8,129.103	N/A	N/A	N/A
Kannan and Kramer [13]	7,198.042	N/A	N/A	N/A
Deb and Gene [14]	6,410.381	N/A	N/A	N/A
Coello [5]	6,288.744	6,293.843	6,308.149	7.4133
Coello and Montes [11]	6,059.946	6,177.253	6,469.322	130.9297
He and Wang [6]	6,061.077	6,147.133	6,363.804	86.4545
Montes and Coello [7]	6,059.745	6,850.004	7,332.879	426.0000
Kaveh and Talatahari [8]	6,059.088	6,067.906	6,085.476	10.256
Present work [1]	5,889.911	5,934.201	6,213.006	63.5417

Fig. 7.6 Schematic of the tension/compression spring with indication of design variables

$$\cos t(x) = (x_3 + 2)x_2x_1^2 \tag{7.25}$$

Subject to

$$\begin{aligned} g_1(x) &= 1 - \frac{x_2^3 x_3}{71785x_1^4} \le 0 \\ g_2(x) &= \frac{4x_2^2 - x_1x_2}{12566(x_2x_1^3 - x_1^4)} + \frac{1}{5108x_1^2} - 1 \le 0 \\ g_3(x) &= 1 - \frac{140.45x_1}{x_2^2x_3} \le 0 \\ g_4(x) &= \frac{x_1 + x_2}{1.5} - 1 \le 0 \end{aligned} \tag{7.26}$$

The bounds on the design variables are:

$$0.05 \leq x_1 \leq 2, \quad 0.25 \leq x_2 \leq 1.3, \quad 2 \leq x_3 \leq 15, \tag{7.27}$$

This problem has been solved by Belegundu [15] using eight different mathematical optimization techniques. Arora [16] also solved this problem using a numerical optimization technique called a constraint correction at the constant cost. Coello [5] as well as Coello and Montes [11] solved this problem using GA-based method. Additionally, He and Wang [6] utilized a co-evolutionary particle swarm optimization (CPSO). Recently, Montes and Coello [7], Kaveh and Talatahari [8] used evolution strategies and the CSS to solve this problem, respectively.

Tables 7.5 and 7.6 compare the best results obtained in this paper and those of the other researches. Once again, CBO found the best design overall. In fact, the lighter design found by Kaveh and Talatahari in [8] actually violates the first two optimization constraints. The statistical data reported in Table 7.6 show that the standard deviation on optimized cost seen for CBO is fully consistent with literature.

7.2.3.4 Test Problem 4: Weight Minimization of the 120-Bar Truss Dome

The fourth test case solved in this study is the weight minimization problem of the 120-bar truss dome shown in Fig. 7.7. This test case was investigated by Soh and Yang [17] as a configuration optimization problem. It has been solved later as a sizing optimization problem by Lee and Geem [18], Kaveh and Talatahari [8] and Kaveh and Khayatazad [19].

The allowable tensile and compressive stresses are set according to the AISC ASD (1989) [20] code, as follows:

$$\begin{cases} \sigma_i^+ = 0.6F_y & for \sigma_i \geq 0 \\ \sigma_i^- & for \sigma_i \leq 0 \end{cases} \tag{7.28}$$

where σ_i^- is calculated according to the slenderness ratio

$$\sigma_i^- = \begin{cases} \left[\left(1 - \dfrac{\lambda_i^2}{2C_c^2}\right)F_y\right] \Big/ \left(\dfrac{5}{3} + \dfrac{3\lambda_i}{8C_c} - \dfrac{\lambda_i^3}{8C_c^3}\right) & for \lambda_i < C_c \\ \dfrac{12\pi^2 E}{23\lambda_i^2} & for \lambda_i \geq C_c \end{cases} \tag{7.29}$$

Table 7.5 Comparison of CBO optimized designs with literature for the tension/compression spring problem

	Optimal design variables			Constraints				
Methods	x_1 (d)	x_2 (D)	x_3 (N)	$g_1(x)$	$g_2(x)$	$g_3(x)$	$g_4(x)$	f(x)
Belegundu [15]	0.050000	0.315900	14.250000	−0.000014	−0.003782	−3.938302	−0.756067	0.0128334
Arora [16]	0.053396	0.399180	9.185400	−0.053396	−0.000018	−4.123832	−0.698283	0.0127303
Coello [5]	0.051480	0.351661	11.632201	−0.002080	−0.000110	−4.026318		0.0127048
Coello and Montes [11]	0.051989	0.363965	10.890522	−0.000013	−0.000021	−4.061338	−0.722698	0.0126810
He and Wang [6]	0.051728	0.357644	11.244543	−0.000845	−1.2600e−05	−4.051300	−0.727090	0.0126747
Montes and Coello [7]	0.051643	0.355360	11.397926	−0.001732	−0.0000567	−4.039301	−0.728664	0.012698
Kaveh and Talatahari [8]	0.051744	0.358532	11.165704	8.78603e−6	0.0011043	−4.063371	−0.726483	0.0126384
Present work [1]	0.051894	0.3616740	11.007846	−3.1073e−4	−1.4189e−5	−4.061846	−0.724287	0.0126697

Table 7.6 Statistical results from different optimization methods for tension/compression string problem

Methods	Best result	Average optimized cost	Worst result	Std Dev
Belegundu [15]	0.0128334	N/A	N/A	N/A
Arora [16]	0.0127303	N/A	N/A	N/A
Coello [5]	0.0127048	0.012769	0.012822	3.9390e−5
Coello and Montes [11]	0.0126810	0.0127420	0.012973	5.9000e−5
He and Wang [6]	0.0126747	0.012730	0.012924	5.1985e−5
Montes and Coello [7]	0.012698	0.013461	0.16485	9.6600e−4
Kaveh and Talatahari [8]	0.0126384	0.012852	0.013626	8.3564e−5
Present work [1]	0.126697	0.1272964	0.128808	5.00376e−5

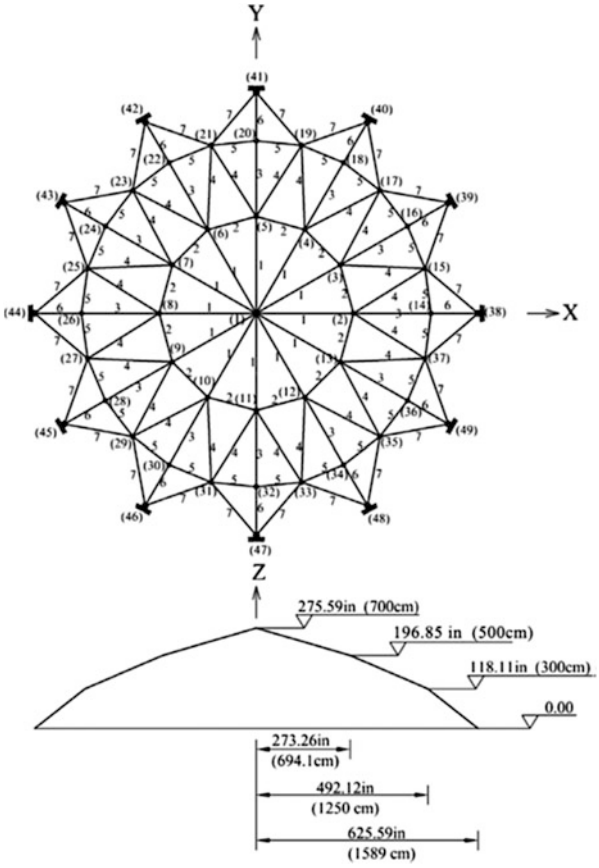

Fig. 7.7 Schematic of the spatial 120-bar dome truss with indication of design variables and main geometric dimensions

where E is the modulus of elasticity, F_y is the yield stress of steel, C_c is the slenderness ratio (λ_i) dividing the elastic and inelastic buckling regions ($C_c = \sqrt{2\pi^2 E / F_y}$), λ_i is the slenderness ratio ($\lambda_i = \frac{KL_i}{r_i}$), K is the effective length factor, L_i is the member length and r_i is the radius of gyration.

The modulus of elasticity is 30,450 ksi (210,000 MPa) and the material density is 0.288 lb/in^3 (7,971.810 kg/m^3). The yield stress of steel is taken as 58.0 ksi (400 MPa). On the other hand, the radius of gyration (r_i) is expressed in terms of cross-sectional areas as $r_i = aA_i^b$ [28]. Here, a and b are constants depending on the types of sections adopted for the members such as pipes, angles, and tees. In this example, pipe sections ($a = 0.4993$ and $b = 0.6777$) are adopted for bars. All members of the dome are divided into seven groups, as shown in Fig. 7.7. The dome is considered to be subjected to vertical loads at all the unsupported joints. These are taken as −13.49 kips (60 kN) at node 1, −6.744 kips (30 kN) at nodes 2 through 14, and −2.248 kips (10 kN) at the remaining of the nodes. The minimum cross-sectional area of elements is 0.775 in^2 (cm^2). In this example, four cases of constraints are considered: with stress constraints and no displacement constraints (Case 1), with stress constraints and displacement limitations of ±0.1969 in (5 mm) imposed on all nodes in x- and y-directions (Case 2), no stress constraints but displacement limitations of ±0.1969 in (5 mm) imposed on all nodes in z-directions (Case 3), and all constraints explained above (Case 4). For Case 1 and Case 2, the maximum cross-sectional area is 5.0 in^2 (32.26 cm^2) while for Case 3 and Case 4 is 20.0 in^2 (129.03 cm^2).

Table 7.7 compares the optimization results obtained in this study with previous research presented in literature. It can be seen that CBO always designed the lightest structure except for Cases 3 and 4 where HPSACO converged to a slightly lower weight. CBO always completed the optimization process within 16,000 structural analyses (40 agents × 400 optimization iterations) while HPSACO required on average 10,000 analyses (400 optimization iterations) and PSOPC required 125,000 analyses (2,500 iterations). The average number of analyses required by the RO algorithm was instead 19,900. Figure 7.8 shows that the convergence rate of CBO is considerably higher than that of PSO and PSOPC.

7.2.3.5 Test Problem 5: Design of Forth Truss Bridge

The last test case was the layout optimization of the forth bridge shown in Fig. 7.9a which is a 16 m long and 1 m high truss of infinite span. Because of infinite span, the cross section of the bridge can be modeled as symmetric about the axis joining nodes 10 and 11. Structural symmetry allowed the 37 elements of which the bridge

Table 7.7 Comparison of CBO optimized designs with literature in the 120-bar dome problem

	Optimal cross-sectional areas (in^2)									
	Case 1					Case 2				
	Kaveh and Khayatazad [19]					Kaveh and Khayatazad [19]				
Element group	PSO	PSOPC	HPSACO	RO	Present work [1]	PSO	PSOPC	HPSACO	RO	Present work [1]
1	3.147	3.235	3.311	3.128	3.1229	15.97	3.083	3.779	3.084	3.0832
2	6.376	3.370	3.438	3.357	3.3538	9.599	3.639	3.377	3.360	3.3526
3	5.957	4.116	4.147	4.114	4.1120	7.467	4.095	4.125	4.093	4.0928
4	4.806	2.784	2.831	2.783	2.7822	2.790	2.765	2.734	2.762	2.7613
5	0.775	0.777	0.775	0.775	0.7750	4.324	1.776	1.609	1.593	1.5918
6	13.798	3.343	3.474	3.302	3.3005	3.294	3.779	3.533	3.294	3.2927
7	2.452	2.454	2.551	2.453	2.4458	2.479	2.438	2.539	2.434	2.4336
Best weight (Ib)	32,432.9	19,618.7	19,491.3	19,476.193	19,454.7	41,052.7	20,681.7	20,078.0	20,071.9	20,064.5
Average weight (Ib)	–	–	–	–	19,466.0	–	–	–	–	20,098.3
Std (Ib)	–	–	–	33.966	7.02	–	–	–	112.135	26.17
	Case 3					Case 4				
1	1.773	2.098	2.034	2.044	2.0660	12.802	3.040	3.095	3.030	3.0284
2	17.635	16.444	15.151	15.665	15.9200	11.765	13.149	14.405	14.806	14.9543
3	7.406	5.613	5.901	5.848	5.6785	5.654	5.646	5.020	5.440	5.4607
4	2.153	2.312	2.254	2.290	2.2987	6.333	3.143	3.352	3.124	3.1214
5	15.232	8.793	9.369	9.001	9.0581	6.963	8.759	8.631	8.021	8.0552
6	19.544	3.629	3.744	3.673	3.6365	6.492	3.758	3.432	3.614	3.3735
7	0.800	1.954	2.104	1.971	1.9320	4.988	2.502	2.499	2.487	2.4899
Weight (lb)	46,893.5	31,776.2	31,670.0	31,733.2	31,724.1	51,986.2	33,481.2	33,248.9	33,317.8	33,286.3
Average weight (Ib)	–	–	–	–	32,162.4	–	–	–	–	33,398.5
Std (Ib)	–	–	–	274.991	240.22	–	–	–	354.333	67.09

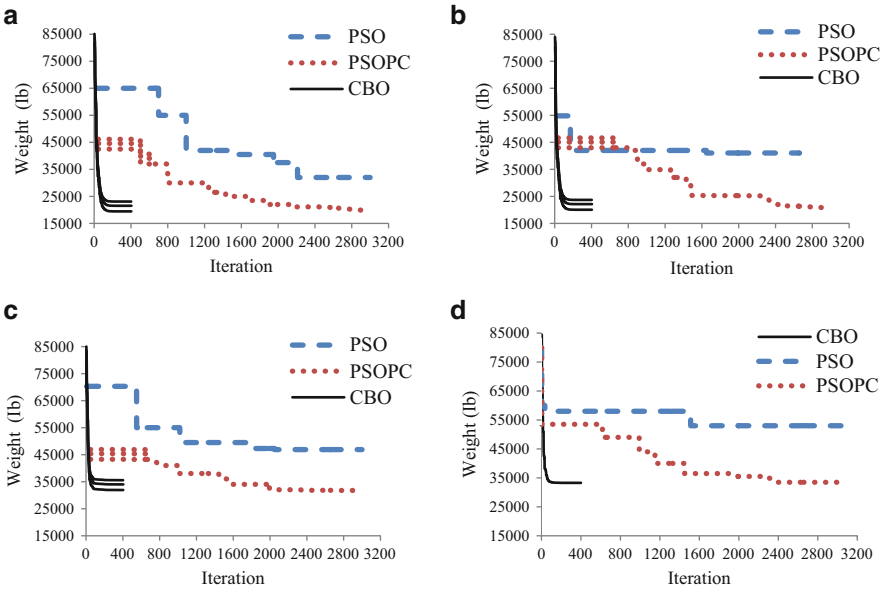

Fig. 7.8 Convergence curves obtained for the different variants of the 120-bar dome problem [2]

is comprised to be grouped into 16 groups (see Table 7.8): hence, there are 16 independent sizing variables. Nodal coordinates were included as layout variables: X-coordinates of nodes could not vary while Y-coordinates (except those of nodes 1 and 20) were allowed to change between −140 and 140 cm with respect to the initial configuration of Fig. 7.9a. Thus, the optimization problem included also 10 layout variables. The cross-sectional areas (sizing variables) could vary between 0.5 and 100 cm^2.

Material properties were set as follows: modulus of elasticity of 210 GPa, allowable stress of 250 MPa, specific weight of 7.8 ton/m^3. The structure is subject to self-weight and concentrated loads shown in Fig. 7.9a.

Table 7.8 compares CBO optimization results with literature. It appears that CBO found the best design overall saving about 1,000 kg with respect to the optimum currently reported in literature. Furthermore, the standard deviation on optimized weight observed for CBO in 20 independent optimization runs was lower than for the other metaheuristic optimization algorithms taken as basis of comparison.

The optimized layout of the bridge is shown in Fig. 7.9b. Figure 7.10 compares the convergence behavior of CBO and RO. Although RO was considerably faster in the early optimization iterations, CBO converged to a significantly better design without being trapped in local optima.

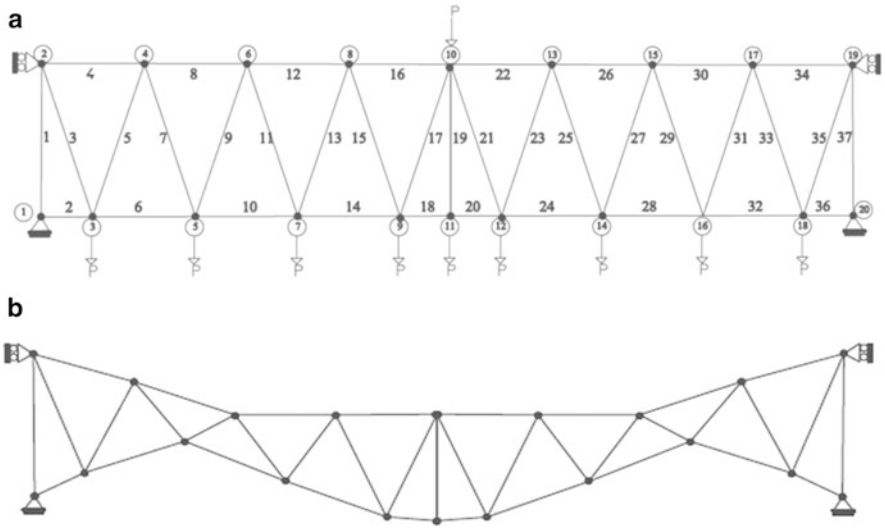

Fig. 7.9 (**a**) Schematic of the Forth truss bridge (**b**) Optimized layout of the forth bridge [2]

7.3 CBO for Optimum Design of Truss Structures with Continuous Variables

This part considers: (1) The CBO algorithm is introduced for optimization of continuous problems. (2) A comprehensive study of sizing optimization for truss structures is presented. The examples are chosen from the literature to verify the effectiveness of the algorithm. These examples are as follows: a 25-member spatial truss with 8 design variables, a 72-member spatial truss with 16 design variables, a 582-member space truss tower with 32 design variables, a 37-member plane truss bridge with 16 design variables, and a 68-member plane truss bridge with 4, 8 and 12 design variables. All the structures are optimized for minimum weight with CBO algorithm, and a comparison is carried out in terms of the best optimum solutions and their convergence rates in a predefined number of analyses. The results indicate that the proposed algorithm is very competitive with other state-of-the-art metaheuristic methods.

7.3.1 Flowchart and CBO Algorithm

The flowchart of the CBO algorithm is shown in Fig. 7.11. The main steps of CBO algorithm are as follows:

Level 1: **Initialization**

Table 7.8 Comparison of CBO optimization results with literature for the forth bridge problem

		Kaveh and Khayatazad [19]			
No	Design variable	BB–BC	PSO	RO	Present Work [2]
1	A_1	56.41	25.20	20.54	23.314
2	A_2	58.20	97.60	44.62	36.867
3	A_3,A_5	53.89	35.00	6.37	9.847
4	A_4	60.21	64.30	50.10	49.679
5	A_6	56.27	14.51	30.39	26.563
6	A_7	57.08	37.91	17.61	12.737
7	A_8	49.19	69.85	41.04	37.120
8	A_{10}	48.67	8.76	8.55	1.545
9	A_9,A_{11}	45.43	47.54	33.93	28.35
10	A_{12}	15.14	6.36	0.63	0.891
11	A_{14}	45.31	27.13	26.92	24.110
12	A_{13}	62.91	3.82	23.42	9.112
13	A_{18}	56.77	50.82	42.06	29.071
14	A_{15},A_{17}	46.66	2.70	2.01	8.222
15	A_{16}	57.95	5.46	8.51	8.715
16	A_{19}	54.99	17.62	1.27	2.107
17	Δy_2, Δy_{19}	6.89	140	70.88	11.093
18	Δy_3, Δy_{18}	17.74	139.65	64.88	50.352
19	Δy_4, Δy_{17}	1.81	117.59	−6.99	−50.529
20	Δy_5, Δy_{16}	23.57	139.70	128.31	119.315
21	Δy_6, Δy_{15}	3.22	−16.51	−64.24	−124.378
22	Δy_7, Δy_{14}	5.85	139.06	139.29	34.219
23	Δy_8, Δy_{13}	4.01	−127.74	−109.62	−120.867
24	Δy_9, Δy_{12}	10.52	−81.03	21.82	−41.323
25	Δy_{10}	−25.99	60.16	−55.09	−115.609
26	Δy_{11}	2.74	−139.97	2.29	−54.590
Best weight (kg)		37,132.3	20,591.9	11,215.7	10,250.9
Average weight (kg)		40,154.1	25,269.3	11,969.2	11,112.63
Std (kg)		1,235.4	2,323.7	545.5	522.54

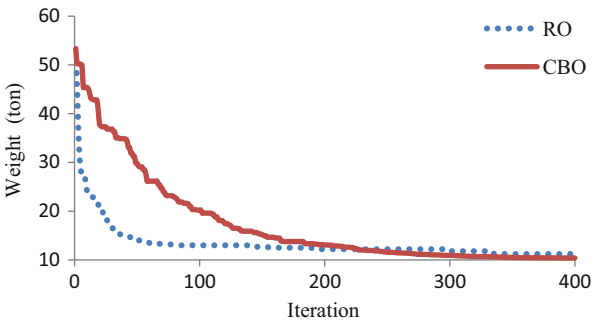

Fig. 7.10 Convergence curves obtained in the forth bridge problem [2]

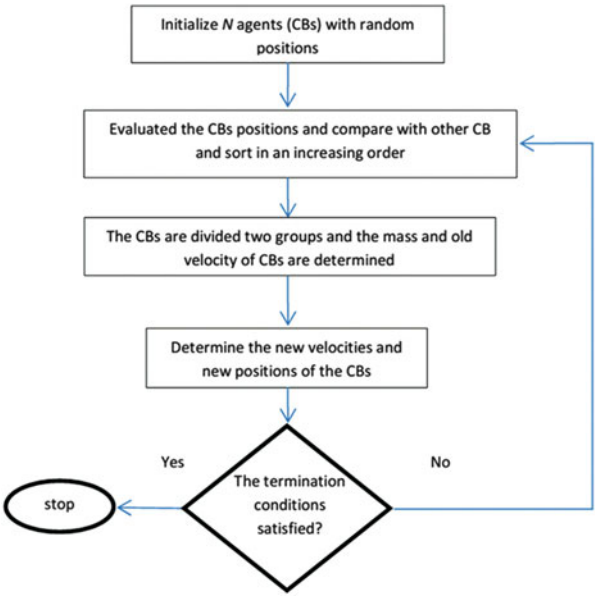

Fig. 7.11 The flowchart of the CBO [2]

- **Step 1**: *Initialization*. Initialize an array of CBs with random positions and their associated values of the objective function (Eq. 7.6).

Level 2: **Search**

- **Step 1**: *CBs ranking*. Compare the value of the objective function for each CB, and sort them in an increasing order.
- **Step 2**: *Groups creation*. CBs are divided into two equal groups: (1) stationary group, (2) moving group. Then, the pairs of CB are defined for collision (Fig. 7.2).
- **Step 3**: *Criteria before the collision*. The value of mass and velocity of each CB for each group are evaluated before the collision (Eqs. 7.7, 7.8, and 7.9).
- **Step 4**: *Criteria after the collision*. The value velocity of each CB in each groups are evaluated after the collision (Eqs. 7.10 and 7.11).
- **Step 5**: *CBs updating*. The new position of each CB is calculated (Eqs. 7.13 and 7.14).

Level 3: **Terminating criterion control**

- **Step 1**: Repeat search level steps until a terminating criterion is satisfied.

7.3.2 Numerical Examples

In order to assess the effectiveness of the proposed methodology a number of continuous optimization benchmark problems are examined. These examples include three well-known space trusses and two planar bridge structures. The number of design variables for the first to fifth examples are 8, 16, 32, 26, respectively and for the last example 4, 8 and 12 variables are used. Similarly, the number of Colliding Bodies or agents for these examples are considered as 30, 40, 50, 40 and 20, respectively. For all of these examples the maximum number of iteration is considered as 400. The algorithm and the direct stiffness method for the analysis of truss structures are coded in Matlab software.

For the sake of simplicity and to be fair in comparisons, the penalty approach is used for the constraint handling. The constrained objective function can formally be stated as follows:

$$Mer(X) = f(X) \times f_{penalty}(X) = f(X) \times \left(1 + \varepsilon_1 \sum_{i=1}^{ni} \max(0, g_i(x))\right)^{\varepsilon_2} \qquad (7.30)$$

where X is the vector of design variables, g_i is the ith constraint from n_i inequality constraints ($g_i(X) \leq 0, i = 1, 2, \ldots, n_i$), and $Mer(X)$ is the merit function; $f(X)$ is the weight of structure; $f_{penalty}(X)$ is the penalty function which results from the violations of the constraints corresponding to the response of the structure. The parameters ε_1 and ε_2 are selected considering the exploration and the exploitation rate of the search space. In this study, ε_1 is selected as unity and ε_2 is taken as 1.5 at the start and linearly increases to 6.

7.3.2.1 A 25-Bar Spatial Truss

Size optimization of the 25-bar planar truss shown in Fig. 7.12 is considered. This is a well-known problem in the field of weight optimization of the structures. In this example, the material density is considered as 0.1 lb/in^3 (2,767.990 kg/m^3) and the modulus of elasticity is taken as 10,000 ksi (68,950 MPa). Table 7.9 shows the two load cases for this example. The structure includes 25 members, which are divided into eight groups, as follows: (1) A_1, (2) A_2–A_5, (3) A_6–A_9, (4) A_{10}–A_{11}, (5) A_{12}–A_{13}, (6) A_{14}–A_{17}, (7) A_{18}–A_{21} and (8) A_{22}–A_{25}.

Maximum displacement limitations of ±0.35 in (8.89 mm) are imposed on every node in every direction and the axial stress constraints vary for each group as shown in Table 7.10. The range of the cross-sectional areas varies from 0.01 to 3.4 in^2 (0.6452 to 21.94 cm^2).

By the use of the proposed algorithm, this optimization problem is solved and Table 7.11 shows the obtained optimal design of CBO, which is compared with GA [21], PSO [22], HS [6] and RO [19]. The best weight of the CBO is 544.310 lb, which is slightly improved compared to other algorithms. It is evident from

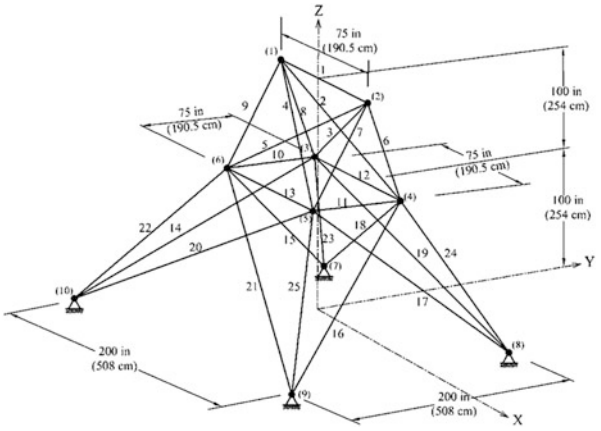

Fig. 7.12 Schematic of a 25-bar spatial truss

Table 7.9 Loading conditions for the 25-bar spatial truss

	Case 1			Case 2		
Node	P_X kips (kN)	P_Y kips (kN)	P_Z kips (kN)	P_X kips (kN)	P_Y kips (kN)	P_Z kips (kN)
1	0.0	20.0 (89)	−5.0 (22.5)	1.0 (4.45)	10.0 (44.5)	−5.0 (22.5)
2	0.0	−20.0 (89)	−5.0 (22.5)	0.0	10.0 (44.5)	−5.0 (22.5)
3	0.0	0.0	0.0	0.5 (22.5)	0.0	0.0
4	0.0	0.0	0.0	0.5 (22.5)	0.0	0.0

Table 7.10 Member stress limitations for the 25-bar spatial truss

Element group	Compressive stress limitations ksi (MPa)	Tensile stress limitation ksi (MPa)
1	35.092 (241.96)	40.0 (275.80)
2	11.590 (79.913)	40.0 (275.80)
3	17.305 (119.31)	40.0 (275.80)
4	35.092 (241.96)	40.0 (275.80)
5	35.092 (241.96)	40.0 (275.80)
6	6.759 (46.603)	40.0 (275.80)
7	6.959 (47.982)	40.0 (275.80)
8	11.082 (76.410)	40.0 (275.80)

Table 7.11 that the number of analysis and standard deviation of 20 independent runs for the CBO are 9,090 and 0.294 lb, respectively, which are much less than the other optimization algorithms. Figure 7.13 provides the convergence diagram of the CBO in 400 iterations.

7.3.2.2 A 72-Bar Spatial Truss Structure

Schematic topology and element numbering of a 72-bar space truss is shown in Fig. 7.14. The elements are classified in 16 design groups according to Table 7.12.

Table 7.11 Comparison of CBO optimized designs with literature in the 25-bar spatial truss

		Optimal cross-sectional areas (in^2)				
Element group		Rajeev et al. GA [21]	Schutte et al. PSO[22]	Lee et al. HS [18]	Kaveh et al. RO [19]	Present work [2]
1	A_1	0.10	0.010	0.047	0.0157	0.0100
2	A_2–A_5	1.80	2.121	2.022	2.0217	2.1297
3	A_6–A_9	2.30	2.893	2.95	2.9319	2.8865
4	A_{10}–A_{11}	0.20	0.010	0.010	0.0102	0.0100
5	A_{12}–A_{13}	0.10	0.010	0.014	0.0109	0.0100
6	A_{14}–A_{17}	0.80	0.671	0.688	0.6563	0.6792
7	A_{18}–A_{21}	1.80	1.611	1.657	1.6793	1.6077
8	A_{22}–A_{25}	3.0	2.717	2.663	2.7163	2.6927
Best weight (Ib)		546	545.21	544.38	544.656	544.310
Average weight (Ib)		N/A	546.84	N/A	546.689	545.256
Std dev		N/A	1.478	N/A	1.612	0.294
No. of analyses		N/A	9,596	15,000	13,880	9,090

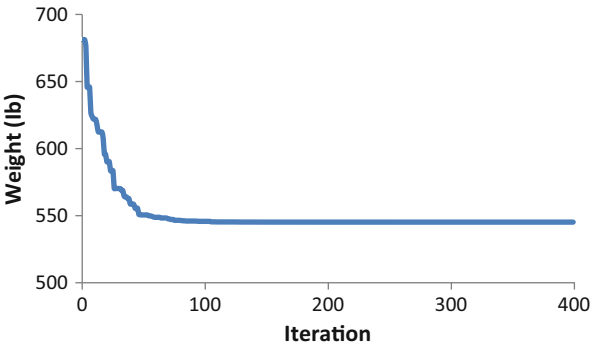

Fig. 7.13 The convergence diagram for the 25-bar spatial truss [2]

The material density is 0.1 lb/in^3 (2,767.990 kg/m^3) and the modulus of elasticity is taken as 10,000 ksi (68,950 MPa). The members are subjected to the stress limits of ±25 ksi (±172.375 MPa). The uppermost nodes are subjected to the displacement limits of ±0.25 in (±0.635 cm) in both x and y directions. The minimum permitted cross-sectional area of each member is taken as 0.10 in^2 (0.6452 cm^2), and the maximum cross-sectional area of each member is 4.00 in^2 (25.81 cm^2). The loading conditions are considered as:

1. Loads 5, 5 and −5 kips in the *x, y and z* directions at node 17, respectively;
2. A load −5 kips in the *z* direction at nodes 17, 18, 19 and 20;

Table 7.12 summarizes the results obtained by the present work and those of the previously reported researches. The best result of the CBO approach is 379.694, while it is 385.76, 380.24, 381.91, 379.85 and 380.458 Ib for the GA [23], ACO [24], PSO [25], BB–BC [26] and RO [19] algorithm, respectively. Also, the number of analyses of the CBO is 15,600, while it is 18,500, 19,621 and 19,084 for the

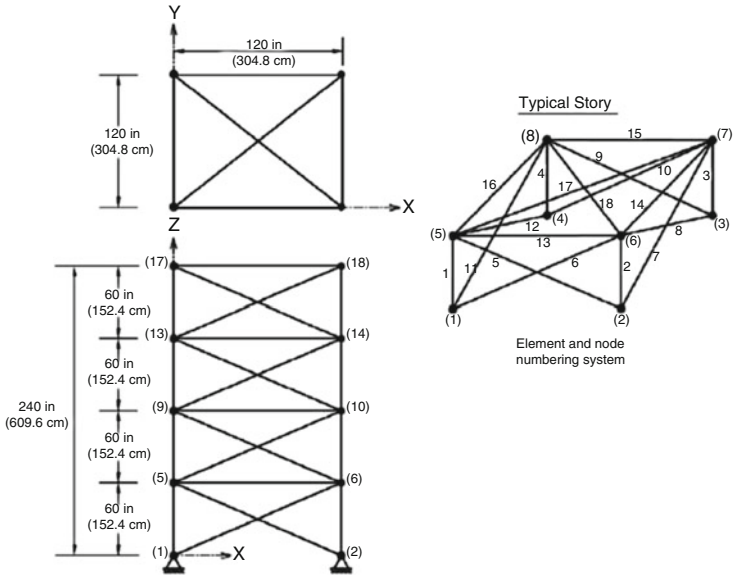

Fig. 7.14 Schematic of a seventy-two bar spatial truss

ACO, BB–BC and RO algorithm, respectively. Also, it is evident from Table 7.12 that the standard deviation of 20 independent runs for the CBO is less than the other optimization algorithms. Figure 7.15 shows the convergence diagrams in terms of the number of iterations for this example. Figure 7.16 shows the allowable and existing stress values in truss member using the CBO.

7.3.2.3 A 582-Bar Tower Truss

The 582-bar spatial truss structure, shown in Fig. 7.17, was studied with discrete variables by other researchers [27, 28]. However, here we have used this structure with continuous sizing variables. The 582 structural members categorized as 32 independent size variables. A single load case is considered consisting of lateral loads of 5.0 kN (1.12 kips) applied in both x- and y-directions and a vertical load of −30 kN (−6.74 kips) applied in the z-direction at all nodes of the tower. The lower and upper bounds on size variables are taken as 3.1 in^2 (20 cm^2) and 155.0 in^2 (1,000 cm^2), respectively.

The allowable tensile and compressive stresses are used as specified by the AISC ASD [20] code, as Eqs. (7.28) and (7.29).

The maximum slenderness ratio is limited to 300 for tension members, and it is recommended to be limited to 200 for compression members according to ASD-AISC [20]. The modulus of elasticity is 29,000 ksi (203,893.6 MPa) and the yield stress of steel is taken as 36 ksi (253.1 MPa). Other constraints are the

Table 7.12 Comparison of CBO optimized designs with literature in the 72-bar spatial truss (in^2)

	Optimal cross-sectional areas (in^2)					
Element group	Erbatur et al. GA [23]	Camp et al. ACO [24]	Perez et al. PSO [25]	Camp BB–BC [26]	Kaveh et al. RO [19]	Present work [2]
1–4	1.755	1.948	1.7427	1.8577	1.8365	1.9028
5–12	00.505	0.508	0.5185	0.5059	0.5021	0.5180
13–16	0.105	0.101	0.1000	0.1000	0.1000	0.1001
17–18	0.155	0.102	0.1000	0.1000	0.1004	0.1003
19–22	1.155	1.303	1.3079	1.2476	1.2522	1.2787
23–30	0.585	0.511	0.5193	0.5269	0.5033	0.5074
31–34	0.100	0.101	0.1000	0.1000	0.1002	0.1003
35–36	0.100	0.100	0.1000	0.1012	0.1001	0.1003
37–40	0.460	0.561	0.5142	0.5209	0.5730	0.5240
41–48	0.530	0.492	0.5464	0.5172	0.5499	0.5150
49–52	0.120	0.1	0.1000	0.1004	0.1004	0.1002
53–54	0.165	0.107	0.1095	0.1005	0.1001	0.1015
55–58	0.155	0.156	0.1615	0.1565	0.1576	0.1564
59–66	0.535	0.550	0.5092	0.5507	0.5222	0.5494
67–70	0.480	0.390	0.4967	0.3922	0.4356	0.4029
71–72	0.520	0.592	0.5619	0.5922	0.5971	0.5504
Best weight (Ib)	385.76	380.24	381.91	379.85	380.458	379.6943
Average weight (Ib)	N/A	383.16	N/A	382.08	382.553	379.8961
Std dev	N/A	3.66	N/A	1.912	1.221	0.0791
No. of analyses	N/A	18,500	N/A	19,621	19,084	15,600

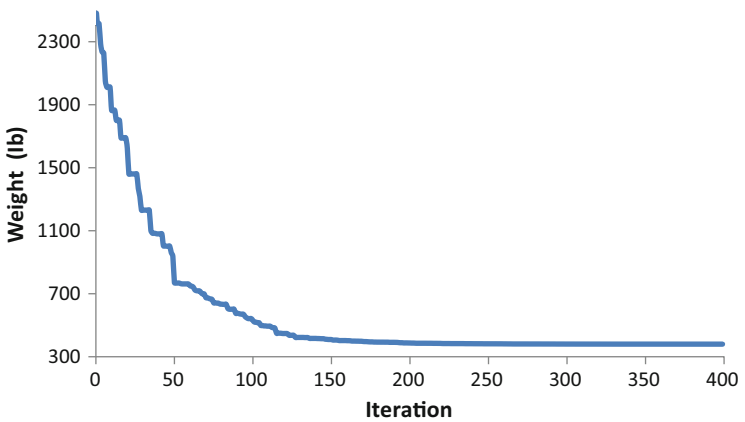

Fig. 7.15 The convergence diagram of the CBO algorithm for the 72-bar spatial truss [2]

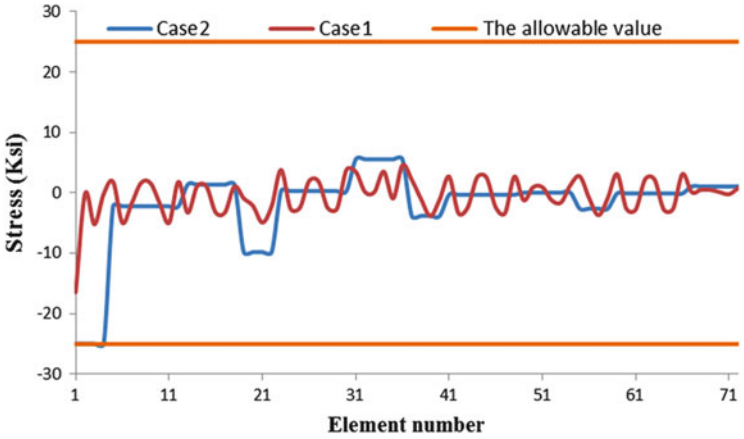

Fig. 7.16 Comparison of the allowable and existing stresses in the elements of the 72-bar truss structure [2]

limitations of nodal displacements which should be no more than 8.0 cm (3.15 in) in all directions.

Table 7.13 lists the optimal values of the 32 size variables obtained by the present algorithm. Figure 7.18 shows the convergence diagrams for the utilized algorithms. Figure 7.19 shows the allowable and existing stress ratio and displacement values of the CBO. Here, the number of structural analyses is taken as 20,000. The maximum values of displacements in the x-, y- and z-directions are 8 cm, 7.61 cm and 2.15 cm, respectively. The maximum stress ratio is 0.47 %.

7.3.2.4 A 52-Bar Dome-Like Truss

Figure 7.20 shows the initial topology and the element numbering of a 52-bar dome-like space truss. This example has been investigated by Lingyun et al. [29], Gomes [30] utilized the NHGA and PSO algorithms. This has also been investigated by Kaveh and Zolghadr [31] using the standard CSS. This example is optimized for shape and configuration. The space truss has 52 bars, and non-structural masses of $m = 50$ kg are added to the free nodes. The material density is 7,800 kg/m^3 and the modulus of elasticity is 210, 000MPa. The structural members of this truss are categorized into eight groups, where all members in a group share the same material and cross-sectional properties. Table 7.14 shows each element group by member numbers. The range of the cross-sectional areas varies from 1 to 10 cm^2. The shape optimization is performed taking into account that the symmetry is preserved in the process of design. Each movable node is allowed to vary ± 2 m. There are two constraints in the first two natural frequencies so that $\omega_1 \leq 15.916$ HZ and $\omega_2 \geq 28.648$ HZ. This example is considered to be a

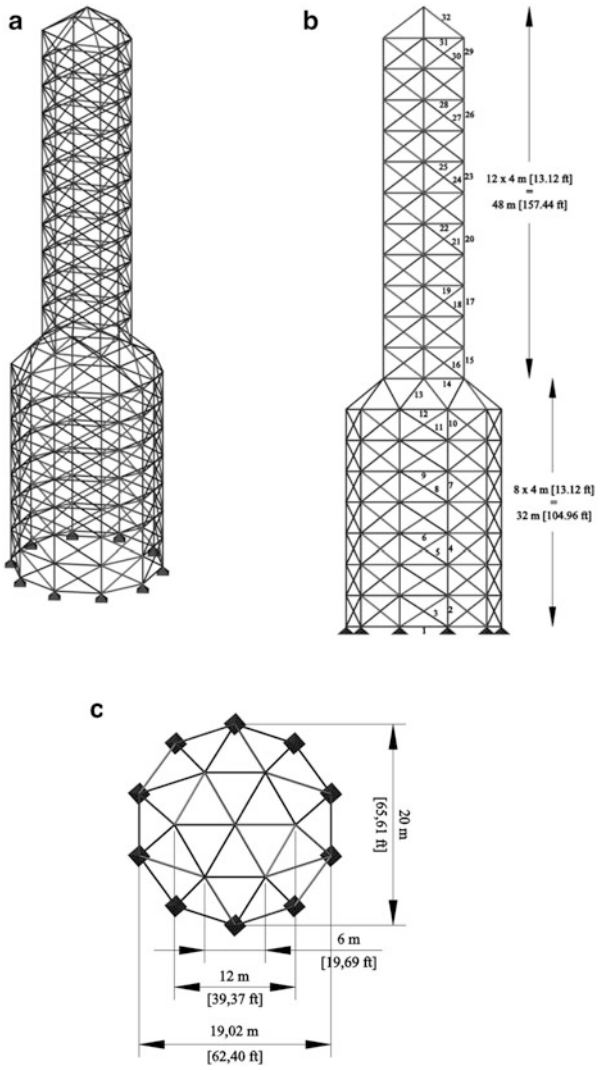

Fig. 7.17 Schematic of a 582-bar tower truss. (**a**) 3D view, (**b**) side view, and (**c**) top view

truss optimization problem with two natural frequency constraints and 13 design variables (five shape variables plus eight size variables).

Table 7.15 compares the cross section, best weight, mean weight and standard deviation of 20 independent runs of CBO with the results of other researches. It is evident that the CBO is better than in term of best weight of the results. Table 7.16 shows the natural frequencies of optimized structure obtained by different authors in the literature and the results obtained by the present algorithm. Figure 7.21 provides the convergence rates of the best result founded by the CBO.

Table 7.13 Optimum design cross-sections for the 582-bar tower truss

	Present work [2]		Present work [2]
Element groups	Area (cm^2)	Element groups	Area (cm^2)
1	20.5526	17	155.6601
2	162.7709	18	21.4951
3	24.8562	19	25.1163
4	122.7462	20	94.0228
5	21.6756	21	20.8041
6	21.4751	22	21.223
7	110.8568	23	53.5946
8	20.9355	24	20.628
9	23.1792	25	21.5057
10	109.6085	26	26.2735
11	21.2932	27	20.6069
12	156.2254	28	21.5076
13	159.3948	29	24.1394
14	107.3678	30	20.2735
15	171.915	31	21.1888
16	31.5471	32	29.6669
	Volume (m^3)		16.1520

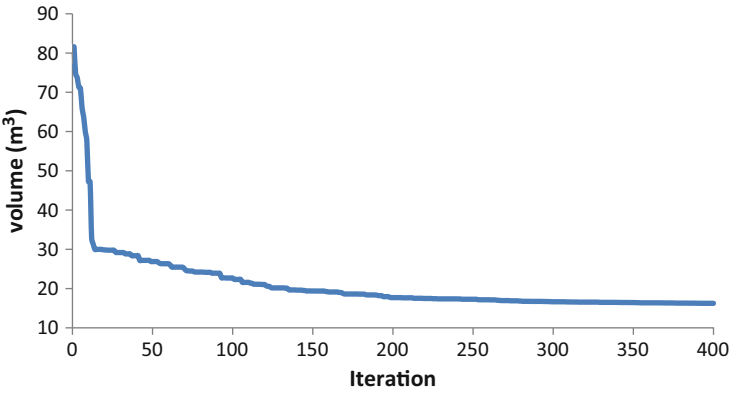

Fig. 7.18 The convergence diagram of the CBO for 582-bar tower truss [2]

7.3.2.5 The Model of Burro Creek Bridge

The last example is the sizing optimization of the planar bridge shown in Fig. 7.22a. This example has been first investigated by Makiabadi et al. [32] using the teaching-learning-based optimization algorithm. This bridge is 680 ft long and 155 ft high truss of the main span. Also, both upper and lower chords shapes are quadratic parabola. Because of symmetry of this truss, one can analysis half of the structure, Fig. 7.22b. The element groups and applied equivalent centralized loads are shown in Fig. 7.22b. The modulus of elasticity of material is 4.2×10^9 lb/ft^2, Fy is taken

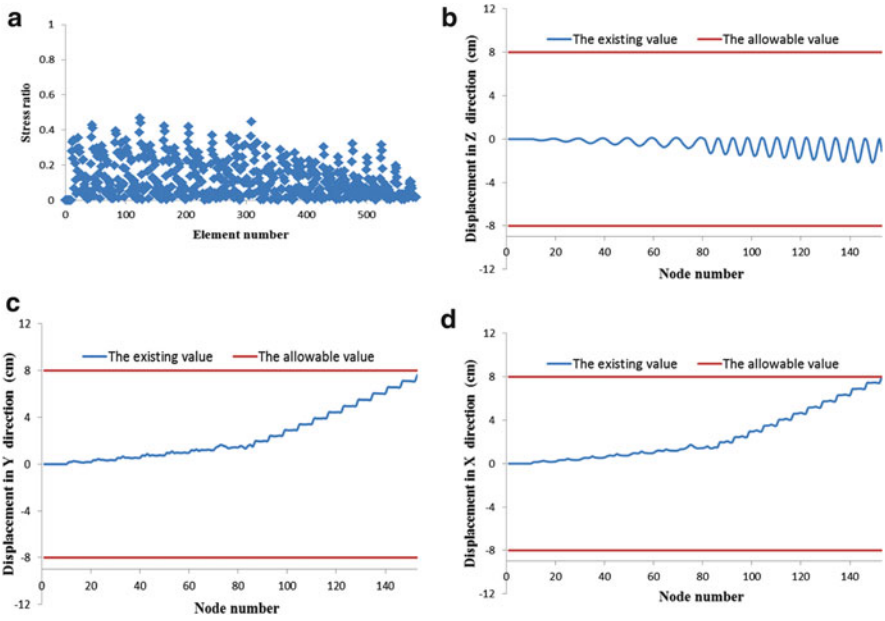

Fig. 7.19 Comparison of the allowable and existing constraints for the 582-bar truss using the DHPSACO [2]. (**a**) Stress ratio, (**b**) displacement in the z- direction, (**c**) displacement in the y-direction, (**d**) displacement in the x-direction

as 72.0×10^5 lb/ft^2 and the density of material is 495 lb/ft^3. For this example, allowable tensile and compressive stresses are considered according to AISC ASD (1989) [20]. According to Australian Bridge Code [33], the allowable displacement is 0.85 ft.

Three design cases are studied according to three different groups of variables including 4, 8 and 12 variables in the design. For three cases, the size variables are chosen from 0.2 in^2 to 5.0 in^2. Table 7.17 shows the full list of three different groups of variables used in the problem.

Table 7.18 compares the results obtained of the CBO with those of the TLS algorithm. The optimum weight of the CBO are 299,756.7, 269,839.5 and 253,871.3 Ib, while these are 368,598.1, 315,885.7, and 298,699.9 for Case I, II and III, respectively. It can be seen that the number of analyses is much less than that of TLS algorithm. Figure 7.23 provides a comparison of the convergence diagrams of the CBO for three cases.

7.3.3 Discussion

CBO utilizes simple formulation to find minimum of functions and does not depend on any internal parameter. Also, the formulation of CBO algorithm does not use the

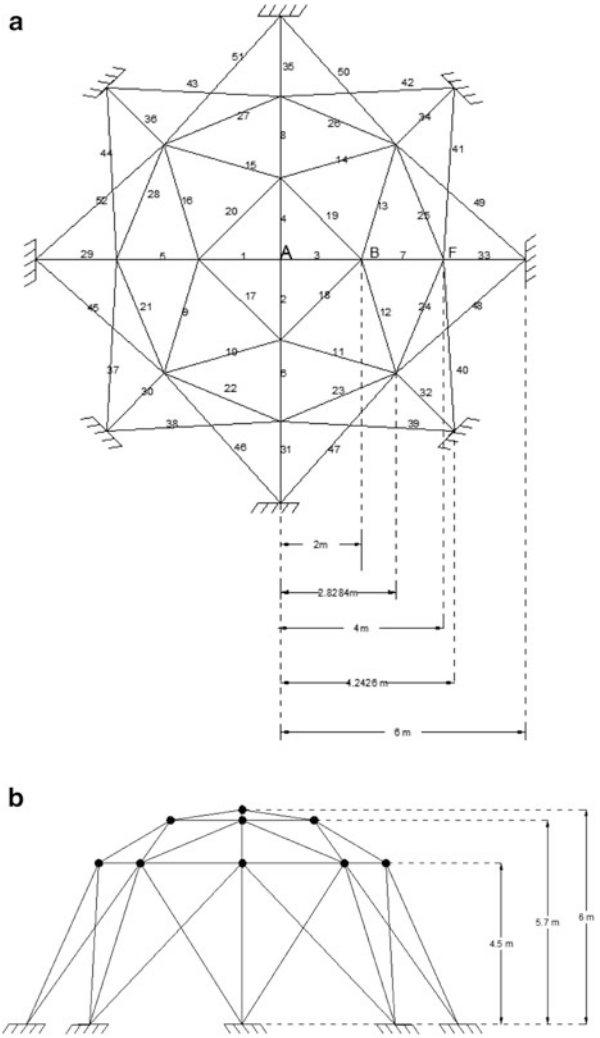

Fig. 7.20 Schematic of the 52-bar space truss. (**a**) Top view and (**b**) side view

Table 7.14 Element grouping

Group number	Elements
1	1–4
2	5–8
3	9–16
4	17–20
5	21–28
6	29–36
7	37–44
8	45–52

Table 7.15 Cross-sectional areas and nodal coordinates obtained by different researchers for the 52-bar space truss

Variable	Initial	Lingyun et al. GA [29]	Gomes PSO [30]	Kaveh et al. CSS [31]	Present work [2]
Z_A (m)	6.000	5.8851	5.5344	5.2716	5.6523
X_B (m)	2.000	1.7623	2.0885	1.5909	1.9665
Z_B (m)	5.700	4.4091	3.9283	3.7039	3.7378
X_F (m)	4.000	3.4406	4.0255	3.5595	3.7620
Z_F (m)	4.500	3.1874	2.4575	2.5757	2.5741
A_1 (cm^2)	2.0	1.0000	0.3696	1.0464	1.0009
A_2 (cm^2)	2.0	2.1417	4.1912	1.7295	1.3326
A_3 (cm^2)	2.0	1.4858	1.5123	1.6507	1.3751
A_4 (cm^2)	2.0	1.4018	1.5620	1.5059	1.6327
A_5 (cm^2)	2.0	1.9110	1.9154	1.7210	1.5521
A_6 (cm^2)	2.0	1.0109	1.1315	1.0020	1.0000
A_7 (cm^2)	2.0	1.4693	1.8233	1.7415	1.6071
A_8 (cm^2)	2.0	2.1411	1.0904	1.2555	1.3354
Best weight (kg)	338.69	236.046	228.381	205.237	197.962
Average weight (kg)	–	–	234.3	213.101	206.858
Std dev	–	–	5.22	7.391	5.750
No. of analyses	–	–	11,270	4,000	4,000

Table 7.16 Natural frequencies (HZ) of the optimized 52-bar planar truss

Frequency number	Initial	Lingyun et al. GA [29]	Gomes PSO [30]	Kaveh et al. CSS [31]	Present work [2]
1	22.69	12.81	12.751	9.246	10.2404
2	25.17	28.65	28.649	28.648	28.6482
3	25.17	28.65	28.649	28.699	28.6504
4	31.52	29.54	28.803	28.735	28.7117
5	33.80	30.24	29.230	29.223	29.2045

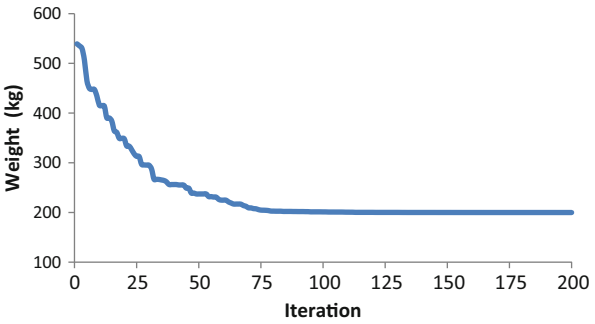

Fig. 7.21 Convergence history for the 52-bar truss [2]

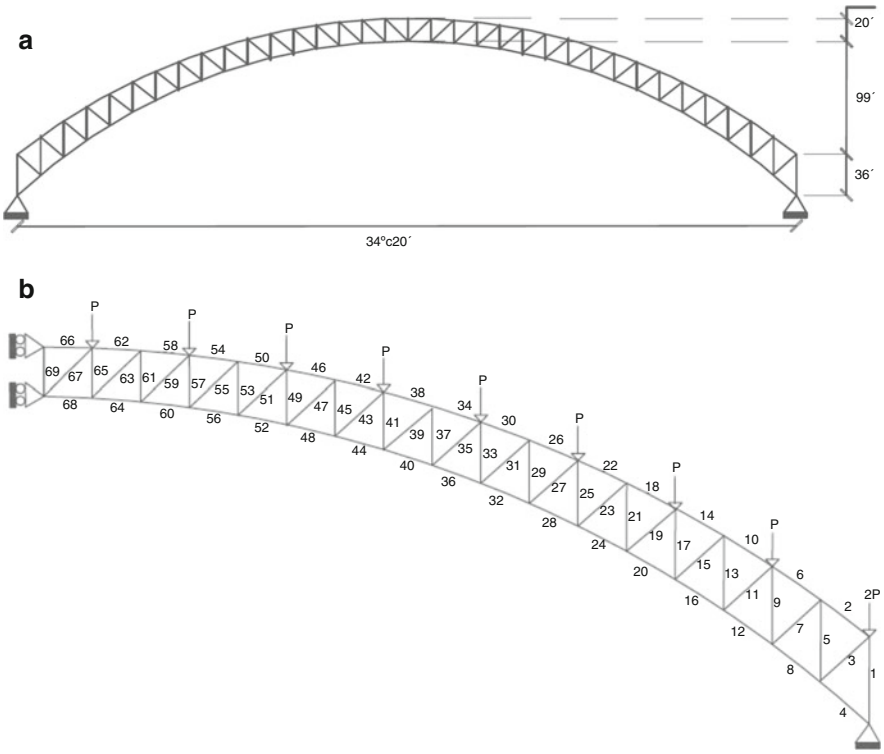

Fig. 7.22 (**a**) Schematic of the Burro Creek Bridge. (**b**) finite element nodel and element numbering of Burro Creek Bridge

memory for saving the best-so-far solution (i.e. the best position of agents from the previous iterations). By defining the coefficient of restitution (COR), a good balance between the global and local search is achieved in CBO. The proposed approach performs well in several test problems both in terms of the number of fitness function evaluations and in terms of the quality of the solutions. The results are compared to those generated with other techniques reported in the literature.

Table 7.17 Three different design variables for the Burro Creek Bridge

	Member number		
Design variables	Case I (4 variables)	Case II (8 variables)	Case III (12 variables)
1	67,63,59,55,51,47,43,39,35,31,27,23,19,15,11,7,3	67,63,59,55,51,47,43,39,35,31,27	67,63,59,55,51,47,43
2	66,62,58,54,50,46,42,38,34,30,26,22,18,14,10,6,2	66,62,58,54,50,46,42,38,34,,30,26	66,62,58,54,50,46,42
3	69,65,61,57,53,49,45,41,37,33,29,25,21,17,13,9,5,1	69,65,61,57,53,49,45,41,37,33,29	69,65,61,57,53,49,45
4	68,64,60,56,52,48,44,40,36,32,28,24,20,16,12,8,4	68,64,60,56,52,48,44,40,36,32,28	68,64,60,56,52,48,44
5		23,19,15,11,7,3	39,35,31,27,23,19
6		22,18,14,10,6,2	38,34,30,26,22,18
7		25,21,17,13,9,5,1	41,37,33,29,25,21
8		24,20,16,12,8,4	40,36,32,28,24,20
9			15,11,7,3
10			14,10,6,2
11			17,13,9,5,1
12			16,12,8,4

Table 7.18 Comparison of CBO optimized cross sectional areas (in^2) with those of TLS for the Burro Creek Bridge

	Maktobi et al. TLS [32]			Present work [2]		
Design variables	Case I	Case II	Case III	Case I	Case II	Case III
1	0.20000	0.2000	0.20000	0.20000	0.20000	0.20010
2	0.39202	0.46247	0.49843	0.35830	0.46532	0.43580
3	0.41654	0.22233	0.20000	0.20000	0.20007	0.20020
4	0.85487	0.57067	0.39476	0.78100	0.48657	0.32630
5		0.20012	0.20000		0.20000	0.20000
6		0.31227	0.42170		0.20004	0.27960
7		0.42791	0.25346		0.20001	0.20010
8		0.84160	0.63739		0.81310	0.70410
9			0.20000			0.20000
10			0.27992			0.20010
11			0.43354			0.20000
12			0.83483			0.74470
Best weight (Ib)	368,598.1	315,885.7	298,699.9	299,756.7	269,839.5	253,871.3
Number of analyses	15,000	35,000	50,000	8,000	8,000	8,000

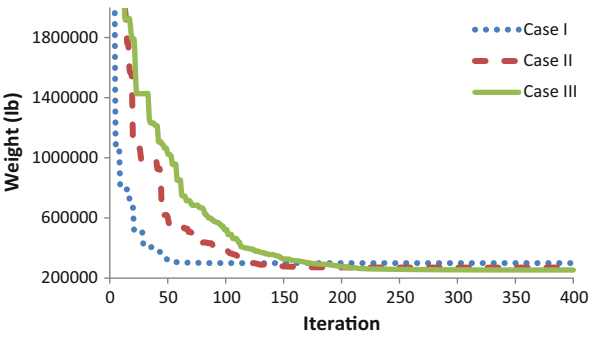

Fig. 7.23 Comparison of the convergence rates between three different cases for the Burro Creek Bridge [2]

References

1. Kaveh A, Mahdavi VR (2014) Colliding bodies optimization: a novel meta-heuristic method. Comput Struct (in press)
2. Kaveh A, Mahdavi VR (2014) Colliding bodies optimization method for optimum design of truss structures with continuous variables. Adv Eng Softw 70:1–12
3. Tolman RC (1979) The principles of statistical mechanics. Clarendon Press, Oxford, Reissued
4. Tsoulos IG (2008) Modifications of real code genetic algorithm for global optimization. Appl Math Comput 203:598–607
5. Coello CAC (2000) Use of a self-adaptive penalty approach for engineering optimization problems. Comput Ind 41:113–127
6. He Q, Wang L (2007) An effective co-evolutionary particle swarm optimization for constrained engineering design problem. Eng Appl Artif Intell 20:89–99

7. Montes EM, Coello CAC (2008) An empirical study about the usefulness of evolution strategies to solve constrained optimization problems. Int J Gen Syst 37:443–473
8. Kaveh A, Talatahari S (2010) A novel heuristic optimization method: charged system search. Acta Mech 213:267–289
9. Ragsdell KM, Phillips DT (1976) Optimal design of a class of welded structures using geometric programming. ASME J Eng Ind Ser B 98:1021–1025
10. Deb K (1991) Optimal design of a welded beam via genetic algorithms. AIAA J 29:2013–2015
11. Coello CAC, Montes EM (1992) Constraint-handling in genetic algorithms through the use of dominance-based tournament. IEEE Trans Reliab 41(4):576–582
12. Sandgren E (1988) Nonlinear integer and discrete programming in mechanical design. In: Proceedings of the ASME design technology conference, Kissimine, FL, pp 95–105
13. Kannan BK, Kramer SN (1994) An augmented Lagrange multiplier based method for mixed integer discrete continuous optimization and its applications to mechanical design. Trans ASME J Mech Des 116:318–320
14. Deb K, Gene AS (1997) A robust optimal design technique for mechanical component design. In: Michalewicz Z, Dasgupta D (eds) Evolutionary algorithms in engineering applications. Springer, Berlin, pp 497–514
15. Belegundu AD (1982) A study of mathematical programming methods for structural optimization. Ph.D. thesis, Department of Civil and Environmental Engineering, University of Iowa, Iowa City, IA
16. Arora JS (1989) Introduction to optimum design. McGraw-Hill, New York, NY
17. Soh CK, Yang J (1996) Fuzzy controlled genetic algorithm search for shape optimization. J Comput Civil Eng ASCE 10:143–150
18. Lee KS, Geem ZW (2004) A new structural optimization method based on the harmony search algorithm. Comput Struct 82:781–798
19. Kaveh A, Khayatazad M (2012) A novel meta-heuristic method: ray optimization. Comput Struct 112–113:283–294
20. American Institute of Steel Construction (AISC) (1989) Manual of steel construction allowable stress design, 9th edn. American Institute of Steel Construction (AISC), Chicago, IL
21. Rajeev S, Krishnamoorthy CS (1992) Discrete optimization of structures using genetic algorithms. J Struct Eng ASCE 118:1233–1250
22. Schutte JJ, Groenwold AA (2003) Sizing design of truss structures using particle swarms. Struct Multidiscip Optim 25:261–269
23. Erbatur F, Hasançebi O, Tütüncü I, Kiliç H (2000) Optimal design of planar and space structures with genetic algorithms. Comput Struct 75:209–224
24. Camp CV, Bichon J (2004) Design of space trusses using ant colony optimization. J Struct Eng ASCE 130:741–751
25. Perez RE, Behdinan K (2007) Particle swarm approach for structural design optimization. Comput Struct 85:1579–1588
26. Camp CV (2007) Design of space trusses using Big Bang–Big Crunch optimization. J Struct Eng ASCE 133:999–1008
27. Kaveh A, Talatahari S (2009) A particle swarm ant colony optimization for truss structures with discrete variables. J Construct Steel Res 65:1558–1568
28. Hasançebi O, Çarbas S, Dogan E, Erdal F, Saka MP (2009) Performance evaluation of metaheuristic search techniques in the optimum design of real size pin jointed structures. Comput Struct 87:284–302
29. Lingyun W, Mei Z, Guangming W, Guang M (2005) Truss optimization on shape and sizing with frequency constraints based on genetic algorithm. J Comput Mech 25:361–368
30. Gomes MH (2011) Truss optimization with dynamic constraints using a particle swarm algorithm. Expert Syst Appl 38:957–968
31. Kaveh A, Zolghadr A (2011) Shape and size optimization of truss structures with frequency constraints using enhanced charged system search algorithm. Asian J Civil Eng 12:487–509

32. Makiabadi MH, Baghlani A, Rahnema H, Hadianfard MA (2013) Optimal design of truss bridges using teaching-learning-base optimization algorithm. Int J Optim Civil Eng 3 (3):499–510
33. AustRoads. 92 (1992) Austroads bridge design code. Australasian Railway Association, Surry Hills, NSW

Chapter 8
Ray Optimization Algorithm

8.1 Introduction

In this chapter a newly developed metaheuristic method, so-called Ray Optimization, is presented. Similar to other multi-agent methods, Ray Optimization has a number of particles consisting of the variables of the problem. These agents are considered as rays of light. Based on the Snell's light refraction law when light travels from a lighter medium to a darker medium, it refracts and its direction changes. This behavior helps the agents to explore the search space in early stages of the optimization process and to make them converge in the final stages. This law is the main tool of the Ray Optimization algorithm. This chapter consists of three parts.

In the first part Ray Optimization (RO) algorithm is developed and applied to some benchmark functions and engineering problems [1].

In the second part, RO is employed for size and shape optimization of truss structures. The goal function of the present optimization method is the minimization of truss weight under the required constraints [2].

In the third part, an improved ray optimization (IRO) algorithm is presented. This technique employs a new approach for generating solution vectors and modifies the procedure which returns the violated agents into the feasible search space. The IRO algorithm applied to some benchmark mathematical optimization problems and truss structure examples and its numerical results are compared to those of the standard RO and some well-known metaheuristic algorithms [3].

A. Kaveh, *Advances in Metaheuristic Algorithms for Optimal Design of Structures*,
DOI 10.1007/978-3-319-05549-7_8, © Springer International Publishing Switzerland 2014

8.2 Ray Optimization for Continuous Variables

The present method is inspired by transition of ray from one medium to another from physics. Here the transition of ray is utilized for finding the global or near-global solution. This algorithm is called Ray Optimization (RO) and uses the Snell's refraction law of the light.

8.2.1 Definitions and Concepts from Ray Theory

The contents of this section mainly follow the definitions and concepts presented in [4].

Certain transparent materials so-called dielectric, refract the light. As the light travels through these materials, its path is changed according to the Snell's refraction law. Each transparent material has an index of refraction. Showing the index of the refraction of the lighter material by n_d, and denoting the index of the refraction of the darker material by n_t, the Snell's law can be expressed as:

$$n_d . \sin(\theta) = n_t . \sin(\phi). \tag{8.1}$$

Where, θ and ϕ are the angles between the normal of two surfaces, $\boldsymbol{n}$, with incoming ray vector and the angle between the normal with the refracted ray vector, respectively, as shown in Fig. 8.1. Having the direction of incoming ray vector and the index of refraction of lighter and darker mediums, one can find the direction of refracted ray vector $\boldsymbol{t}$.

The computation for finding $\boldsymbol{t}$ is rather lengthy, but not difficult. To help us along the way, we use Fig. 8.1, where the vectors $\boldsymbol{d}$, $\boldsymbol{n}$ and $\boldsymbol{b}$ are unit vectors. This makes the formula slightly less complicated. The steps for finding $\boldsymbol{t}$ in a two dimensional space are outlined in the following:

1. We express $\boldsymbol{t}$ in terms of $\boldsymbol{n}$ and $\boldsymbol{b}$.

Let $\boldsymbol{t_n}$ be the component of $\boldsymbol{t}$ along $\boldsymbol{n}$ and $\boldsymbol{t_b}$ be the component of $\boldsymbol{t}$ along $\boldsymbol{b}$. Then, since $\boldsymbol{t}$ is a unit vector, we have

$$\cos(\phi) = \frac{\|\boldsymbol{t}_n\|}{\|\boldsymbol{t}\|} = \|\boldsymbol{t}_n\|. \tag{8.2}$$

Similarly,

$$\sin(\phi) = \|\boldsymbol{t}_b\|. \tag{8.3}$$

Now

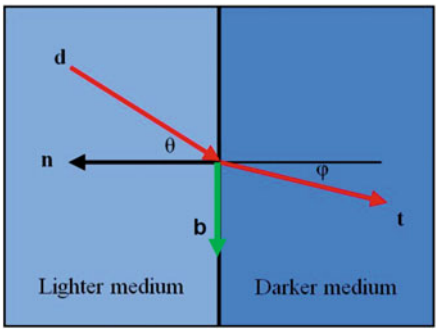

Fig. 8.1 Incident and refracted rays and their specifications [1]

$$\boldsymbol{t}_n = -\|\boldsymbol{t}_n\|.\boldsymbol{n}, \tag{8.4}$$

and

$$\boldsymbol{t}_b = \|\boldsymbol{t}_b\|.\boldsymbol{b}. \tag{8.5}$$

Therefore,

$$\boldsymbol{t} = -\cos(\phi).\boldsymbol{n} + \sin(\phi).\boldsymbol{b}. \tag{8.6}$$

2. Express $\boldsymbol{b}$ in terms of the known quantities using the fact that $\boldsymbol{b}$ is the unit length vector parallel to the projection of $\boldsymbol{d}$ onto the perpendicular to $\boldsymbol{n}$. Let $\boldsymbol{d_n}$ be the component of $\boldsymbol{d}$ along $\boldsymbol{n}$ and $\boldsymbol{d_b}$ be the component of $\boldsymbol{d}$ along $\boldsymbol{b}$, the direction perpendicular to $\boldsymbol{n}$. Then

$$\boldsymbol{d} = \boldsymbol{d}_n + \boldsymbol{d}_b. \tag{8.7}$$

Thus

$$\boldsymbol{d}_b = \boldsymbol{d} - \boldsymbol{d}_n = \boldsymbol{d} - (\boldsymbol{d}.\boldsymbol{n}).\boldsymbol{n}. \tag{8.8}$$

Also, since $\boldsymbol{d}$ is a unit vector, we have

$$\sin(\theta) = \frac{\|\boldsymbol{d}_b\|}{\|\boldsymbol{d}\|} = \|\boldsymbol{d}_b\|. \tag{8.9}$$

Thus

$$\boldsymbol{b} = \frac{\boldsymbol{d}_b}{\|\boldsymbol{d}_b\|} = \frac{\boldsymbol{d} - (\boldsymbol{d}.\boldsymbol{n}).\boldsymbol{n}}{\sin(\theta)}. \tag{8.10}$$

3. Using (8.1), one can express everything in terms of $\boldsymbol{n}$, $\boldsymbol{d}$, n_d and n_t.

Therefore

$$\begin{aligned} \boldsymbol{t} = -\cos(\phi).\boldsymbol{n} + \sin(\phi).\boldsymbol{b} = -\cos(\phi).\boldsymbol{n} + \sin(\phi).\frac{\boldsymbol{d} - (\boldsymbol{d}.\boldsymbol{n}).\boldsymbol{n}}{\sin(\theta)} = \\ -\cos(\phi).\boldsymbol{n} + \frac{n_d}{n_t}.(\boldsymbol{d} - (\boldsymbol{d}.\boldsymbol{n}).\boldsymbol{n}) \end{aligned} \tag{8.11}$$

Now we express cos(ϕ) in terms of known quantities. Using the identity, we have

$$\cos(\phi) = \sqrt{1 - \sin^2(\phi)}, \tag{8.12}$$

and employing Eq. (8.1), we obtain:

$$\cos(\phi) = \sqrt{1 - \frac{\mathrm{n}_d^2}{n_t^2}.\sin^2(\theta)}. \tag{8.13}$$

Finally, we have

$$\boldsymbol{t} = -\boldsymbol{n}.\sqrt{1 - \frac{\mathrm{n}_d^2}{n_t^2}.\sin^2(\theta)} + \frac{n_d}{n_t}.(\boldsymbol{d} - (\boldsymbol{d}.\boldsymbol{n}).\boldsymbol{n}) \tag{8.14}$$

Here, $\boldsymbol{t}$ is a normalized vector.

In tracing a ray in a 2-dimesional space, $\boldsymbol{d}$, $\boldsymbol{t}$, $\boldsymbol{n}$ were placed in z = 0 plane. In a 3-dimensional space, it is clear that one can pass a plane through two vectors like $\boldsymbol{n}$ and $\boldsymbol{d}$, which intersect each other at one point. Thus the ray tracing in 3-dimensional spaces is a special state of ray tracing in 2-dimensional spaces which occurs in a plane with an arbitrary orientation. Now if one can find two normalized vectors that are perpendicular to each other, like $\boldsymbol{i}^*$ and $\boldsymbol{j}^*$, then $\boldsymbol{n}$ and $\boldsymbol{d}$ can be rewritten in terms of these unit vectors. Finally, after finding $\boldsymbol{t}$ in the new coordinate system, it can be rearranged based on the primary coordinate system. One of these new unit components can be $\boldsymbol{n}$ as $\boldsymbol{i}^*$. The other one can be found by the following relationship:

$$\boldsymbol{n}.\boldsymbol{d} = \|\boldsymbol{n}\|.\|\boldsymbol{d}\|.\cos(\omega) = \cos(\omega) \tag{8.15}$$

Where ω is the angle between $\boldsymbol{n}$ and $\boldsymbol{d}$. Now if

$$\boldsymbol{n}.\boldsymbol{d} = 0 \tag{8.16}$$

Then $\omega = \pi/2$ and $\boldsymbol{d}$ will be $\boldsymbol{j}^*$, and if

$$0 < \boldsymbol{n}.\boldsymbol{d} \leq 1 \tag{8.17}$$

then the direction of $\boldsymbol{j}^*$ will be obtained by $\left(\boldsymbol{n} - \frac{\boldsymbol{d}}{\boldsymbol{n}.\boldsymbol{d}}\right)$, since

$$\boldsymbol{n}.\left(\boldsymbol{n} - \frac{\boldsymbol{d}}{\boldsymbol{n}.\boldsymbol{d}}\right) = \boldsymbol{n}.\boldsymbol{n} - \frac{\boldsymbol{n}.\boldsymbol{d}}{\boldsymbol{n}.\boldsymbol{d}} = 1 - 1 = 0 \tag{8.18}$$

And finally

$$\boldsymbol{j}^* = \frac{\left(\boldsymbol{n} - \frac{\boldsymbol{d}}{\boldsymbol{n}.\boldsymbol{d}}\right)}{norm\left(\boldsymbol{n} - \frac{\boldsymbol{d}}{\boldsymbol{n}.\boldsymbol{d}}\right)} \tag{8.19}$$

Where norm is function in MATLAB that provides the magnitude of a vector. Similarly if

$$-1 \leq \boldsymbol{n}.\boldsymbol{d} < 0 \tag{8.20}$$

$\boldsymbol{j}^*$ can be obtained by:

$$\boldsymbol{j}^* = \frac{\left(\boldsymbol{n} + \frac{\boldsymbol{d}}{-\boldsymbol{n}.\boldsymbol{d}}\right)}{norm\left(\boldsymbol{n} + \frac{\boldsymbol{d}}{-\boldsymbol{n}.\boldsymbol{d}}\right)}. \tag{8.21}$$

Now, in the new form, $\boldsymbol{d}$ and $\boldsymbol{n}$ are stated as:

$$\boldsymbol{n}^* = (1, 0), \tag{8.22}$$

And

$$\boldsymbol{d}^* = (\boldsymbol{d}.\boldsymbol{i}^*, \boldsymbol{d}.\boldsymbol{j}^*). \tag{8.23}$$

Therefore after calculating $\boldsymbol{t}^* = (t_1{}^*, t_2{}^*)$ in a two dimensional space, $\boldsymbol{t}$ in a three dimensional space is obtained as

$$\boldsymbol{t} = t_1^*.\boldsymbol{i}^* + t_2^*.\boldsymbol{j}^*. \tag{8.24}$$

It should be mentioned that for avoiding singularity in MATLAB, some changes are considered as shown in Table 8.1.

Table 8.1 The component of the new coordinate system

	$-0.05 \le \boldsymbol{n}.\boldsymbol{d} \le 0.05$	$0.05 \le \boldsymbol{n}.\boldsymbol{d} \le 1$	$-1 \le \boldsymbol{n}.\boldsymbol{d} \le -0.05$
i^*	n	n	n
j^*	d	$\boldsymbol{j}^* = \frac{\left(\boldsymbol{n}-\frac{\boldsymbol{d}}{\boldsymbol{n}.\boldsymbol{d}}\right)}{norm\left(\boldsymbol{n}-\frac{\boldsymbol{d}}{\boldsymbol{n}.\boldsymbol{d}}\right)}$	$\boldsymbol{j}^* = \frac{\left(\boldsymbol{n}+\frac{\boldsymbol{d}}{-\boldsymbol{n}.\boldsymbol{d}}\right)}{norm\left(\boldsymbol{n}+\frac{\boldsymbol{d}}{-\boldsymbol{n}.\boldsymbol{d}}\right)}$

8.2.2 Ray Optimization Method

8.2.2.1 Scattering and Evaluation Step

Like every other metaheuristic method, the RO has also a number of agents containing the variables of the design problem. As it was mentioned before, the ray tracing which is the main base of the RO was addressed in two and three dimensional spaces but it is necessary to introduce a procedure for performing the steps of the algorithm in higher dimensional spaces.

Suppose a solution vector has four design variables. In the first step, the goal function for this solution vector is determined. Now, this solution vector must be moved to a new position in the search space based on the RO algorithm. To achieve this, the solution vector is divided into two groups each group having two members and then the corresponding agents are moved to their new positions. In other word, we move the first group to the new position based on a 2D space and also the second group is moved to the new position in another 2D space. Now, we have a new solution vector with four variables which is ready for determining the goal function.

If a solution vector for another problem has seven variables, then we divide it into two groups of two variables and one group of three variables, and repeat the above procedure in two 2D spaces and one 3D space for the movement. After the movements, we join them together to have a unit solution vector. For any other number of variables the grouping into two and three elements can be performed. Therefore by using this approach, the problem of higher dimensions is dealt with and one can return to the first step of the algorithm. In this step, the agents must be scattered in the search space, and this requirement is provided by:

$$\mathrm{X}_{ij} = \mathrm{X}_{j,\min} + rand.\left(\mathrm{X}_{j,\max} - \mathrm{X}_{j,\min}\right) \tag{8.25}$$

Where X_{ij} is the jth variable of the ith agent. $X_{j,min}$ and $X_{j,max}$ are the minimum and maximum limits of the jth variable, and *rand* is a random number with its range being between 0 and 1. At the end of this step, after the evaluation of the goal function for each agent, the position of the best agent is saved as the global best and the position of each agent is saved as its local best.

8.2.2.2 Movement Vector and Motion Refinement Step

For each of the above mentioned agents, a groups of movement vectors should be assigned according to their division, and if the agent has a one 3-variable group and two 2-variable groups, it must have a one 3-variable movement vector group and two 2-variable movement vector groups, respectively. For the first movement, these vectors are obtained by:

$$\boldsymbol{v}_{ij} = -1 + 2.rand. \tag{8.26}$$

Where $\boldsymbol{v}_{ij}$ is the jth component of the ith agent and it may belong to a 2-variable or a 3-variable group. After finding the components of each 2 or 3-variable group, these must be converted to normalized vectors. The reason of this action will be presented in the subsequent steps. Now by adding the movement vector of each agent, they move to their new positions, but there is a possibility of boundary violation, so they must be refined. This objective is fulfilled by the following procedure:

If an agent violates a boundary, it intersects the boundary at a specified point, because of having definite movement vector. Now using this point and initial position of the agent, one can make a new vector whose direction is the same as the prior movement vector and its length is a multiple of the distance which is between the initial position and the boundary intersection. This multiple should naturally be less than 1. Since the best answer, especially in engineering problems, is close to the boundaries [5], it is locked on 0.9. During the implementation of this stage, it is found that when a movement vector is very close to a unit one-component vector such (1,0) and (0,1) for 2-dimensional spaces and (1,0,0) , (0,1,0) and (0,0,1) for 3-dimensional spaces, it causes singularity in the process of finding the intersection point at the boundary. For solving this problem, after normalizing the movement vector, if one component of this vector is equal or greater than 0.95, the other component(s) are not considered and upon this solitary component, the new movement vector is made.

After motion refinement and evaluation of the goal function, again the so-far best agent at this stage is selected as the global best and for each agent, the so-far best position by this stage, is selected as its local best.

8.2.2.3 Origin Making and Convergent Step

Now each agent must be moved to its new position, and first the point to which each particle moves must be determined. This point is named origin and it is specified by:

$$\boldsymbol{O}_i^k = \frac{(ite + k).\boldsymbol{GB} + (ite - k).\boldsymbol{LB_i}}{2.ite} \tag{8.27}$$

Where $\boldsymbol{O}_i^k$ is the origin of ith agent for the kth iteration, ite is the total number of

iteration for the optimization process, and $\boldsymbol{GB}$ and $\boldsymbol{LB}_i$ are the global best and local best of the ith agent, respectively. As Eq. (8.27) implies, at the beginning of iterations, the origin is approximately at the middle of the local best and global best. Thus exploration which is an important parameter in optimization is achieved. By progressing the iterations there is a balance between exploration and second important parameter, known as the exploitation. By passing the mid steps of iterations, the origin will be close to global best and the exploitation is empowered.

As it was mentioned before, the ray tracing is used for the movement and convergent step. Each ray of light has a normalized vector with which passes the medium. In the optimization, the movement vector is a positive multiple of this vector. When the ray comes to a new darker medium, the direction of its vector will be converted upon its angle between the initial direction and normal of surfaces of mediums and the ratio of refraction index. After refraction, the new direction will be closer to the normal than the initial direction, so one can say it converges to the normal. Therefore if in the process of optimization, the normal is selected as a vector whose origin is $\boldsymbol{O}$ and its end is the current position of agent, it should be anticipated that the agent will converge to $\boldsymbol{O}$. With refinement of $\boldsymbol{O}$ during the optimization, the agents approach to best result.

In order to show how the agents converge the to a point, consider an agent with arbitrary position and movement vector in the two dimensional space, like (4,5) and (0.707, −0.707), respectively. If this particle wants to move to a predefined point like the origin, it can be moved toward this point as illustrated in Fig. 8.2. It should be mentioned that, some restrictions are imposed for this convergence. These consist of a fixed ratio 0.6 as the index refraction and 200 times as the number of iterations. In order to show how the ratio of the index refraction affects the search procedure, see Fig. 8.3 where the above problem is solved with different values of the index refraction ratios. As can be seen, when the index refraction ratio is a number close to 1, the exploration is increased, but by decreasing this value the convergence occurs rapidly.

Now, the direction of the new movement vector is determined. According to (8.14), it is a normalized vector and it requires a logical coefficient. Thus the final form of the movement vector after finding the new direction is given by:

$$\boldsymbol{V}_{i,l} = \boldsymbol{V}_{i,l}^{'}.norm(\boldsymbol{X}_{i,l} - \boldsymbol{O}_{i,l}). \tag{8.28}$$

Where $\boldsymbol{V}_{i,l}^{'}$, $\boldsymbol{X}_{i,l}$, $\boldsymbol{O_{i,l}}$, and $\boldsymbol{V}_{i,l}$ are the normalized movement vector, current position of the agent, the origin, and refined movement vector of the ith agent, respectively, that belong to lth group.

In some cases it is possible that for an agent, $\boldsymbol{O_{i,l}}$ and its current position are the same, so the direction of normal cannot be obtained. This problem occurs when the agent is the so-far best one. Therefore, it is logical to permit it to move in the same direction because of finding a more adequate answer, but the length of this vector should be changed according to:

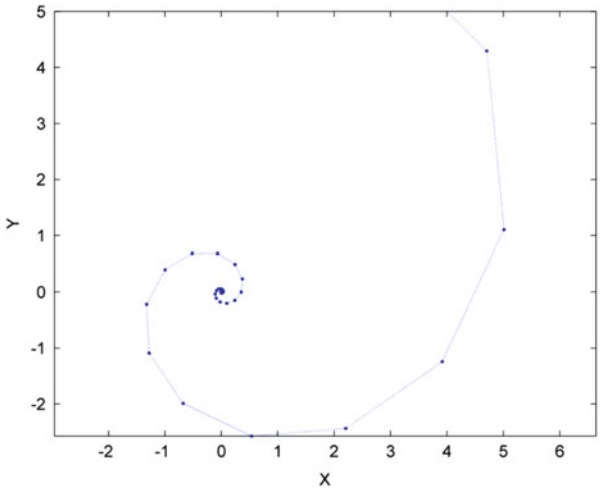

Fig. 8.2 The movement of an agent to the origin [1]

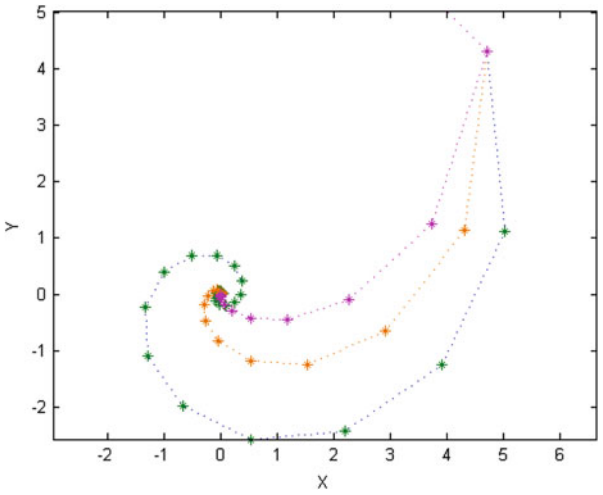

Fig. 8.3 Ratio of the index refraction of the magnet line, brown line and blue line are 0.5, 0.65 and 0.85, respectively [1]

$$\boldsymbol{V}_{i,l}{}^{k+1} = \frac{\boldsymbol{V}_{i,l}{}^{k}}{norm\left(\boldsymbol{V}_{i,l}{}^{k}\right)}.rand.0.001 \tag{8.29}$$

In this equation, $\boldsymbol{V}_{i,l}{}^{k}$ is the movement vector of the kth iteration that belongs to lth group of the ith agent, and $\boldsymbol{V}_{i,l}{}^{k+1}$ is the movement vector of the $(k+1)$th iteration. Also, for a fine and stochastic search, the initial normalized vector is multiplied by 0.001 and a random number between 0 and 1 is utilized.

One of the important features of each metaheuristic algorithm is that, it should have a stochastic nature to find the best answer. Here, this feature is added to the RO by adding a random change to the movement vector. In other word, there is a

possibility like ***stoch*** that specifies whether a movement vector must be changed or not. If this occurs, a new movement vector will be made considered as

$$\boldsymbol{V}_{\boldsymbol{ijl}}^{(\boldsymbol{k+1})} = -1 + 2.rand, \tag{8.30}$$

where $\boldsymbol{V}_{\mathrm{ijl}}^{(\mathrm{k}+1)}$ is the *j*th component of the *l*th group that belongs to the *i*th agent in *(k + 1)*th iteration. However, the length of this vector should be refined. Therefore the following relationship is considered:

$$\boldsymbol{V}_{il}{}^{k+1} = \frac{\boldsymbol{V}_{il}{}^{(k+1)'}}{norm\left(\boldsymbol{V}_{il}{}^{(k+1)'}\right)} \cdot \frac{a}{d}.rand \tag{8.31}$$

Here, *a* is calculated by the following relationship:

$$a = \sqrt{\sum_{i=1}^{n} \left(\boldsymbol{X}_{i,\max} - \boldsymbol{X}_{i,\min}\right)^2} \qquad n = \left\{ \begin{array}{ll} 2 & \text{for two variable groups} \\ 3 & \text{for three variable groups} \end{array} \right\} \tag{8.32}$$

Where $\boldsymbol{X}_{\boldsymbol{i},max}$ and $\boldsymbol{X}_{i,min}$ are the maximum and minimum limits of variables that belongs to *i*th component of the movement vector, and *d* is a number that divides *a* into smaller segements for effective search. It is found that if *d* and *stoch* are chosen as 7.5 and 0.35, the best result for optimization process will be obtained.

8.2.2.4 Finish or Redoing Step

By approaching to a pre-specific criterion, the process of optimization ceases. Some criteria are considered as

- Maximum number of iteration: by approaching to a predefined number of iteration, the process of optimization will be terminated.
- Number of ineffective iteration: if by passing a predefined number of iteration, there is no improvement in the goal function, the process of optimization will be ceased.
- Approaching to a minimum goal function error: sometimes the best answer of a goal function is specified, like mathematical benchmarks, so if an answer is found with a predefined error compared to real answer, the process of optimization will be terminated.

However if one of these criteria is not fulfilled, the process of optimization will continue and with new movement vector, the agents will move to their new positions. This cycle is continued until a predefined criterion is fulfilled.

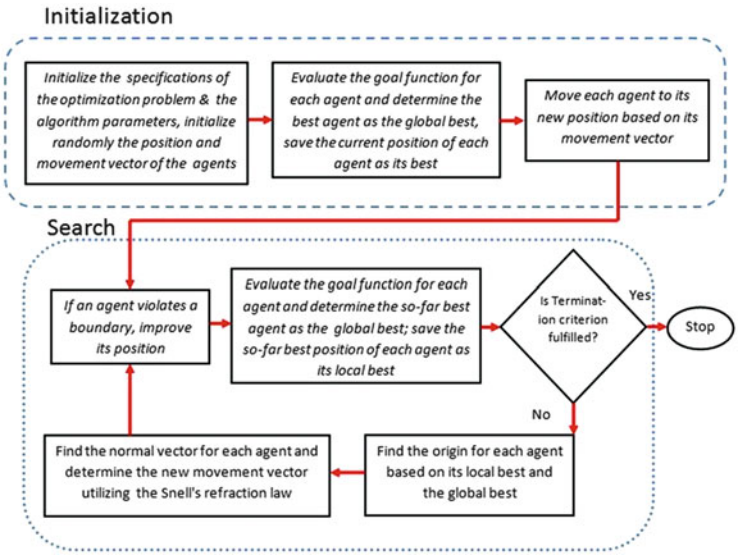

Fig. 8.4 The flowchart of the RO [1]

The flowchart of the optimization is illustrated in Fig. 8.4. The movement of a typical agent is shown in Fig. 8.5. As can be seen, in the origin making and convergent step, because of similar global best position and local best position that belong to agent 1, this agent moves in the same direction but with a smaller length. The movement vector for the agent 2 is determined based on the light refraction law. Finally, for showing the stochastic behavior of RO, agent 3 moves in a random direction with a controlled length.

8.2.3 Validation of the Ray Optimization

In order to validate the efficiency of the RO, some mathematical examples are considered from literature. These examples are investigated in Sect. 8.2.3.1. For further investigation, in Sect. 8.2.3.2, some well-studied engineering design problems are also studied from the literature.

8.2.3.1 Mathematical Optimization Problems

In order to verifying the efficiency of the RO algorithm, some benchmark functions chosen from [6] are optimized by this algorithm, Table 8.2. A perspective view and the corresponding contour lines for these benchmarks are depicted in Figs. 8.6, 8.7, 8.8, and 8.9, where the number of variables is taken as 2. Like any other

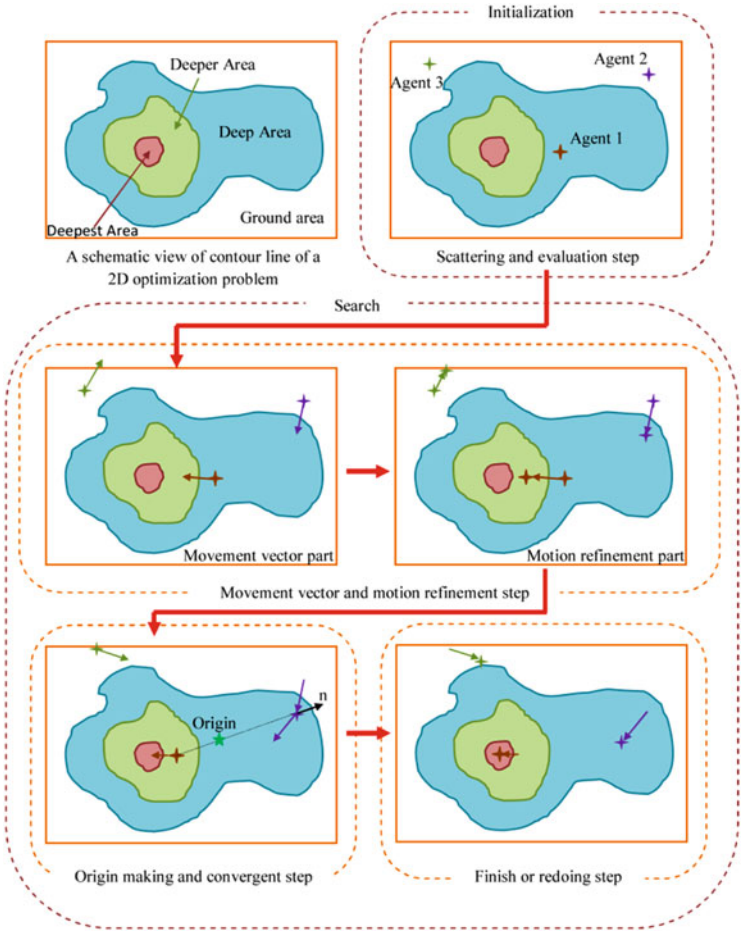

Fig. 8.5 Schematic view of the agent movement [1]

metaheuristics, considering a large number of agent causes vigorous search, but it increases the time and cost of the optimization. Also considering a small number of agents leads to a weak search. Therefore, for optimizing the mathematical problems, the number of agents is taken as 20. However it should mentioned we used 100 agents for the EXP16, SHEKEL5, SHEKEL7, SHEKEL10. Table 8.3 shows the result of the optimization by the present approach. In this table, the numbers in columns specify the average number of function evaluations. In this part for providing the stochastic behavior of metaheuristic algorithms, the number of independent runs is chosen as 50. The numbers in the parentheses represent the ratio of successful runs in which the algorithm has found the global minimum with a predefined accuracy, which is taken as $\varepsilon = f_{min} - f_{final} = 10^{-4}$. The absence of the parentheses shows that the algorithm has been successful in all independent runs.

Table 8.2 Specifications of the benchmark problems

Function name	Interval	Function	Global minimum
Aluffi-Pentiny	$\boldsymbol{X} \in [-10,10]^2$	$f(\boldsymbol{X}) = \frac{1}{4}x_1^4 - \frac{1}{2}x_1^2 + \frac{1}{10}x_1 + \frac{1}{4}x_2^2$	−0.352386
Bohachevsky 1	$\boldsymbol{X} \in [-100,100]^2$	$f(\boldsymbol{X}) = x_1^2 + 2x_2^2 - \frac{3}{10}\cos(3\pi x_1) - \frac{4}{10}\cos(4\pi x_2) + \frac{7}{10}$	0.0
Bohachevsky 2	$\boldsymbol{X} \in [-50,50]^2$	$f(\boldsymbol{X}) = x_1^2 + 2x_2^2 - \frac{3}{10}\cos(3\pi x_1)\cos(4\pi x_2) + \frac{3}{10}$	0.0
Becker and lago	$\boldsymbol{X} \in [-10,10]^2$	$f(\boldsymbol{X}) = (\lvert x_1\rvert - 5)^2 + (\lvert x_2\rvert - 5)^2$	0.0
Branin	$0 \le x_2 \le 15\ -5 \le x_1 \le 10$	$f(\boldsymbol{X}) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos(x_1) + 10$	0.397887
Camel	$\boldsymbol{X} \in [-5,5]^2$	$f(\boldsymbol{X}) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	−1.0316
Cb3	$\boldsymbol{X} \in [-5,5]^2$	$f(\boldsymbol{X}) = 2x_1^2 - 1.05x_1^4 + \frac{1}{6}x_1^6 + x_1x_2 + x_2^2$	0.0
Cosine mixture	$n = 4, \boldsymbol{X} \in [-1,1]^n$	$f(\boldsymbol{X}) = \sum_{i=1}^{n} x_i^2 - \frac{1}{10}\sum_{i=1}^{n}\cos(5\pi x_i)$	−0.4
De Joung	$\boldsymbol{X} \in [-5.12,5.12]^2$	$f(\boldsymbol{X}) = x_1^2 + x_2^2 + x_3^2$	0.0
Exponential	$n = 2,4,8, \boldsymbol{X} \in [-1,1]^n$	$f(\boldsymbol{X}) = -\exp\left(-0.5\sum_{i=1}^{n} x_i^2\right)$	−1.0
Goldstein And price	$\boldsymbol{X} \in [-2,2]^2$	$f(\boldsymbol{X}) = \left[1 + (x_1 + x_2 + 1)^2\right.$ $\left.(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)\right]$ $\left[30 + (2x_1 - 3x_2)^2(18 - 32x_1 - 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)\right]$	3.0
Griewank	$\boldsymbol{X} \in [-100,100]^2$	$f(\boldsymbol{X}) = 1 + \frac{1}{200}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{2}\cos\left(\frac{x_i}{\sqrt{i}}\right)$	0.0
Rastrigin	$\boldsymbol{X} \in [-1,1]^2$	$f(\boldsymbol{X}) = \sum_{i=1}^{n}\left(x_i^2 - \cos(18x_i)\right)$	−2.0

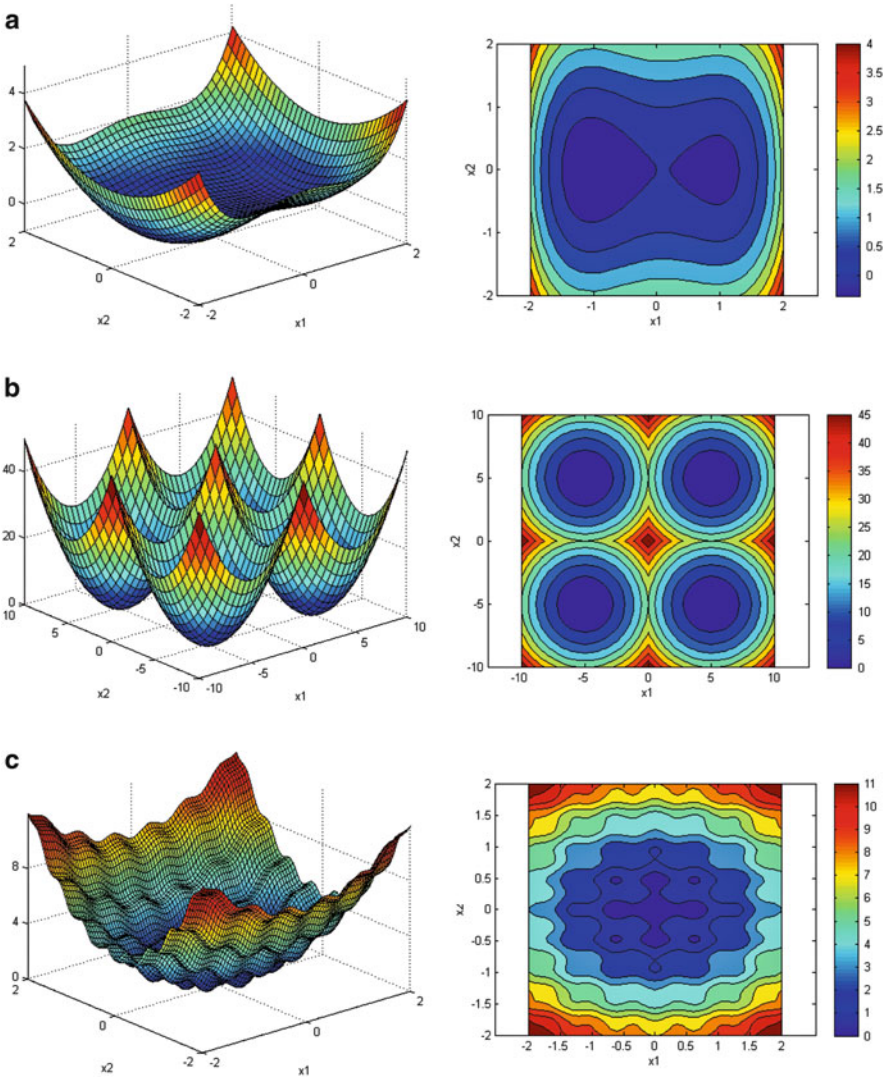

Fig. 8.6 A perspective view and the related contour lines for some of function when $n = 2$ [1]: (**a**) Aluffi-Pentiny, (**b**) Becker and lago, (**c**) Bohachevsky 1

The results show that the present algorithm has a higher efficiency than GA and some of its variants for these benchmark functions.

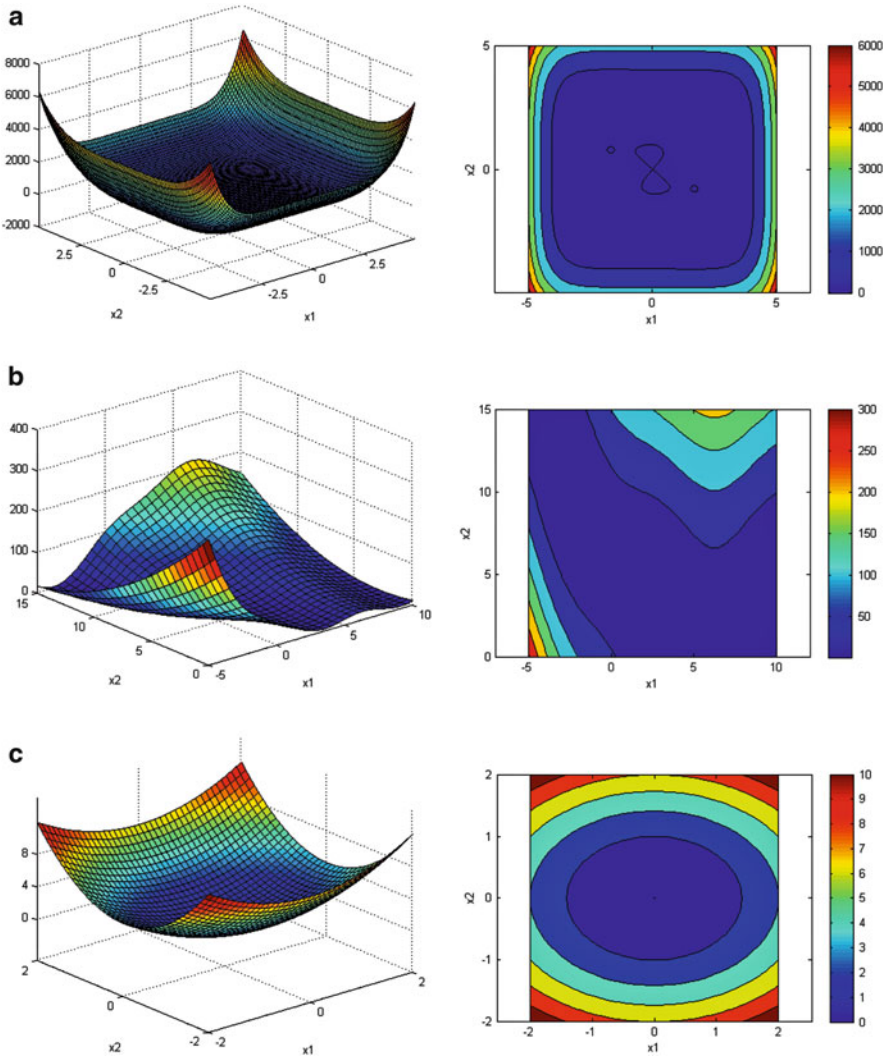

Fig. 8.7 A perspective view and the related contour lines for some of function when $n = 2$ [1]: (**a**) Camel, (**b**) Branin, (**c**) Bohachevsky 2

8.2.3.2 Engineering Design Problems

In this section, two engineering design problems are presented to show the efficiency of the RO. For the sake of simplicity, the penalty approach is used for constraint handling. In using the penalty function, if the constraints are not violated, the penalty will be zero; otherwise, the value of the penalty is calculated by dividing the violation of the allowable limit to the limit itself. It should be mentioned that in

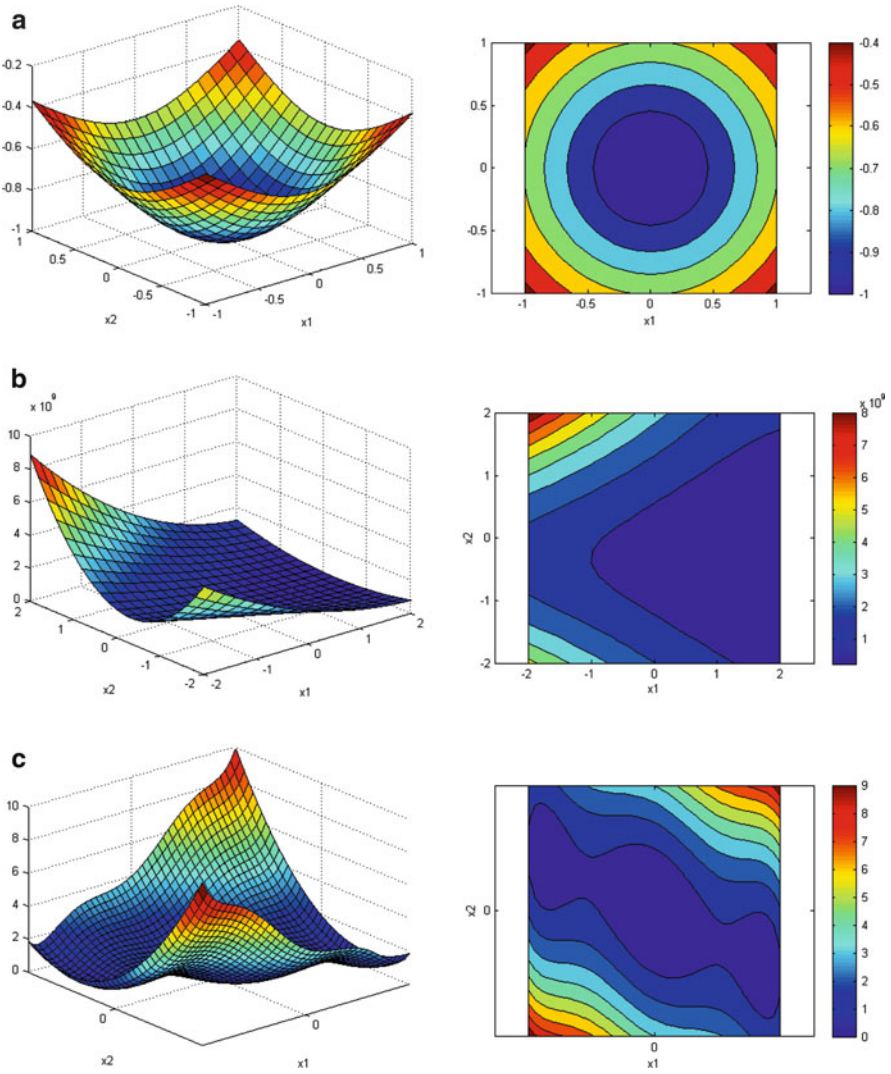

Fig. 8.8 A perspective view and the related contour lines for some of function when $n = 2$ [1]: (**a**) Exponential, (**b**) Goldstein and price, (**c**) Cb3

the RO, the use of this type of constraint handling is not a necessity and any other type of constraint handling approach can be employed.

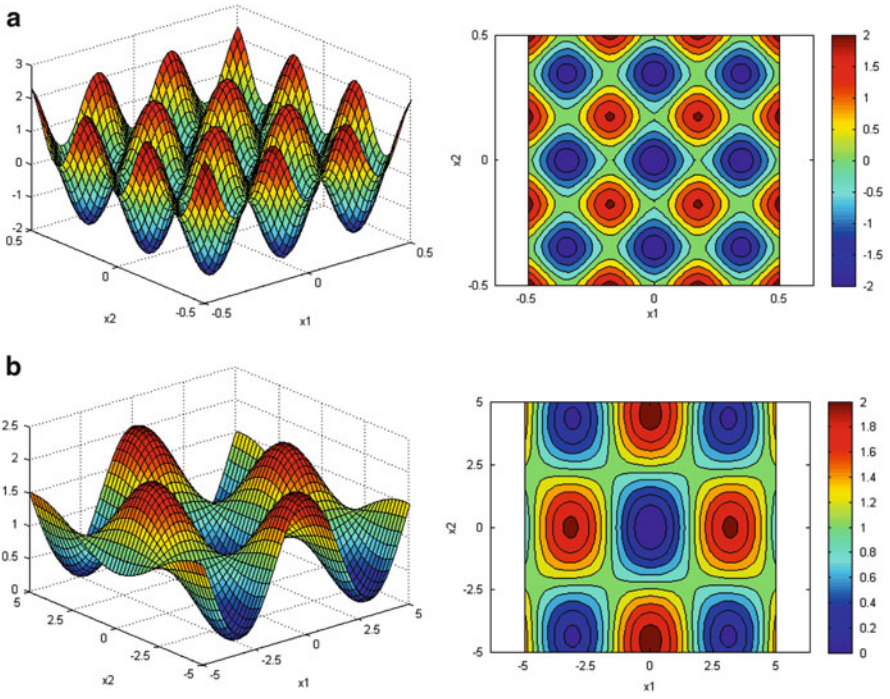

Fig. 8.9 A perspective view and the related contour lines for some of function when $n = 2$ [1]: (**a**) Griewank, (**b**) Rastrigin

A Tension/Compression Spring Design Problem

Belegunda [7] and Arora [8] described this problem for the first time. The goal of this problem is to minimize the weight of a tension/compression spring subjected to constraints on shear stress, surge frequency, and minimum deflection as shown in Fig. 8.10. Here, the design variables are the mean coil diameter D; the wire diameter d, and the number of active coils. Table 8.4 shows the cost function, constraints and the bounds of the design space.

Belegunda [7] has solved this problem using eight different mathematical optimization techniques (only the best results are shown). Also, it has been solved by Arora [8] using a numerical optimization technique called constraint at the constant cost. Finally, this problem is solved by RO using 40 agents, and the results of optimization are shown in Table 8.5. Considering these results, it can be concluded that the RO performs better than the above-mentioned methods. The average value and the standard deviation for 50 independents runs are 0.13547 and 0.001159, respectively.

Table 8.3 Performance comparison for the benchmark problems

FUNCTION	GEN	GEN–S	GEN–S–M–LS	Kaveh and Khayatazad [1]
AP	1,360(0.99)	1,360	1,253	331
Bf1	3,992	3,356	1,615	677
Bf2	20,234	3,373	1,636	582
BL	19,596	2,412	1,436	303
Branin	1,442	1,418	1,257	463
Camel	1,358	1,358	1,300	332
Cb3	9,771	2,045	1,118	262
CM	2,105	2,105	1,539	802
Dejoung	9,900	3,040	1,281	452
EXP2	938	936	807	136
EXP4	3,237	3,237	1,496	382
EXP8	3,237	3,237	1,496	1,287
EXP16	8,061	8,061	1,945	17,236(0.46)
GRIEWANK	18,838(0.91)	3,111(0.91)	1,652(0.99)	1,091(0.98)
RASTRIGIN	1,533(0.97)	1,523(0.97)	1,381	1,013(0.98)
Goldstein and Price	1,478	1,478	1,325	451
SHEKEL5	2,527(0.61)	2,507(0.61)	2,049(0.67)	3,401(0.52)
SHEKEL7	2,567(0.72)	2,500(0.72)	2,032(0.75)	3,459(0.76)
SHEKEL10	2,641(0.71)	2,598(0.71)	2,141(0.76)	4,152(0.66)
TOTAL	114,815(94.26)	49,655(94.31)	28,759(95.63)	36,812(91.36)

Fig. 8.10 A tension/compression spring

Table 8.4 Specifications of the tension/compression spring problem

Cost function	$f(\boldsymbol{X}) = (x_3 + 2)x_2x_1^2$
Constraint functions	$g_1(\boldsymbol{X}) = 1 - \frac{x_2^3 x_3}{71785x_1^4} \leq 0$
	$g_2(\boldsymbol{X}) = \frac{4x_2^4 - x_1x_2}{12566\left(x_2x_1^3 - x_1^4\right)} + \frac{1}{5108x_1^2} - 1 \leq 0$
	$g3(\boldsymbol{X}) = 1 - \frac{140.45x_1}{x_2^2x_3} \leq 0$
	$g4(\boldsymbol{X}) = \frac{x_1 + x_2}{1.5} - 1 \leq 0$
Variable region	$0.05 \leq x_1 \leq 2$
	$0.25 \leq x_2 \leq 1.3$
	$2 \leq x_3 \leq 15$

Table 8.5 Optimum results for the tension/compression spring design

	Optimal design variable			
Methods	X_1(d)	X_2(D)	X_3(N)	f_{cost}
Belegunda [7]	0.050000	0.315900	14.250000	0.0128334
Arora [8]	0.053396	0.399180	9.185400	0.0127303
Kaveh and Khayatazad [1]	0.051370	0.349096	11.76279	0.0126788

A Welded Beam Design Problem

The other well-studied engineering design problem is the welded beam design problem that is often used as a benchmark problem [9]. The goal is to find the minimum fabricating cost of the welded beam subjected to constraints on shear stress (τ), bending stress (σ), bulking load (P_c), end deflection (δ) and side constraint. The design variables are h($=x_1$), L($=x_2$), t($=x_3$) and b($=x_4$), Fig. 8.11. Table 8.6 shows the cost function, constraints and the bounds of the design space.

This problem has been solved by Radgsdell and Philips [9] and Deb [10]. Radgsdell and Philips compared optimal results of different optimization methods which were mainly based on mathematical optimization algorithms. These methods are APPROX (Griffith and Stewart's Successive linear approximation), DAVID (Deviation-Fletcher-Powell with a penalty function), SIMPLEX (Simplex method with a penalty function), and RANDOM (Richardson's random method) algorithms. Finally, RO solved this problem by using 40 agents in 50 independents runs. The results are collected in Table 8.7 indicating that the RO has a better solution than the above mentioned methods. The mean of the results and the standard deviation of 50 independent runs are 1.9083 and 0.173744, respectively.

8.3 Ray Optimization for Size and Shape Optimization of Truss Structures

In this part, Ray Optimization is employed for size and shape optimization of truss structures. The goal function of the present optimization method is the minimization of truss weight under the required constraints.

8.3.1 Formulation

Metaheuristic algorithms have been extensively used for solving the truss optimization. The mathematical formulation of this optimization problem can be expressed as:

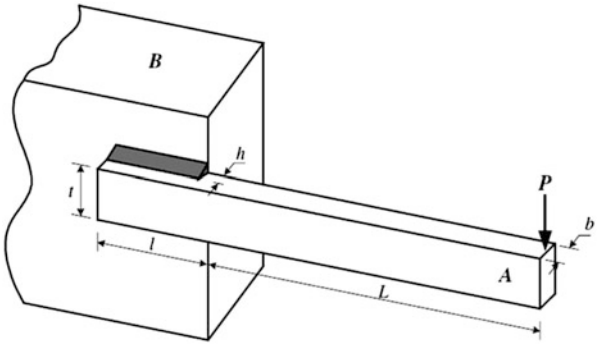

Fig. 8.11 A welded beam system

Table 8.6 Specifications of a welded beam design problem

Cost function	$f(\boldsymbol{X}) = 1.1047x_1^2x_2 + 0.04811x_3x_4(14.0 + x_2)$
Constraint functions	$g_1(\boldsymbol{X}) = \tau(\boldsymbol{X}) - \tau_{\max} \leq 0$
	$g2(\boldsymbol{X}) = \sigma(\boldsymbol{X}) - \sigma_{\max} \leq 0$
	$g_3(\boldsymbol{X}) = x_1 - x_4 \leq 0$
	$g_4(\boldsymbol{X}) = 0.1047x_1^2 + 0.04811x_3x_4(14.0 + x_2) - 5.0 \leq 0$
	$g_5(\boldsymbol{X}) = 0.125 - x_1 \leq 0$
	$g6(\boldsymbol{X}) = \delta(\boldsymbol{X}) - \delta_{\max} \leq 0$
	$g_7(\boldsymbol{X}) = P - P_c(\boldsymbol{X}) \leq 0$
	$\tau(\boldsymbol{X}) = \sqrt{(\tau')^2 + 2\tau'\tau''\frac{x_2}{2R} + +(\tau'')^2}$
	$\tau' = \frac{P}{\sqrt{2}x_1x_2}\ \tau'' = \frac{MR}{J}$
	$M = P\left(\frac{L+x_2}{2}^2\right), \quad R = \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1+x_3}{2}\right)^2}$
	$J = 2\left\{\sqrt{2}x_1x_2\left[\frac{x_2^2}{12} + \left(\frac{x_1+x_3}{2}\right)^2\right]\right\}$
	$\sigma(\boldsymbol{X}) = \frac{6PL}{x_4x_3^2}, \qquad \delta(\boldsymbol{X}) = \frac{4PL^3}{Ex_4x_3^3}$
	$P_c(\boldsymbol{X})^c = \frac{4.013E\sqrt{\frac{x_3^2x_4^6}{36}}}{L^2}\left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right)$
	$P = 6000lb,\ L = 14in$
	$E = 30 \times 10^6 psi,\ \mathrm{G} = 12 \times 10^6 psi$
	$\tau_{\max} = 13.6 \times 10^3 psi,\ \sigma_{\max} = 30 \times 10^3 psi,\ \delta_{\max} = 0.25in$
Variable regions	$0.1 \leq x_{1,4} \leq 2$
	$0.1 \leq x_{2,3} \leq 10$

Table 8.7 Optimum results for the design of welded beam

Methods	Optimal design variables				
	X_1(h)	X_2(l)	X_3(t)	X_4(b)	f_{cost}
Regsdell and Philips [9]					
APPROX	0.2444	6.2189	8.2915	0.2444	2.3815
DAVID	0.2434	6.2552	8.2915	0.2444	2.3841
SIMPLEX	0.2792	5.6256	7.7512	0.2796	2.5307
RANDOM	0.4575	4.7313	5.0853	0.6600	4.1185
Deb [10]	0.248900	6.173000	8.178900	0.253300	2.433116
Kaveh and Khayatazad [1]	0.203687	3.528467	9.004233	0.207241	1.735344

$$\begin{aligned} &\text{minimize} && W(\{x\}) = \sum_{i=1}^{n} \gamma_i . A_i . L_i(x) \\ &\text{subject to :} && \delta_{\min} \le \delta_i \le \delta_{\max}, && i = 1, 2, \ldots, m \\ &&& \sigma_{\min} \le \sigma_i \le \sigma_{\max}, && i = 1, 2, \ldots, n \\ &&& \sigma_i^b \le \sigma_i \le 0, && i = 1, 2, \ldots, ns \\ &&& A_{\min} \le A_i \le A_{\max}, && i = 1, 2, \ldots, ng \end{aligned} \tag{8.33}$$

where $W(\{x\})$ is the weight of the structure; n is the number of members making up the structure; m is the number of nodes; ns is the number of compression elements; ng is the number of groups (number of design variables); γ is the material density of the member i; L_i is the length of the member i which depends on the nodal coordinates; x; A_i is cross-sectional area of the member i chosen between A_{min} and A_{max}; min showing the lower bound and max indicating the upper bound; σ_i and δ_i are the stress and nodal deflection, respectively; σ_i^b is the allowable buckling stress in member i when it is in compression.

8.3.2 Design Examples

In this section, three examples are optimized utilizing the new algorithm:

- A 200-bar spatial truss structure;
- A model of First of Forth bridge;
- A 37-bar simply supported truss.

After optimizing these structures, their results are compared to the solution of the other methods to demonstrate the efficiency of the present method. For the first, second and third examples 100, 75 and 80 agents are used, respectively. Also, for these examples, the maximum number of iteration is considered as 400. In order to asses the effect of the initial solution vector on the final result and because of the random nature of the algorithm, these examples are independently optimized 20, 30 and 30 times, respectively. For the sake of simplicity, the penalty approach is used for constraint handling. In using the penalty function, if the constraints are not

violated, the penalty will be zero; otherwise the value of the penalty is calculated by dividing the violation of the allowable limit to the limit itself. It should be mentioned that in the RO, one does not necessarily need to use this type of constraint handling and any other type of constraint handling approach can be employed. All the algorithms, and the direct stiffness method for the analysis of truss structures are coded in MATLAB.

8.3.2.1 A 200-Bar Planar Truss Structure

The first example considered for size optimization is the 200-bar plane truss structure, shown in Fig. 8.12. All members are made of steel: the material density and modulus of elasticity are 0.283 lb/in^3 (7,933.410 kg/m^3) and 30,000 ksi (206,000 MPa), respectively. This truss is subjected to constraints only on stress limitations of $\pm$10 ksi (68.95 MPa). There are three loading conditions: (1) 1.0 kip (4.45 kN) acting in the positive x-direction at nodes 1, 6, 15, 20, 29, 43, 48, 57, 62, and 71; (2) 10 kips (44.5 kN) acting in the negative y-direction at nodes 1, 2, 3, 4, 5, 6, 8, 10, 12, 14, 15, 16, 17, 18, 19, 20, 22, 24,. . ., 71, 72, 73, 74, and 75; and (3) Conditions 1 and 2 acting together. The 200 members of this truss are divided into 29 groups, as shown in Table 8.8. The minimum cross-sectional area of all members is 0.1 in^2 (0.6452 cm^2) and the maximum cross-sectional area is 20 in^2 (129.03 cm^2).

The RO algorithm found the best weight as 25,193.228 lb after 326 iterations (32,600 analyses), with the standard deviation being 1,221.146 lb. Table 8.8 compares the optimal results of the PSO, PSOPC, HPSACO [5] and RO. Figure 8.13 shows the existing and allowable element stress values for Cases 1, 2 and 3. As can be seen, the design controlling factors in this problem are Cases 1 and 2.

8.3.2.2 A Model of the First of Forth Bridge

This example was first analyzed by Gil and Andreu [11] to obtain the optimal sizing and configuration variables. This structure is 16 m long and 1 m high truss beam of infinite span. But in this part, for better control on the shape variables, the height of the truss is changed to 3 m. If there are infinite span, the structure can be modeled as is shown in Fig. 8.14a. Because of symmetry of this problem, one can analysis half of the structure, Fig. 8.14b. Notice that it is necessary to adjust the bar cross-section and the force which lay on the axis of symmetry by considering half of them in the main structure and loading. In the field of shape optimization, the vertical coordinates of nodes which are free vertically, are the design variables. The origins of these nodes are those positions which are shown in Fig. 8.14b, and the range of their variation is in between $-$140 cm and 140 cm. Thus there are ten variables for shape optimization. In the field of size optimization, the cross-section of each bar, with their areas being 0.5 cm^2 through 100 cm^2 are other design variables. In Fig. 8.14b, the encircled numbers show the element grouping, and there are 16 groups of

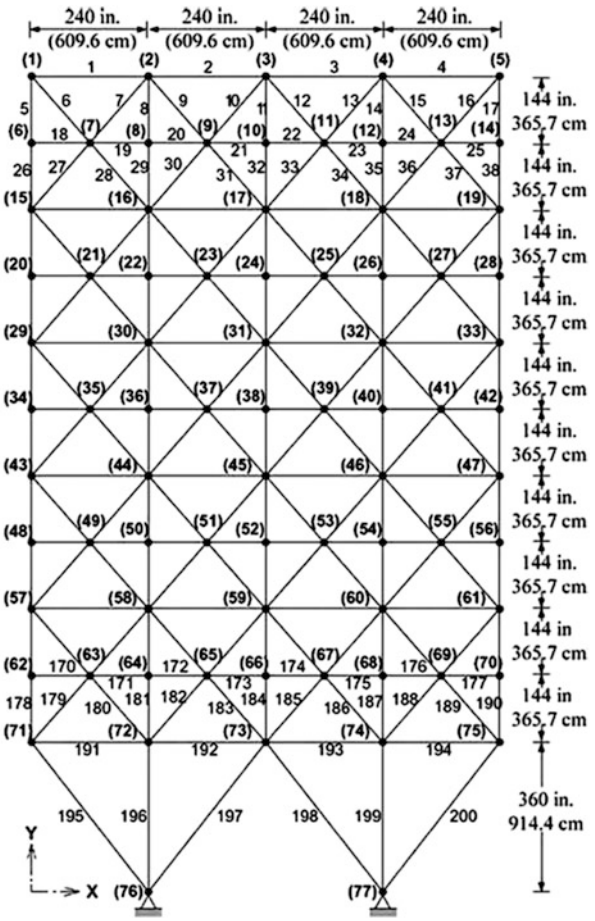

Fig. 8.12 Schematic of a 200-bar planar truss structure

element for size optimization. The modulus of elasticity of material is 2.1×10^8 kN/m^2, the allowable stress value is 25 kN/cm^2 and the specific weight of material is 7.8 ton/m^3. The applied forces are self-weight and the loads shown in Fig. 8.14a.

After designing the structure, the best weight of the structure is obtained as 11,215.7 kg and the corresponding cross-sections and coordinates are as provided in Table 8.9. The standard deviation and average weight for 30 independent runs is 545.5 kg and 11,969.2 kg, respectively. The prior work obtained 7,102 kg as the optimal weight [11]. Figure 8.15 shows the convergence rate for this structure. The best results obtained for the PSO and BB–BC are 20,591.9 kg and 37,132.3 kg, respectively. The standard deviation for 30 independent runs for the latter two methods are 2,323.7 kg and 1,235.4 kg, respectively. Finally 25,269.3 kg and 40,154.1 kg are obtained as the average weight for 30 independent runs for the PSO and BB–BC.

Table 8.8 Optimal design comparison for the 200-bar planar truss

		Optimal cross-sectional areas (in^2)				
		Kaveh and Talatahari [5]			Kaveh and Khayatazad [2]	
Group	Variables members (A_i, i = 1,. . .,200)	PSO	PSOPC	HPSACO	(in^2)	(cm^2)
1	1,2,3,4	0.8016	0.7590	0.1033	0.382	2.466
2	5,8,11,14,17	2.4028	0.9032	0.9184	2.116	13.647
3	19,20,21,22,23,24	4.3407	1.1000	0.1202	0.102	0.660
4	18,25,56,63,94,101,132,139,170,177	5.6972	0.9952	0.1009	0.141	0.909
5	26,29,32,35,38	3.9538	2.1350	1.8664	3.662	23.620
6	6,7,9,10,12,13,15,16,27,28,30,31,33,34,36,37	0.5950	0.4193	0.2826	0.176	1.137
7	39,40,41,42	5.6080	1.0041	0.1000	0.121	0.781
8	43,46,49,52,55	9.1953	2.8052	2.9683	3.544	22.861
9	57,58,59,60,61,62	4.5128	1.0344	0.1000	0.108	0.695
10	64,67,70,73,76	4.6012	3.7842	3.9456	5.565	35.892
11	44,45,47,48,50,51,53,54,65,66,68,69,71,72,74,75	0.5552	0.5269	0.3742	0.542	3.493
12	77,78,79,80	18.7510	0.4302	0.4501	0.138	0.890
13	81,84,87,90,93	5.9937	5.2683	4.96029	5.139	33.149
14	95,96,97,98,99,100	0.1000	0.9685	1.0738	0.101	0.652
15	102,105,108,111,114	8.1561	6.0473	5.9785	8.742	56.388
16	82,83,85,86,88,89,91,92,103,104,106,107,109,110,112,113	0.2712	0.7825	0.78629	0.431	2.781
17	115,116,117,118	11.1520	0.5920	0.73743	0.998	6.435
18	119,122,125,128,131	7.1263	8.1858	7.3809	7.212	46.516
19	133,134,135,136,137,138	4.4650	1.0362	0.66740	0.152	0.979
20	140,143,146,149,152	9.1643	9.2062	8.3000	8.452	54.516
21	120,121,123,124,126,127,129,130,141,142,144,145,147, 148,150,151	2.7617	1.4774	1.19672	0.835	5.383
22	153,154,155,156	0.5541	1.8336	1.0000	0.413	2.661
23	157,160,163,166,169	16.1640	10.6110	10.8262	10.146	65.440
24	171,172,173,174,175,176	0.4974	0.9851	0.1000	0.874	5.639
25	178,181,184,187,190	16.2250	12.5090	11.6976	11.384	73.428

26	158,159,161,162,164,165,167,168,179,180,182,183, 185,186,188,189	1.0042	1.9755	1.3880	1.197	7.721
27	191,192,193,194	3.6098	4.5149	4.9523	5.747	37.069
28	195,197,198,200	8.3684	9.8000	8.8000	7.823	50.461
29	196,199	15.5620	14.5310	14.6645	13.655	88.074
Weight (lb)		44,081.4	28,537.8	25,156.5	25,193.22	112,109.9

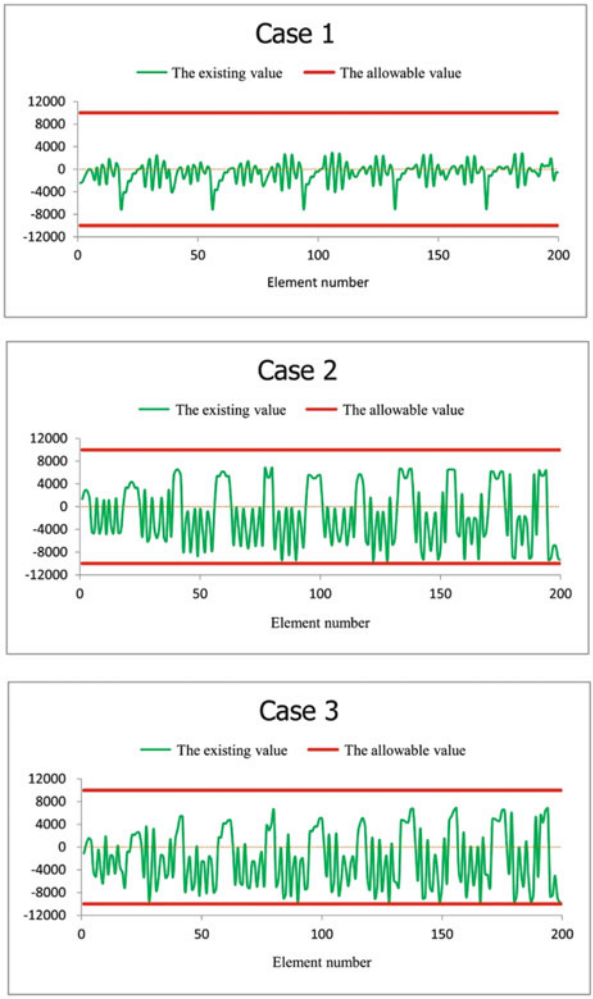

Fig. 8.13 Existing and allowable element stress value for the 200-bar planar truss [2]

Figure 8.14c shows the obtained optimal shape. Because of the support positions, the behavior of this truss is similar to that of a fixed beam. In a fixed beam bearing concentrated loads along its length, the internal moment at the ends and middle is greater than the other sections. Therefore for bearing this kind of loading, the moment of inertia in these sections must be greater. As can be seen, the optimal shape matches with this requirement.

The final result of concatenating infinite span with the optimal solution of the RO is shown in Fig. 8.14d. This final shape is similar to the famous bridge "the First of Forth Bridge", Fig. 8.14e. The reason of such difference is the construction requirements instead of saving material.

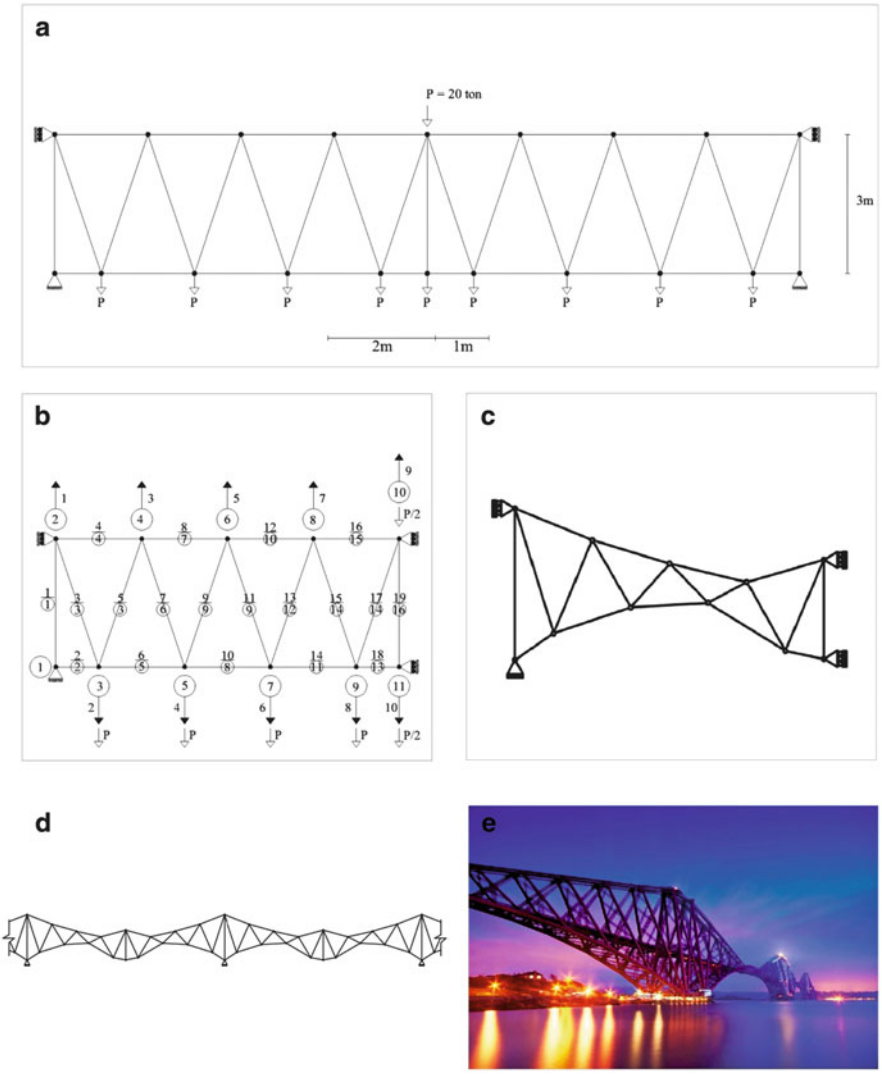

Fig. 8.14 (**a**) Problem diagram. (**b**) Analytical model. (**c**) Optimal shape. (**d**) Optimized configuration formed by concatenating basic modules. (**e**) The First of Forth Bridge, built during 1883–1890 [11]

8.3.2.3 A 37-Bar Simply Supported Truss

This example has been investigated by Wang et al. [12] using the evolutionary node shift method and by Lingyun et al. [13] utilizing the NHGA algorithm. It is a simple supported Pratt Type 37-bar truss as shown by Fig. 8.16a. There are non-structural masses of m = 10 kg attached to each of the bottom nodes of the lower chord, which

Table 8.9 Optimal cross-sections and coordinates for the model of first of forth bridge

	Optimal cross-section (cm^2)				Optimal coordinate (cm)		
Element group	BB–BC	PSO	Kaveh and Khayatazad [2]	Coordinate variable	BB–BC	PSO	Kaveh and Khayatazad [2]
1	56.41	25.20	20.54	1	6.89	140	70.88
2	58.20	97.60	44.62	2	17.74	139.65	64.88
3	53.89	35.00	6.37	3	1.81	117.59	−6.99
4	60.21	64.30	50.10	4	23.57	139.70	128.31
5	56.27	14.51	30.39	5	3.22	−16.51	−64.24
6	57.08	37.91	17.61	6	5.85	139.06	139.29
7	49.19	69.85	41.04	7	4.01	−127.74	−109.62
8	48.67	8.76	8.55	8	10.52	−81.03	21.82
9	45.43	47.54	33.93	9	−25.99	60.16	−55.09
10	15.14	6.36	0.63	10	−2.74	−139.97	2.29
11	45.31	27.13	26.92				
12	62.91	3.82	23.42				
13	56.77	50.82	42.06		BB–BC	PSO	Kaveh and Khayatazad [2]
14	46.66	2.70	2.01	Best weight (kg)	37,132.3	20,591.9	11,215.7
15	57.95	5.46	8.51	Average weight (kg)	40,154.1	25,269.3	11,969.2
16	54.99	17.62	1.27	Std (kg)	1,235.4	2,323.7	545.5

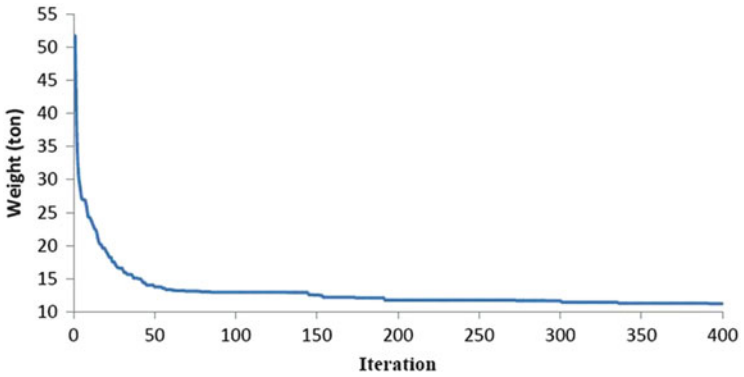

Fig. 8.15 Convergence rate for the model of the First of Forth Bridge [2]

are modeled as bar elements with rectangular cross-sectional areas of 4×10^{-3} m^2. The other bars are modeled as simple bar elements in the field of size optimization with the range of 1×10^{-4} m^2 thorough 3.5×10^{-4} m^2. The material property for the bar elements are selected as $E = 2.1 \times 10^{11}$ N/m^2 and $\rho = 7{,}800$ kg/m^3. Also, this example is considered as a truss shape optimization since all nodes of the upper

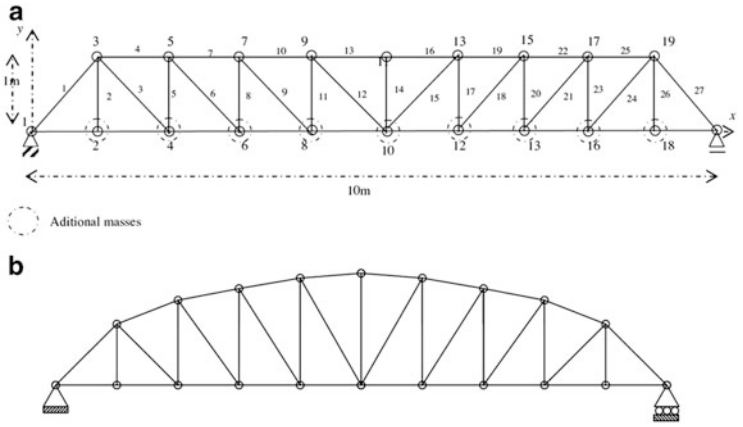

Fig. 8.16 (**a**) A 37-bar truss structure with added masses. (**b**) The 37-Bar structure optimized by the present work [12]

Table 8.10 Optimal design cross sections for different methods for the 37-bar truss structure

Variable no.	Wang et al. [12]	Lingyun et al. [13]	PSO [14]	Kaveh and Khayatazad [2]
Y_3,Y_{19} (m)	1.2086	1.1998	0.9637	1.0010
Y_5,Y_{17} (m)	1.5788	1.6553	1.3978	1.3909
Y_7,Y_{15} (m)	1.6719	1.9652	1.5929	1.5893
Y_9,Y_{13} (m)	1.7703	2.0737	1.8812	1.7507
Y_{11} (m)	1.8502	2.3050	2.0856	1.8336
A_1,A_{27} (cm^2)	3.2508	2.8932	2.6797	3.0124
A_2,A_{26} (cm^2)	1.2364	1.1201	1.1568	1.0623
A_3,A_{24} (cm^2)	1.0000	1.0000	2.3476	1.0005
A_4,A_{25} (cm^2)	2.5386	1.8655	1.7182	2.2647
A_5,A_{23} (cm^2)	1.3714	1.5962	1.2751	1.6339
A_6,A_{21} (cm^2)	1.3681	1.2642	1.4819	1.6717
A_7,A_{22} (cm^2)	2.4290	1.8254	4.6850	2.0591
A_8,A_{20} (cm^2)	1.6522	2.0009	1.1246	1.6607
A_9,A_{18} (cm^2)	1.8257	1.9526	2.1214	1.4941
A_{10},A_{19} (cm^2)	2.3022	1.9705	3.8600	2.4737
A_{11},A_{17} (cm^2)	1.3103	1.8294	2.9817	1.5260
A_{12},A_{15} (cm^2)	1.4067	1.2358	1.2021	1.4823
A_{13},A_{16} (cm^2)	2.1896	1.4049	1.2563	2.4148
A_{14} (cm^2)	1.0000	1.0000	3.3276	1.0034
Weight (kg)	366.50	368.84	377.20	364.04

chord are allowed to vary in the y-axis in a symmetrical manner in the range 1 m thorough 2.5 m. There are three constraints in the first three natural frequencies as $\omega_1 = 20$ Hz, $\omega_2 = 40$ Hz and $\omega_3 = 60$ Hz. Therefore, it is considered as a truss optimization problem with three frequency constraints and nineteen design variables (five shape variables and 14 sizing variables). Table 8.10 shows a comparison

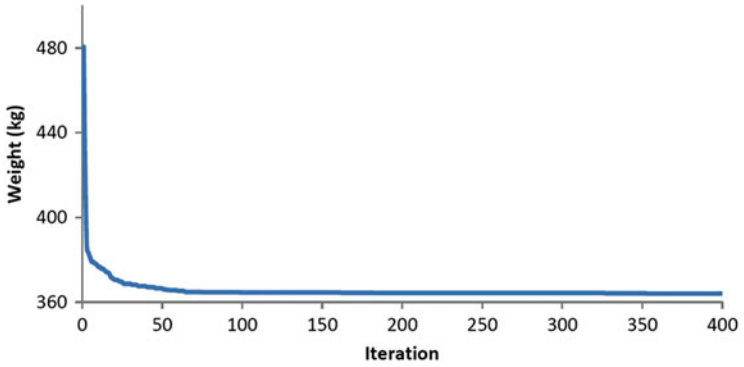

Fig. 8.17 Convergence history of the RO for the simply supported 37-bar truss with added masses [2]

among the optimal design cross sections of several methods including the present work (RO). As it can be seen, the best result for the RO, Wang et al. [12], Lingyun et al. [13], PSO [14] are 364.04 kg, 366.5 kg, 368.84 kg and 377.20 kg, respectively.

Figure 8.16b shows the final truss shape of the design optimizations, and Fig. 8.17 illustrates the weight convergence history for the RO algorithm for the 37-bar truss with added masses.

8.4 An Improved Ray Optimization Algorithm for Design of Truss Structures

8.4.1 Introduction

This part develops an improved ray optimization (IRO) algorithm for solving optimization problems. IRO employs a new approach for generating new solution vectors which has no limitation on the number of variables, so in the process of algorithm there is no need to divide the variables into groups like standard RO. The procedure which returns the violated agents into feasible search space is also modified. The Simulation results of the IRO for benchmark mathematical optimization problems and truss structures are compared to those of the standard RO and some well-known metaheuristic algorithms, respectively. Numerical results indicate the effectiveness and robustness of the proposed IRO [3].

8.4.2 Improved Ray Optimization Algorithm

In the improved ray optimization algorithm, a memory which saves some or all the historically best positions of agents, is considered as local best memory (***LBM***). If the so far best positions of all agents are saved (especially when the number of agents is large), the computational cost grows. Therefore in this technique, when the number of agents are more than or equal to 25, the size of the local best memory is considered as 25, otherwise the size of the LBM is taken as half number of agents.

When an agent violates the boundary limitations in standard RO, all the components are changed. However in IRO, only the components that violate the boundary are refunded. This violated component must be regenerated by the following formula:

$$X_{ij}^{k+1} = X_{ij}^{k} + 0.9\left(Int_{ij} - X_{ij}^{k}\right) \tag{8.34}$$

Where X_{ij}^{k+1} and X_{ij}^{k} are the refined component and component of the *j*th variable for the *i*th agent in *(k + 1)*th and *k*th iteration, respectively. Int_{ij} is the intersection point of violated agent with boundary (If an agent violates a boundary, it intersects the boundary at a specified point, because of having definite movement vector).

The main idea of standard RO is approximating the new movement vector with a normal vector. To achieve this aim, if the number of variables was more than three, the proposed formula cannot be applied directly and first the main problem must be divided into some sub-problems and after the calculation, merge the results of the sub-problems to evaluate the goal function. When the number of variables is large the computational cost grows considerably. Instead of this approach the following formula (which has no limit on the number of variables) is applied to calculate the direction of the new movement vector as illustrated in Fig. 8.18.

$$\boldsymbol{Tv_i} = O_i - X_i \tag{8.35}$$

$$\boldsymbol{V_i^{k+1}} = \alpha.Tv_i^{\,k} + \beta.V_i^{k} \tag{8.36}$$

Where $\boldsymbol{Tv_i}$ is target vector, $\boldsymbol{V}_{\text{i}}^{\text{k+1}}$and $\boldsymbol{V}_{\text{i}}^{\text{k}}$ are movement vectors in *(k + 1)*th and *k*th iteration, respectively. $\boldsymbol{O}_i^{\,k}$ is the origin according to (8.27) but in this method, ***LB*** is considered randomly from local best memory. α and β are the factors that control the exploitation and exploration as shown in Fig. 8.19. An efficient optimization algorithm should perform good exploration in early iterations and good exploitation in final iterations. Thus, α and β are increasing and decreasing functions respectively, and are defined as:

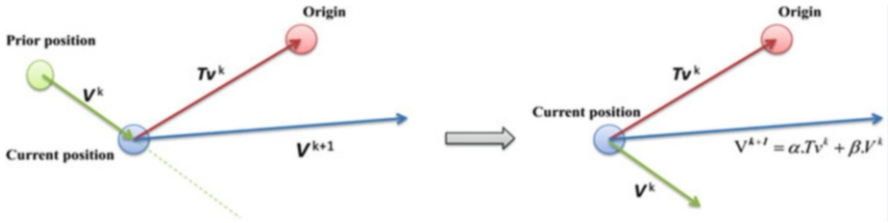

Fig. 8.18 Generation of the new movement vector [3]

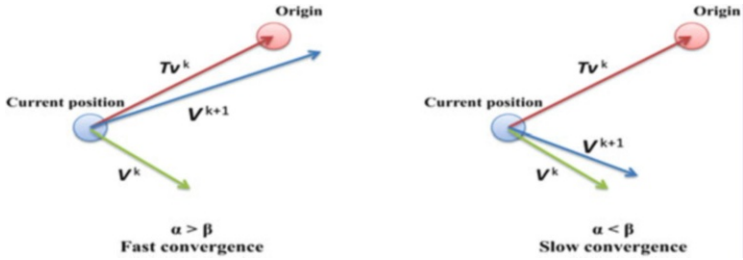

Fig. 8.19 Influence of the α and β parameters [3]

$$\alpha = 1 + \left(\frac{k}{ite}\right) \tag{8.37}$$

$$\beta = 1 - 0.5\left(\frac{k}{ite}\right) \tag{8.38}$$

Finally all the $\boldsymbol{V}_{\mathrm{i}}^{\mathrm{k+1}}$ vectors should be normalized.

The magnitude of movement vectors must be calculated because in the previous formulas only the direction of the movement vector is defined.

One of the important features of each metaheuristic algorithm is its ability to escape from the trap when agents fall into a local minimum, so in the standard RO there is a possibility like ***stoch*** that specifies whether a movement vector must be changed or not, therefore we have

(a) with probability like ***stoch***,

$$\boldsymbol{V}_{ij}^{k+1} = -1 + 2.rand \tag{8.39}$$

$$\boldsymbol{V}_i^{K+1} = \frac{\boldsymbol{V}_i^{k+1}}{norm\left(\boldsymbol{V}_i^{k+1}\right)} \cdot \frac{a}{d}.rand \tag{8.40}$$

(b) with probability like (***1-stoch***),

If $norm(O_i^k - X_i^k) = 0$,

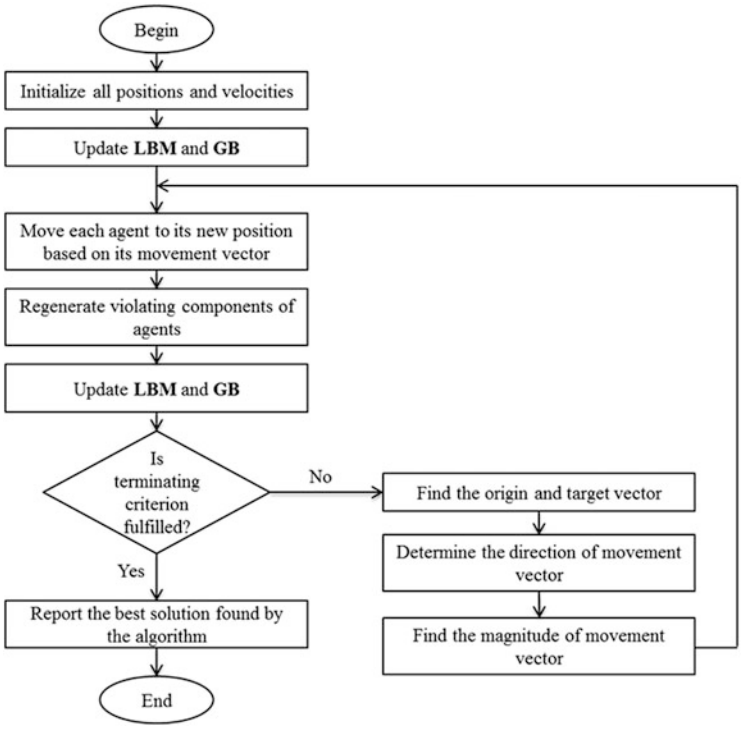

Fig. 8.20 The flowchart of the IRO [3]

$$\boldsymbol{V}_i^{k+1} = \frac{\boldsymbol{V}_i^k}{norm\left(\boldsymbol{V}_i^k\right)}.rand.0.001 \tag{8.41}$$

Otherwise,

$$\boldsymbol{V}_i^{k+1} = \boldsymbol{V}_i^{k+1}.norm\left(\boldsymbol{X}_i^k - \boldsymbol{O}_i^k\right) \tag{8.42}$$

In the case of problems that have function constraints (behavior constraints), the following formulas are utilized instead of (8.42)

$$\boldsymbol{V}_i^{K+1} = \boldsymbol{V}_i^{k+1}.\frac{a}{d} \tag{8.43}$$

$$d = d + r.d.\left(\frac{k}{ite}\right) \tag{8.44}$$

Where r is a constant factor and when the number of iterations rises, the value of d increase and it help the algorithm to handle the constraint well. a and d are the factors that were defined in Sect. 8.2.2.3.

The flowchart of the IRO algorithm is illustrated in Fig. 8.20.

8.4.3 Mathematical and Structural Design Examples

8.4.3.1 Standard Mathematical Functions

To test the ability of the proposed algorithm and to compare its results with those of the standard RO, some benchmark mathematical functions are considered as in Sect. 8.2.3.1. Each of these functions tests the optimization algorithm in special conditions, identifying the weak points of the optimization algorithm. These functions are selected by Tsoulos [6] for evaluating modifications of Genetic Algorithm and utilized by Kaveh and Khayatazad [1] for investigating the standard RO.

The IRO method with different number of agents has been tested for some of these functions. Assigning the number of agents as 50 in the CM, Griewank and Rastring functions and as 10 for other ones shows a better performance. We also have tried to tune other parameters of algorithm. From our simulations it is recommended to set parameters as 0.35 and 700 for ***Stoch*** and d, respectively. After implementing IRO algorithm using MATLAB, it has been run independently 50 times to carry out meaningful statistical analysis. The algorithm stops when the variations of function values are less than a given tolerance as 10^{-4}. Table 8.11 reports the performance of GEN–S–M–LS as the best modificationof GA [6], the standard RO [1] and proposed IRO respectively, where the numbers are in the format: average number of evaluations $\pm$ one standard deviation (success rate). Considering this table, the standard RO and the improved RO show better performances in terms of the required number of analyses and success rate.

8.4.3.2 Continuous and Discrete Trusses

In this section, common truss optimization examples as benchmark problems are optimized with the IRO algorithm. The final results are compared to the solutions of other methods to demonstrate the efficiency of the IRO. In the sequel, penalty function formula and constraint conditions for truss structures are briefly overviewed at the first subsection then the examples are presented. The examples contain a 25-bar transmission tower, a 72-bar spatial truss and a dome shaped space truss. From our simulations, setting the number of agents and ***Stoch*** 25 and 0.35 are efficient for design examples, respectively. Table 8.12 tabulates other parameters for each case.

Optimum Design of Truss Structures

In order to handle the constraints, a penalty approach is utilized. In this method, the aim of the optimization is redefined by introducing the cost function as:

Table 8.11 Performance comparison for the benchmark problems

FUNCTION	GEN–S–M–LS	Ray optimization	Kaveh et al. [3]
AP	1,253	331	$253 \pm 38.7985(100)$
Bf1	1,615	677	$438 \pm 48.4636(100)$
Bf2	1,636	582	$395 \pm 45.9674(100)$
BL	1,436	303	$194 \pm 31.2733(100)$
Branin	1,257	463	$312 \pm 81.0515(100)$
Camel	1,300	332	$184 \pm 21.0855(100)$
Cb3	1,118	262	$247 \pm 36.4549(100)$
CM	1,539	802	$1{,}290 \pm 65.3543(100)$
Dejoung	1,281	452	$213 \pm 26.3344(100)$
EXP2	807	136	$90 \pm 20.5115(100)$
EXP4	1,496	382	$220 \pm 50.5624(100)$
EXP8	1,496	1,287	$512 \pm 97.7743(100)$
EXP16	1,945	17,236(0.46)	$1{,}141 \pm 142.76(100)$
GRIEWANK	1,652(0.99)	1,091(0.98)	$1{,}383 \pm 100.3458(100)$
RASTRIGIN	1,381	1,013(0.98)	$1{,}662 \pm 202.3105(100)$
Goldstein and Price	1,325	451	$361 \pm 59.0105(100)$
TOTAL	22,537(99.94)	25,800(96.38)	8,895(100)

Table 8.12 Algorithm parameters for truss examples

Parameter	25-bar	72-bar	120-bar
d	15	15	10
r	7	7	20

$$f_{\cos t}(\{X\}) = (1 + \varepsilon_1 . \upsilon)^{\varepsilon_2} \times W(\{X\}), \quad \upsilon = \sum_{j=1}^{n} \max\left[0, g_j(\{X\})\right] \tag{8.45}$$

$$g_j(\{X\}) \leq 0, j = 1, 2, \ldots, n \tag{8.46}$$

Where $W(\{x\})$ is the weight of the structure (Sect. 8.3.1), υ denotes the sum of the violations of the design, $g_j\{X\}$ denotes design constraints and n represents the number of constraints. The constants ε_1 and ε_2 are selected considering the exploration and the exploitation rate of the search space. Here, ε_1 is set to unity, ε_2 is selected in a way that it decreases the penalties and reduces the cross-sectional areas. Thus, in the first steps of the search process, ε_2 is set to 1.5 and ultimately increased to 3.

The constraint conditions for truss structures are briefly explained in the following. The stress limitations of the members are imposed according to the provisions of ASD-AISC [15] as follows:

$$\begin{cases} \sigma_i^+ = 0.6\,\mathrm{F}_y & \text{for } \sigma_i \geq 0 \\ \sigma_i^- & \text{for } \sigma_i < 0 \end{cases} \tag{8.47}$$

$$\sigma_i^- = \begin{cases} \left[\left(1 - \dfrac{\lambda_i^2}{2\,\mathrm{c}_c^2}\right)F_y\right] \Big/ \left(\dfrac{5}{3} + \dfrac{3\lambda_i}{8\,c_c} + \dfrac{\lambda_i^3}{8\,\mathrm{c}_c^3}\right) & \textit{for } \lambda_i \geq c_c \\ \dfrac{12\,\pi^2\,E}{23\,\lambda_i^2} & \textit{for } \lambda_i \geq c_c \end{cases} \tag{8.48}$$

Where E is the modulus of elasticity; F_y is the yield stress of steel; c_c denotes the slenderness ratio (λ_i) dividing the elastic and inelastic buckling regions ($c_c = \sqrt{2\pi^2 E/F_y}$); λ_i = the slenderness ratio ($\lambda_i = kl_i/r_i$); k = the effective length factor; L_i = the member length; and r_i = the radius of gyration. The radius of gyration (r_i) can be expressed in terms of cross-sectional areas as $r_i = a\,A_i^b$. Here, a and b are the constants depending on the types of sections adopted for the members such as pipes, angles, and tees. In this study, pipe sections ($a = 0.4993$ and $b = 0.6777$) are adopted for bars [16].

The other constraint corresponds to the limitation of the nodal displacements:

$$\delta_i - \delta_i^u \leq 0 \qquad i = 1, 2, \ldots, nn \tag{8.49}$$

Where δ_i is the nodal deflection; δ_i^u is the allowable deflection of node i; and nn is the number of nodes.

A 25-Bar Space Truss with Discrete Variables

The 25-bar transmission tower is used widely in structural optimization to verify various metaheuristic algorithms. The topology and nodal numbering of the truss is shown in Fig. 8.21 [17]. The material density is considered as 0.1 lb/in^3 (2,767.990 kg/m^3) and the modulus of elasticity is taken as 10^7 psi (68,950 MPa). Twenty-five members are categorized into eight groups, as follows: (1) A_1, (2) A_2–A_5, (3) A_6–A_9, (4) A_{10}–A_{11}, (5) A_{12}–A_{13}, (6) A_{14}–A_{17}, (7) A_{18}–A_{21}, and (8) A_{22}–A_{25}.

A single load case {(kips) (kN)} is applied to the structure, at nodes 1, 2, 3 and 4 as follows: 1{(0, −10, −10) (0, −44.5, −44.5)}, 2{(1, −10, −10) (4.45, −44.5, −44.5)}, 3{(0.6, 0, 0) (2.67, 0, 0)} and 4{(0.5, 0, 0) (2.225, 0, 0)}. The allowable stresses and displacements are respectively ±40 ksi (275.80 MPa) for each member and ±0.35 in (±8.89 mm) for each node in the x, y and z directions. The range of discrete cross-sectional areas is from 0.1 to 3.4 in^2 (0.6452 to 21.94 cm^2) with 0.1 in^2 (0.6452 cm^2) increment (resulting in 34 discrete cross sections) for each of the eight element groups [18].

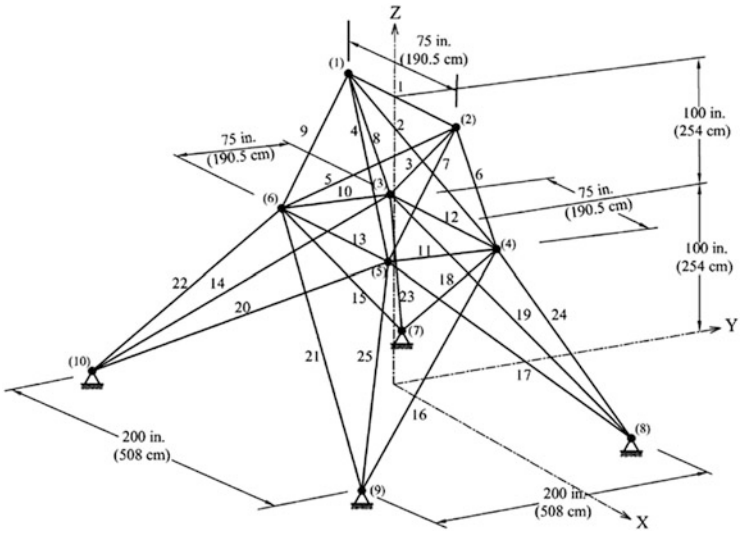

Fig. 8.21 Schematic of a 25-bar space truss

Table 8.13 Performance comparison for the 25-bar spatial truss

		Optimal cross-sectional areas (in^2)					
						Kaveh et al. [3]	
Element group		GA [18]	GA [18]	ACO [19]	BB–BC phases 1, 2 [18]	in^2	cm^2
1	A_1	0.10	0.10	0.10	0.10	0.10	0.645
2	A_2–A_5	1.80	0.50	0.30	0.30	0.30	1.935
3	A_6–A_9	230	3.40	3.40	3.40	3.40	21.935
4	A_{10}–A_{11}	020	0.10	0.10	0.10	0.10	0.645
5	A_{12}–A_{13}	010	1.90	2.10	2.10	2.10	13.548
6	A_{14}–A_{17}	0.80	0.90	1.00	1.00	1.00	6.452
7	A_{18}–A_{21}	.80	0.50	0.50	0.50	0.50	3.226
8	A_{22}–A_{25}	3.00	3.40	3.40	3.40	3.40	21.935
Best weight (lb)		546.01	485.05	484.85	484.85	484.85	219.92 (kg)
Average weight (lb)		N/A	N/A	486.46	485.10	484.90	219.94 (kg)
Number of analyses		800	15,000	7,700	9,000	925	

Table 8.13 presents the performance of the IRO and other algorithms. The IRO algorithm achieves the best solution weighted by 484.85 lb (219.92 kg), after 925 analyses. Although, this is identical to the best design developed using BB–BC algorithm [18] and a multiphase ACO procedure [19], it perform better than others when the number of analyses and average weight for 50 runs are compared.

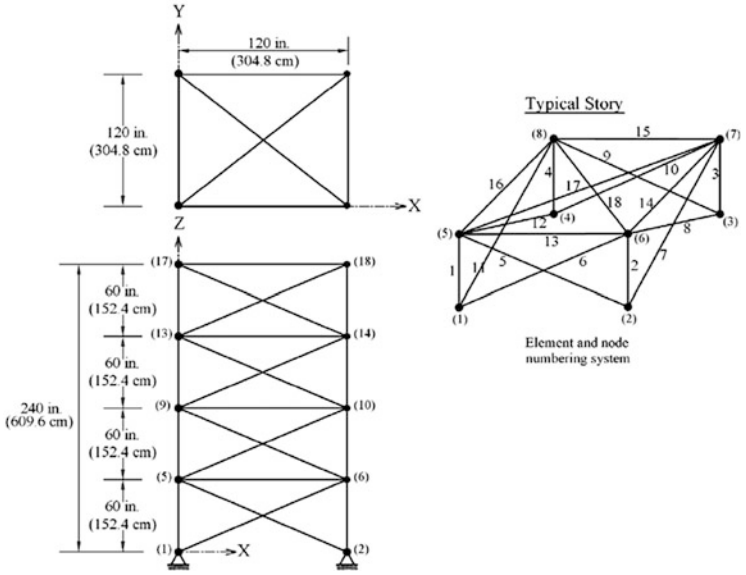

Fig. 8.22 Schematic of a 72-bar space truss

Table 8.14 Multiple loading conditions for the 72-bar truss

Case	Node	F_x kips (kN)	F_y kips (kN)	F_z kips (kN)
1	17	0.0	0.0	−5.0 (−22.25)
	18	0.0	0.0	−5.0 (−22.25)
	19	0.0	0.0	−5.0 (−22.25)
	20	0.0	0.0	−5.0 (−22.25)
2	17	5.0 (22.25)	5.0 (22.25)	−5.0 (−22.25)

A 72- Bar Space Truss with Discrete Variables

For the 72-bar spatial truss structure shown in Fig. 8.22 [20], the material density is 0.1 lb/in^3 (2,767.990 kg/m^3) and the modulus of elasticity is 10^7 psi (68,950 MPa). The 72 structural members of this spatial truss are categorized into 16 groups using symmetry: (1) A_1–A_4, (2) A_5–A_{12}, (3) A_{13}–A_{16}, (4) A_{17}–A_{18}, (5) A_{19}– A_{22}, (6) A_{23}–A_{30}, (7) A_{31}–A_{34}, (8) A_{35}–A_{36}, (9) A_{37}–A_{40}, (10) A_{41}–A_{48}, (11) A_{49}–A_{52}, (12) A_{53}–A_{54}, (13) A_{55}–A_{58}, (14) A_{59}–A_{66} (15), A_{67}– A_{70}, and (16) A_{71}–A_{72}. In this example, designs for a multiple load cases using discrete design variables are performed. The values and directions of the two load cases applied to the 72-bar spatial truss are listed in Table 8.14. The members are subjected to the stress limits of ±25 ksi (±172.375 MPa). Maximum displacement limitations of ±0.25 in (±6.35 mm), are imposed on every node in every direction and on the uppermost nodes in both x and y directions respectively.

Table 8.15 Performance comparison for the 72-bar spatial truss with discrete variable

Element group		Optimal cross-sectional areas (in^2)						
		GA [21]	PSOPC [21]	HPSO [21]	HPSACO [22]	ICA [21]	Kaveh et al. [3]	
							in^2	cm^2
1	A_1–A_4	0.196	4.490	4.970	1.800	1.99	1.99	12.839
2	A_5–A_{12}	0.602	1.457	1.228	0.442	0.442	0.563	3.632
3	A_{13}–A_{16}	0.307	0.111	0.111	0.141	0.111	0.111	0.716
4	A_{17}–A_{18}	0.766	0.111	0.111	0.111	0.141	0.111	0.716
5	A_{19}–A_{22}	0.391	2.620	2.880	1.228	1.228	1.228	7.923
6	A_{23}–A_{30}	0.391	1.130	1.457	0.563	0.602	0.563	3.632
7	A_{31}–A_{34}	0.141	0.196	0.141	0.111	0.111	0.111	0.716
8	A_{35}–A_{36}	0.111	0.111	0.111	0.111	0.141	0.111	0.716
9	A_{37}–A_{40}	1.800	1.266	1.563	0.563	0.563	0.563	3.632
10	A_{41}–A_{48}	0.602	1.457	1.228	0.563	0.563	0.442	2.852
11	A_{49}–A_{52}	0.141	0.111	0.111	0.111	0.111	0.111	0.716
12	A_{53}–A_{54}	0.307	0.111	0.196	0.250	0.111	0.111	0.716
13	A_{55}–A_{58}	1.563	0.442	0.391	0.196	0.196	0.196	1.265
14	A_{59}–A_{66}	0.766	1.457	1.457	0.563	0.563	0.563	3.632
15	A_{67}–A_{70}	0.141	1.228	0.766	0.442	0.307	0.391	2.523
16	A_{71}–A_{72}	0.111	1.457	1.563	0.563	0.602	0.563	3.632
Weight (lb)		427.203	941.82	933.09	39,380	392.84	389.33	176.60 (kg)
Number of analyses		N/A	150,000	50,000	5,330	4,500	17,925	

In this case, the discrete variables are selected from 64 discrete values from 0.111 to 33.5 in^2 (71.613 to 21,612.860 mm^2). For more information, the reader can refer to Table 2 in Kaveh and Talatahari [21].

Table 8.15 shows the best solution vectors, the corresponding weights and the required number of analyses for present algorithm and some other metaheuristic algorithms. The IRO algorithm can find the best design among the other existing studies. Although the number of required analyses by the IRO algorithm is more than ICA algorithm, but the best weight of the IRO algorithm is 389.33 lb (176.60 kg) that is 3.51 lb (1.59 kg) lighter than the best result obtained by ICA

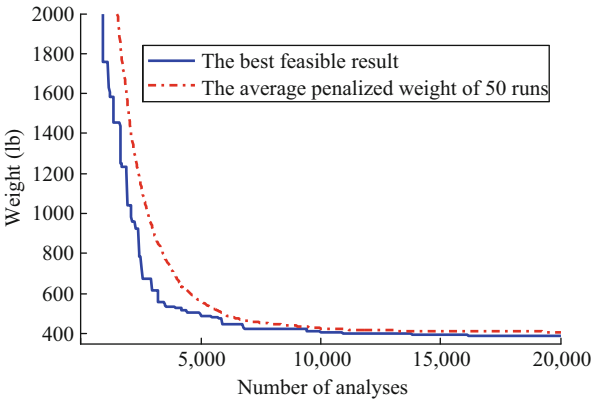

Fig. 8.23 Convergence history of the 72-bar space truss [3]

algorithm [21].The convergence history of the best result and the average penalized weight of 50 runs are shown in Fig. 8.23. Convergence speed in IRO is acceptable and step-like movements in diagram of IRO performance exhibit how it escapes from local minimum points, to find a better optimum point. It is important to note that this case has an expended search space than is requisite. The performance of the IRO decreased from 389.87 ± 1.1643 to 408.17 ± 71.2108 considering 47 and all 50 independent runs, respectively. In the other words, IRO yields to unexpected designs in just three of 50 independent runs. Unfortunately comprehensive statistical study of this case is not available in optimization literature.

Design of a 120-Bar Dome Shaped Truss with Continuous Variables

The topology, nodal numbering and element grouping of the 120-bar dome truss are shown in Fig. 8.24. For clarity, not all the element groups are numbered in this figure. The 120 members are categorized into seven groups, because of symmetry. Other conditions of problem are as follows [21], the modulus of elasticity is 30,450 ksi (210,000 MPa) and the material density is 0.288 lb/in^3 (7,971.810 kg/m^3). The yield stress of steel is taken as 58.0 ksi (400 MPa). The dome is considered to be subjected to vertical loading at all the unsupported joints. These loads are taken as -13.49 kips (-60 kN) at node 1, -6.744 kips (-30 kN) at nodes 2 through 14, and -2.248 kips (-10 kN) at the rest of the nodes. The minimum cross-sectional area of all members is 0.775 in^2 (5 cm^2) and the maximum cross-sectional area is taken as 20.0 in^2 (129.032 cm^2). The constraints are stress constraints [as defined by (8.47) and (8.48)] and displacement limitations of ± 0.1969 in (± 5 mm), imposed on all nodes in x, y and z directions.

Table 8.16 shows the best solution vectors, the corresponding weights and the required number of analyses for convergence of the present algorithm and some other metaheuristic algorithms. The IRO-based algorithm needs 18,300 analyses to find the best solution while this number is equal to 150,000, 32,600, 10,000, 10,000, 7,000 and 6,000 analyses for a PSO-based algorithm [5], a PSO and ACO hybrid

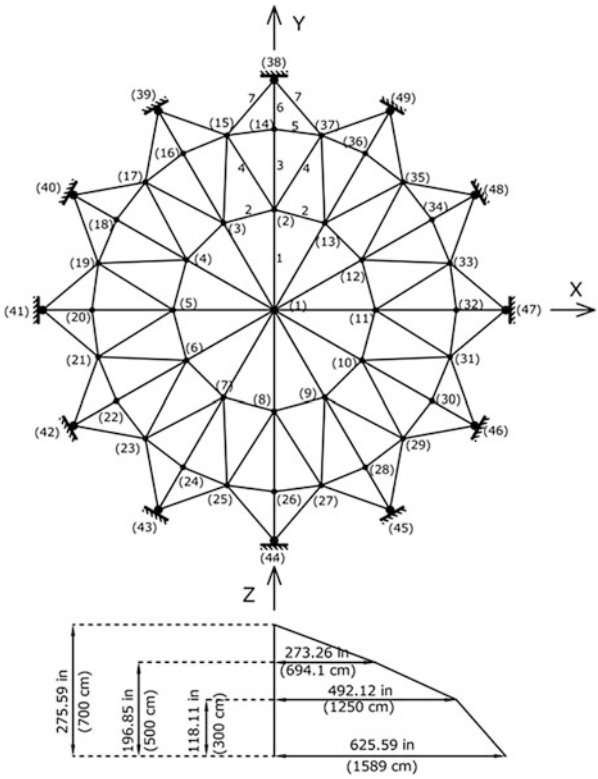

Fig. 8.24 Schematic of a 120-bar dome shaped truss

algorithm [5], a combination algorithm based on PSO, ACO and HS [5], an improved BB–BC method using PSO properties [20], the CSS algorithm [17] and the ICA algorithm [21], respectively. As a result, the IRO optimization algorithm only has better convergence rates than PSOPC and PSACO algorithms. Comparing the final results of the IRO and those of the other metaheuristics shows that IRO finds the so nearly optimum design to the best results of other efficient methods while the difference between the result of the IRO and that obtained by the HPSACO [5], as the first best result, is 9 lbs. A comparison of the allowable and existing stresses and displacements of the 120-bar dome truss structure using IRO is shown in Fig. 8.25. The maximum value for displacement is equal to 0.1969 in (5 mm) and the maximum stress ratio is equal to 99.99 %.

Table 8.16 Performance comparison for the 120-bar dome shaped truss with continuous variables

		Optimal cross-sectional areas (in^2)							
								Kaveh et al. [3]	
Element group		PSOPC [5]	PSACO [5]	HPSACO [5]	HBB–BC [20]	CSS [17]	ICA [21]	in^2	cm^2
1	A_1	3.040	3.026	3.095	3.037	3.027	3.0275	3.0252	19.5174
2	A_2	13.149	15.222	14.405	14.431	14.606	14.4596	14.8354	95.7121
3	A_3	5.646	4.904	5.020	5.130	5.044	5.2446	5.1139	32.9928
4	A_4	3.143	3.123	3.352	3.134	3.139	3.1413	3.1305	20.1967
5	A_5	8.759	8.341	8.631	8.591	8.543	8.4541	8.4037	54.2173
6	A_6	3.758	3.418	3.432	3.377	3.367	3.3567	3.3315	21.4935
7	A_7	2.502	2.498	2.499	2.500	2.497	2.4947	2.4968	16.1084
Best weight (lb)		33,481.2	33,263.9	33,248.9	33,287.9	33,251.9	33,256.2	33,256.48	15,084.89 (kg)
Average weight (lb)		N/A	N/A	N/A	N/A	N/A	N/A	33,280.85	15,095.94 (kg)
Number of analyses		150,000	32,600	10,000	10,000	7,000	6,000	18,300	

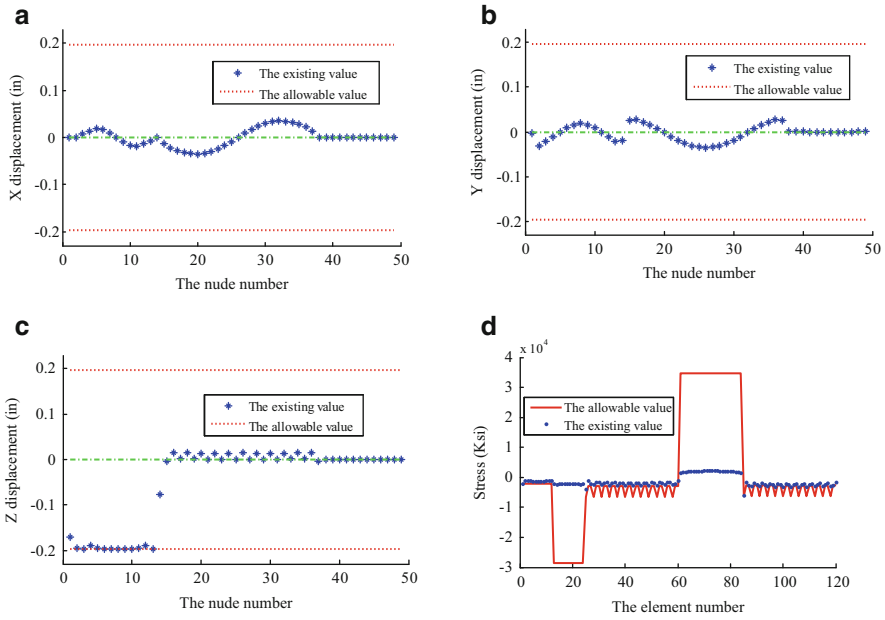

Fig. 8.25 Comparison of the allowable and existing constraints for the 120-bar dome shaped truss using the IRO [3]: (**a**) Displacement in the x direction, (**b**) Displacement in the y direction, (**c**) Displacement in the z direction, (**d**) Stresses

References

1. Kaveh A, Khayatazad M (2012) A new meta-heuristic method: ray optimization. Comput Struct 112–113:283–294
2. Kaveh A, Khayatazad M (2013) Ray optimization for size and shape optimization of truss structures. Comput Struct 117:82–94
3. Kaveh A, Ilchi Ghazaan M, Bakhshpoori T (2013) An improved ray optimization algorithm for design of truss structures. Period Polytech 57(2):97–112
4. Laval PB (2003) Mathematics for computer graphics-ray tracing III. Kennesaw-MATH–4490, Kennesaw State University, Kennesaw, GA
5. Kaveh A, Talatahari S (2009) Particle swarm optimizer, ant colony strategy and harmony search scheme hybridized for optimization of truss structures. Comput Struct 87:267–283
6. Tsoulos IG (2008) Modifications of real code genetic algorithm for global optimization. Appl Math Comput 203:598–607
7. Belegundu AD (1982) A study of mathematical programming methods for structural optimization. Ph.D. thesis, Department of Civil and Environmental Engineering, University of Iowa, Iowa, IA
8. Arora JS (1989) Introduction to optimum design. McGraw-Hill, New York, NY
9. Ragsdell KM, Phillips DT (1976) Optimal design of a class of welded structures using geometric programming. ASME J Eng Ind Ser B 98:1021–1925
10. Deb K (1991) Optimal design of a welded beam via genetic algorithms. AIAA J 29:2013–2015
11. Gil L, Andreu A (2003) Shape and cross-section optimization of a truss structure. Comput Struct 79:681–689

12. Wang D, Zhang WH, Jiang JS (2004) Truss optimization on shape and sizing with frequency constraints. AIAA J 42:1452–1456
13. Lingyun W, Mei Z, Guangming W, Guang M (2005) Truss optimization on shape and sizing with frequency constraints based on genetic algorithm. J Comput Mech 25:361–368
14. Gomes HM (2011) Truss optimization with dynamic constraints using a particle swarm algorithm. Expert Syst Appl 38:957–968
15. American Institute of Steel Construction, AISC (1989) Manual of steel construction allowable stress design, 9th edn. American Institute of Steel Construction, Chicago, IL
16. Lee KS, Geem ZW (2004) A new structural optimization method based on the harmony search algorithm. Comput Struct 82:781–798
17. Kaveh A, Talatahari S (2010) Optimal design of skeletal structures via the charged system search algorithm. Struct Multidiscip Optim 41:893–911
18. Camp CV (2007) Design of space trusses using Big Bang–Big Crunch optimization. J Struct Eng ASCE 133:999–1008
19. Camp CV, Bichon J (2004) Design of space trusses using ant colony optimization. J Struct Eng ASCE 130:741–751
20. Kaveh A, Talatahari S (2009) Size optimization of space trusses using Big Bang-Big Crunch algorithm. Comput Struct 87:1129–1140
21. Kaveh A, Talatahari S (2010) Optimum design of skeletal structures using imperialist competitive algorithm. Comput Struct 88:1220–1229
22. Kaveh A, Talatahari S (2009) A particle swarm ant colony optimization for truss structures with discrete variables. J Construct Steel Res 65:1558–1568

Chapter 9
Modified Big Bang–Big Crunch Algorithm

9.1 Introduction

The Big Bang–Big Crunch (BB–BC) method developed by Erol and Eksin [1] consists of two phases: a Big Bang phase, and a Big Crunch phase. In the Big Bang phase, candidate solutions are randomly distributed over the search space. Similar to other evolutionary algorithms, initial solutions are spread all over the search space in a uniform manner in the first Big Bang. Erol and Eksin [1] associated the random nature of the Big Bang to energy dissipation or the transformation from an ordered state (a convergent solution) to a disorder or chaos state (new set of solution candidates).

This chapter consists of two parts. In the first part the developed Modified Big Bang–Big Crunch (MBB–BC) optimization algorithm is employed for optimal design of truss structures [2]. In the second part optimal design of the Schwedler and ribbed domes is performed [3].

9.2 Modified BB-BC Method

9.2.1 Introduction to BB–BC Method

The BB–BC method developed by Erol and Eksin [1] consists of two phases: a Big Bang phase, and a Big Crunch phase. In the Big Bang phase, candidate solutions are randomly distributed over the search space. Similar to other evolutionary algorithms, initial solutions are spread all over the search space in a uniform manner in the first Big Bang. Erol and Eksin [1] associated the random nature of the Big Bang to energy dissipation or the transformation from an ordered state (a convergent solution) to a disorder or chaos state (new set of solution candidates).

The Big Bang phase is followed by the Big Crunch phase. The Big Crunch is a convergence operator that has many inputs but only one output, which is named as

A. Kaveh, *Advances in Metaheuristic Algorithms for Optimal Design of Structures*,
DOI 10.1007/978-3-319-05549-7_9, © Springer International Publishing Switzerland 2014

the "*center of mass*", since the only output has been derived by calculating the center of mass. Here, the term mass refers to the inverse of the merit function value. The point representing the center of mass that is denoted by $A_i^{c(k)}$ is calculated according to:

$$A_i^{c(k)} = \frac{\sum_{j=1}^{N} \frac{1}{Mer^j} \cdot A_i^{(k,j)}}{\sum_{j=1}^{N} \frac{1}{Mer^j}} \quad i = 1, 2, \ldots, ng \tag{9.1}$$

where $A_i^{(k,j)}$ is the ith component of the jth solution generated in the kth iteration; N is the population size in Big Bang phase. After the Big Crunch phase, the algorithm creates the new solutions to be used as the Big Bang of the next iteration step, by using the previous knowledge (center of mass). This can be accomplished by spreading new off-springs around the center of mass using a normal distribution operation in every direction, where the standard deviation of this normal distribution function decreases as the number of iterations of the algorithm increases:

$$A_i^{(k+1,j)} = A_i^{c(k)} + \frac{r_j \alpha_1 (A_{\max} - A_{\min})}{k+1} \quad i = 1, 2, \ldots, ng \tag{9.2}$$

where r_j is a random number from a standard normal distribution which changes for each candidate, and α_1 is a parameter for limiting the size of the search space.

These successive explosion and contraction steps are carried out repeatedly until a stopping criterion has been met. A maximum number of iterations is utilized as a stopping criterion.

BB–BC does not require an explicit relationship between the objective function and constraints. Instead, the objective function for a set of design variables can be penalized to reflect any violation of the design constraints. In utilizing the penalty functions, if the constraints are between the allowable limits, the penalty will be zero; otherwise, the amount of penalty is obtained by dividing the violation of allowable limit to the limit itself. After analyzing a structure, the deflection of each node and the stress in each member are obtained. These values are compared to the allowable limits to calculate the penalty functions as:

$$\begin{cases} \sigma_i^{\min} < \sigma_i < \sigma_i^{\max} & \Rightarrow \Phi_\sigma^{(i)} = 0 \\ \sigma_i^{\min} > \sigma_i \ \text{ or } \ \sigma_i^{\max} < \sigma_i & \Rightarrow \Phi_\sigma^{(i)} = \dfrac{\sigma_i - \sigma_i^{\min/\max}}{\sigma_i^{\min/\max}} \end{cases} \qquad i = 1, 2, \ldots, n \tag{9.3}$$

$$\begin{cases} \sigma_b < \sigma_i < 0 & \Rightarrow \Phi_{\sigma b}^{(i)} = 0 \\ \sigma_i < 0 \quad \wedge \quad \sigma_i < \sigma_b & \Rightarrow \Phi_{\sigma b}^{(i)} = \dfrac{\sigma_i - \sigma_b}{\sigma_b} \end{cases} \qquad i = 1, 2, \ldots, ns \tag{9.4}$$

$$\begin{cases} \delta_i^{\min} < \delta_i < \delta_i^{\max} & \Rightarrow \Phi_{\delta}^{(i)} = 0 \\ \delta_i^{\min} > \delta_i \quad \text{or} \quad \delta_i^{\max} < \delta_i & \Rightarrow \Phi_{\delta}^{(i)} = \dfrac{\delta_i - \delta_i^{\min/\max}}{\delta_i^{\min/\max}} \end{cases} \qquad i = 1, 2, \ldots, m \tag{9.5}$$

In optimizing structures, the main objective is to find the minimum amount of the merit function. This function is defined as [4]:

$$Mer^k = \varepsilon_1 \cdot W^k + \varepsilon_2 \cdot \left(\Phi_{\sigma}^k + \Phi_{\delta}^k + \Phi_{\sigma b}^k\right)^{\varepsilon_3} \tag{9.6}$$

Mer^k is the merit function for the kth candidate; ε_1, ε_2 and ε_3 are coefficients of merit function. Φ_{σ}^k, Φ_{δ}^k and $\Phi_{\sigma b}^k$ is the summation of stress penalties, summation of nodal deflection penalties and summation of buckling stress penalties for candidate k, respectively.

For multiple loadings, after analyzing the structure and determining the penalty functions for each component of the load, the total penalty function is calculated through addition of penalty functions of stress, buckling stress for each member, and deflection for each node, as:

$$Mer^k = \varepsilon_1 \cdot W^k + \varepsilon_2 \cdot \sum_{i=1}^{np} \left(\Phi_{\sigma(i)}^k + \Phi_{\delta(i)}^k + \Phi_{\sigma b(i)}^k\right)^{\varepsilon_3} \tag{9.7}$$

where np is the number of multiple loadings. In this part, for a better control on other parameters, ε_1 is set to 1. The coefficient ε_2 is taken as the weight of the structure and the coefficient ε_3 is set in a way that the penalties decrease. The cross-sectional areas can also be reduced. Therefore, in the first iterations of the search process, ε_3 is set to 1.5 but gradually it is increased to 3 [4].

The pseudo-code of the BB–BC algorithm can be summarized as follows:

Step 1: Generate initial candidates in a random manner (considering allowable boundaries).

Step 2: Calculate the merit function values of all the candidate solutions [Eqs. (9.7) and (9.8)].

Step 3: Find the center of the mass (Eq. 9.2).

Step 4: Calculate new candidates around the center of the mass (Eq. 9.3).

Step 5: Return to Step 2 and repeat the process until the condition for the stopping criterion is fulfilled.

9.2.2 A Modified BB–BC Algorithm

The advantages of applying BB–BC algorithm for structural design are similar to other evolutionary algorithms. BB–BC is a multi-agent and randomized search technique that in each cycle, a number of search space points are tested. The random selection and the information obtained in each cycle (center of mass) are used to choose new points in subsequent cycles. The BB–BC method has the ability to handle a mixture of discrete and continuous design variables and multiple loading cases.

Although BB–BC performs well in the exploitation (the fine search around a local optimum), there are some problems in the exploration (global investigation of the search place) stage. If all of the candidates in the initial Big Bang are collected in a small part of search space, the BB–BC method may not find the optimum solution and with a high probability, it may be trapped in that subdomain. One can consider a large number for candidates to avoid this defect, but it causes an increase in the function evaluations as well as the computational costs. This chapter uses the Particle Swarm Optimization (PSO) capacities to improve the exploration ability of the BB–BC algorithm.

The Particle Swarm Optimization is motivated from the social behavior of bird flocking and fish schooling which has a population of individuals, called particles, that adjust their movements depending on both their own experience and the population's experience [5]. At each iteration, a particle moves towards a direction computed from the best visited position (local best) and the best visited position of all particles in its neighborhood (global best). The modified BB–BC approach similarly not only uses the center of mass but also utilizes the best position of each candidate ($A_i^{lbest(k,j)}$) and the best global position ($A_i^{gbest(k)}$) to generate a new solution, as:

$$A_i^{(k+1,j)} = \alpha_2 A_i^{c(k)} + (1-\alpha_2)\left(\alpha_3 A_i^{gbest(k)} + (1-\alpha_3)A_i^{lbest(k,j)}\right) + \frac{r_j \alpha_1 (A_{\max} - A_{\min})}{k+1} \quad \begin{cases} i = 1,2,\ldots,ng \\ j = 1,2,\ldots,N \end{cases} \tag{9.8}$$

where $A_i^{lbest(k,j)}$ is the best position of the jth particle up to the iteration k and $A_i^{gbest(k)}$ is the best position among all candidates up to the iteration k; α_2 and α_3 are adjustable parameters controlling the influence of the global best and local best on the new position of the candidates, respectively.

Another improvement in the BB–BC method is employing Sub-Optimization Mechanism (SOM) as an auxiliary tool which works as a search-space updating mechanism. SOM, based on the principles of finite element method, was introduced by Kaveh et al. [4, 6]. Similar to the finite element method which requires dividing of the problem domain into many subdomains and using these patches instead of the

main domain, SOM divides the search space into sub-domains and performs optimization process into these patches, and then based on the resulted solutions the undesirable parts are deleted, and the remaining space is divided into smaller parts for more investigation in the next stage. This process continues until the remaining space becomes less than the required size to satisfy accuracy.

This mechanism can be considered as the repetition of the following steps for definite times, nc, (in the stage k of the repetition) [4, 6]:

Step 1: Calculating cross-sectional area bounds for each group. If $A_i^{gbest(k_{SOM}-1)}$ is the global best solution obtained from the previous stage ($k_{SOM} - 1$) for design variable i, then:

$$\begin{cases} A_{\min,i}^{(k_{SOM})} = A_i^{gbest(k_{SOM}-1)} - \beta_1 \cdot \left(A_{\max,i}^{(k_{SOM}-1)} - A_{\min,i}^{(k_{SOM}-1)} \right) \geq A_{\min,i}^{(k_{SOM}-1)} \\ A_{\max,i}^{(k_{SOM})} = A_i^{gbest(k_{SOM}-1)} + \beta_1 \cdot \left(A_{\max,i}^{(k_{SOM}-1)} - A_{\min,i}^{(k_{SOM}-1)} \right) \leq A_{\max,i}^{(k_{SOM}-1)} \end{cases} \quad \begin{matrix} i = 1,2,\ldots,ng \\ k_{SOM} = 2,\ldots,nc \end{matrix} \tag{9.9}$$

where β_1 is an adjustable factor which determines the amount of the remaining search space and in this research it is taken as 0.3 [6]; $A_{\min,i}^{(k_{SOM})}$ and $A_{\max,i}^{(k_{SOM})}$ are the minimum and the maximum allowable cross-sectional areas at the stage k_{SOM}, respectively. In stage 1, the amounts of $A_{\min,i}^{(1)}$ and $A_{\max,i}^{(1)}$ are set to:

$$A_{\min,i}^{(1)} = A_{\min}, A_{\max,i}^{(1)} = A_{\max} \quad i = 1, 2, \ldots, ng \tag{9.10}$$

Step 2: Determining the amount of increase in allowable cross-sectional areas. In each stage, the number of permissible value for each group is considered as β_2, and therefore the amount of the accuracy rate of each variable is equal to:

$$A_i^{*(k_{SOM})} = \frac{\left(A_{\max,i}^{(k_{SOM})} - A_{\min,i}^{(k_{SOM})} \right)}{\beta_2 - 1} \quad i = 1, 2, \ldots, ng \tag{9.11}$$

where $A_i^{*(k_{SOM})}$ is the amount of increase in allowable cross-sectional area; Unlike ACO, β_2 (the number of subdomains) does no affect the optimization time and in the BB–BC optimization, β_2 is set to 100.

Step 3: Creating the series of the allowable cross-sectional areas. The set of allowable cross-sectional areas for group i can be defined as:

$$\begin{aligned} &A_{\min,i}^{(k_{SOM})}, A_{\min,i}^{(k_{SOM})} + A_i^{*(k_{SOM})}, \ldots, A_{\min,i}^{(k_{SOM})} + (\beta_2 - 1) \cdot A_i^{*(k_{SOM})} \\ &= A_{\min,i}^{(k_{SOM})} \quad i = 1, 2, \ldots, ng \end{aligned} \tag{9.12}$$

Step 4: Determining the optimum solution of the stage k_{SOM}. The last step is performing an optimization process using the BB–BC algorithm.

The stopping creation for SOM can be described as:

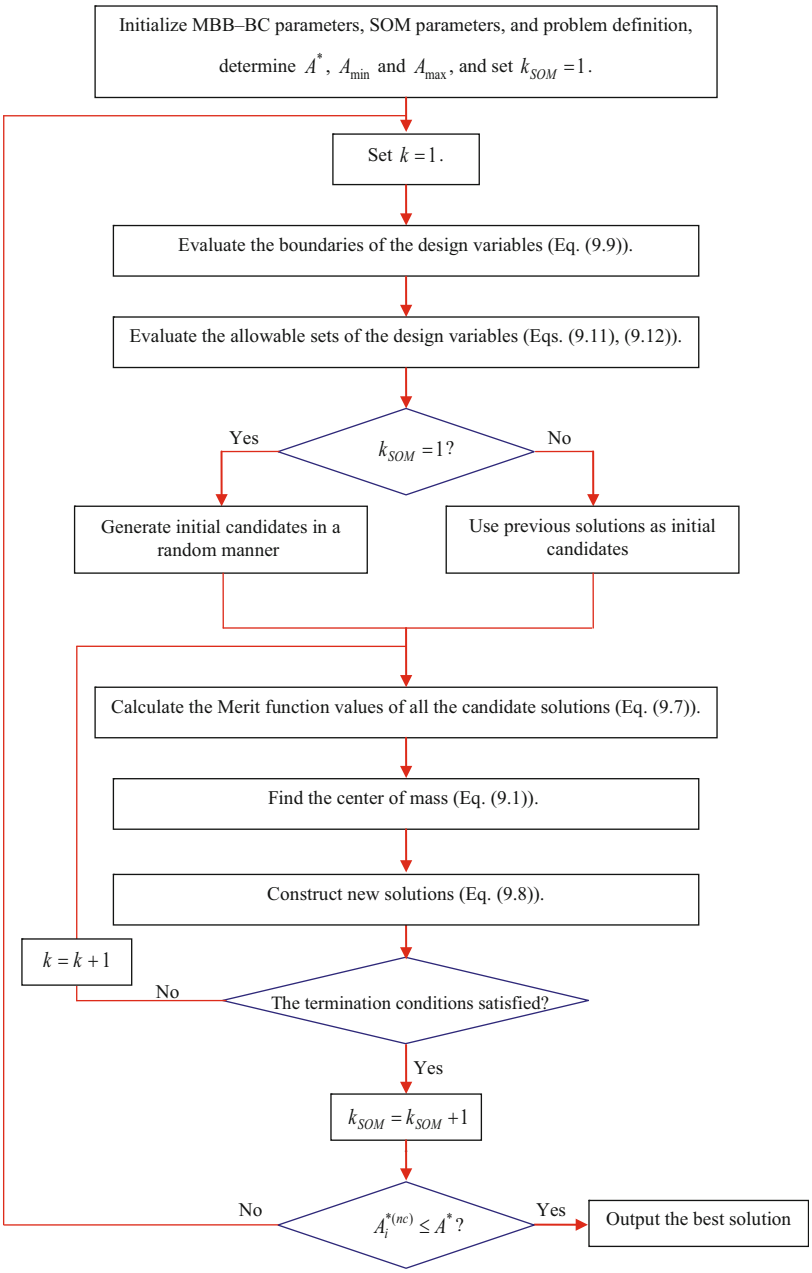

Fig. 9.1 The flowchart for the MBB–BC algorithm [2]

$$A_i^{*(nc)} \leq A^* \quad i = 1, 2, \ldots, ng \tag{9.13}$$

where $A_i^{*(nc)}$ is the amount of accuracy rate of the last stage; and A^* is the amount of accuracy rate of the primary problem.

Sub-Optimization Mechanism continues the search process until a solution is obtained with the required accuracy. SOM performs as a search-space updating rule which improves the search process with updating the search space from one stage to the next stage. Also, SOM helps distribute the initial particles in the first Big Bang. Another advantage of SOM is to select a small number of candidates because of reducing the search space. The MBB–BC procedure is illustrated in Fig. 9.1.

9.3 Size Optimization of Space Trusses Using a MBB–BC Algorithm

9.3.1 Formulation

Truss optimization is one of the most active branches of the structural optimization. Size optimization of truss structures involves determining optimum values for member cross-sectional areas, A_i, that minimize the structural weight W. This minimum design should also satisfy the inequality constraints that limit design variable sizes and structural responses. The optimal design of a truss can be formulated as:

$$\begin{aligned} \text{minimize} \quad & W(\{x\}) = \sum_{i=1}^{n} \gamma_i \cdot A_i \cdot L_i \\ \text{subject to :} \quad & \delta_{\min} \leq \delta_i \leq \delta_{\max} \qquad i = 1, 2, \ldots, m \\ & \sigma_{\min} \leq \sigma_i \leq \sigma_{\max} \qquad i = 1, 2, \ldots, n \\ & \sigma_i^b \leq \sigma_i \leq 0 \qquad i = 1, 2, \ldots, ns \\ & A_{\min} \leq A_i \leq A_{\max} \qquad i = 1, 2, \ldots, ng \end{aligned} \tag{9.14}$$

where $W(\{x\})$ is the weight of the structure; n is the number of members making up the structure; m is the number of nodes; ns is the number of compression elements; ng is the number of groups (number of design variables); γ_i is the material density of member i; L_i is the length of member i; A_i is the cross-sectional area of member i chosen between $A_{\min}$ and $A_{\max}$; min is the lower bound and max = upper bound; σ_i and δ_i is the the stress and nodal deflection, respectively; σ_i^b is the allowable buckling stress in member i when it is in compression.

In this part, the MBB–BC is implemented to solve the truss optimization problems. The MBB–BC method consists of two phases: a Big Bang phase where candidate solutions are randomly distributed over the search space, and a Big Crunch phase working as a convergence operator where the center of mass is

generated. Then new solutions are created by using the center of mass to be used as the next Big Bang. These successive phases are carried repeatedly until a stopping criterion has been met. This algorithm not only considers the center of mass as the average point in the beginning of each Big Bang, but also similar to Particle Swarm Optimization-based approaches [5], utilizes the best position of each particle and the best visited position of all particles. As a result because of increasing the exploration of the algorithm, the performance of the BB–BC approach is improved. Another reformation is to use Sub-Optimization Mechanism (SOM), introduced by Kaveh et al. [4, 6] for ant colony approaches. SOM is based on the principles of finite element method working as a search-space updating technique. Some changes are made to prepare SOM for the MBB–BC algorithm. Numerical simulation based on the MBB–BC method including medium- and large-scaled trusses and comparisons with results obtained by other heuristic approaches demonstrate the effectiveness of the present algorithm.

9.3.2 Design Examples

In this section, five truss structures are optimized utilizing the present method. Then the final results are compared to the solutions of other advanced heuristic methods to demonstrate the efficiency of this work. These optimization examples include:

- A 25-bar spatial truss structure;
- A 72-bar spatial truss structure;
- A 120-bar dome shaped truss;
- A square on diagonal double-layer grid;
- A 26-story-tower spatial truss.

For the proposed algorithm, a population of 50 individuals is used for the first through third examples and a population of 100 candidates is selected for two last examples. A^* for all the examples is selected as 0.01. The algorithms are coded in Matlab and the structures are analyzed using the direct stiffness method.

In order to investigate the effect of the initial solution on the final result and because of the stochastic nature of the algorithm, each example is independently solved several times. The initial population in each of these runs is generated in a random manner. Last example is optimized by the MBB–BC optimization for 20 times, while performance comparisons of the MBB–BC method in other examples based on 50 evaluations. The performance of the present algorithm in the first example is compared to the simple and improved heuristic approaches, it is compared to the simple heuristic algorithms in the second example and in the third example, some improved approaches are considered from literature. Example 4 is optimized using GA, PSO, a hybrid PSO (PSOPC [7]), BB–BC and the MBB–BC method. Last example which has 942 members is solved by the present algorithm and the results are compared to those of GA, PSO and the BB–BC method.

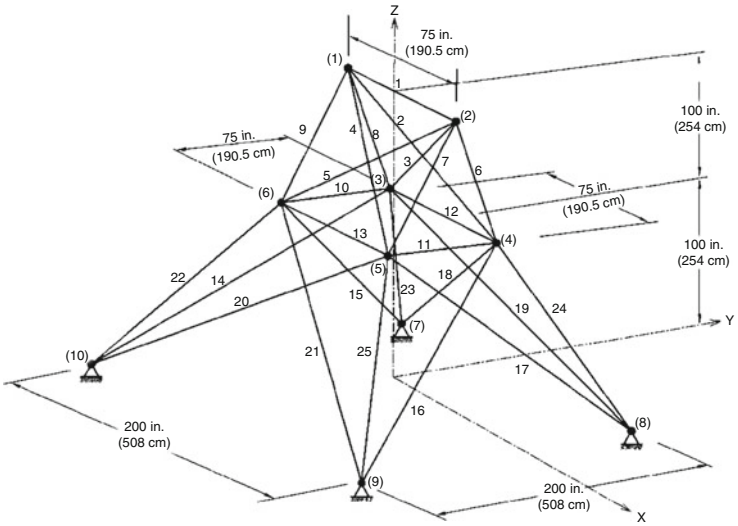

Fig. 9.2 Schematic of a twenty five-bar spatial truss

9.3.2.1 Twenty Five-Bar Spatial Truss

The topology and nodal numbers of a 25-bar spatial truss structure are shown in Fig. 9.2. In this example, designs for a multiple load case are performed and the results are compared to those of other optimization techniques employed by [8–13]. In these studies, the material density is considered as 0.1 lb/in^3 (2,767.990 kg/m^3) and the modulus of elasticity is taken as 10,000 ksi (68,950 MPa).

Twenty five members are categorized into eight groups, as follows:

(1) A_1, (2) A_2–A_5, (3) A_6–A_9, (4) A_{10}–A_{11}, (5) A_{12}–A_{13}, (6) A_{14}–A_{17}, (7) A_{18}–A_{21}, and (8) A_{22}–A_{25}.

This spatial truss is subjected to two loading conditions shown in Table 9.1. Maximum displacement limitations of ±0.35 in (8.89 mm) are imposed on every node in every direction and the axial stress constraints vary for each group as shown in Table 9.2. The range of cross-sectional areas varies from 0.01 to 3.4 in^2 (0.6452 cm^2 to 21.94 cm^2).

Using $\alpha_1 = 1.0$ allows an initial search of the full range of values for each design variable. Figure 9.3 shows the effect of various values for α_2 and α_3 on the average weight of designs for the 25-bar truss. This figure shows that $\alpha_2 = 0.40$ and $\alpha_3 = 0.80$ are suitable values for MBB–BC algorithm. These parameter values are used for all other examples presented.

The MBB–BC algorithm achieves the best solution after 12,500 searches. However, the BB–BC algorithm finds the best solution after about 20,566 analyses [14] which is 64 % more than the present work. The best weight of the MBB–BC is 545.16 lb while the best result of BB–BC is 545.38 lb. In addition, the MBB–BC

Table 9.1 Loading conditions for the 25-bar spatial truss

	Case 1			Case 2		
Node	P_X kips (kN)	P_Y kips (kN)	P_Z kips (kN)	P_X kips (kN)	P_Y kips (kN)	P_Z kips (kN)
1	0.0	20.0 (89)	−5.0 (22.25)	1.0 (4.45)	10.0 (44.5)	−5.0 (22.25)
2	0.0	−20.0 (89)	−5.0 (22.25)	0.0	10.0 (44.5)	−5.0 (22.25)
3	0.0	0.0	0.0	0.5 (2.22)	0.0	0.0
6	0.0	0.0	0.0	0.5 (2.22)	0.0	0.0

Table 9.2 Member stress limitation for the 25-bar spatial truss

Element group		Compressive stress limitations ksi (MPa)	Tensile stress limitations ksi (MPa)
1	A_1	35.092 (241.96)	40.0 (275.80)
2	A_2–A_5	11.590 (79.913)	40.0 (275.80)
3	A_6–A_9	17.305 (119.31)	40.0 (275.80)
4	A_{10}–A_{11}	35.092 (241.96)	40.0 (275.80)
5	A_{12}–A_{13}	35.092 (241.96)	40.0 (275.80)
6	A_{14}–A_{17}	6.759 (46.603)	40.0 (275.80)
7	A_{18}–A_{21}	6.959 (47.982)	40.0 (275.80)
8	A_{22}–A_{25}	11.082 (76.410)	40.0 (275.80)

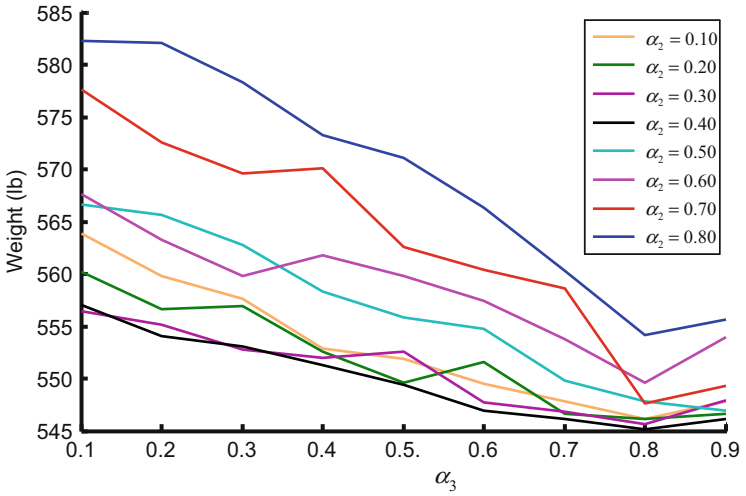

Fig. 9.3 Effect of MBB–BC parameters on average weight of the 25-bar truss

algorithm has better performance than the BB–BC algorithm with respect to the average weight and standard deviation. Although the MBB–BC approach has worse performance than the improved methods (IACS [4] and PSACO [15] and HPSACO [16]), it performs better than other simple algorithms (GA [8], PSO [17]) when the best weight, the average weight or the standard deviation are compared. Also, the MBB–BC approach has smaller required number of iterations for convergence than

Table 9.3 Performance comparison for the 25-bar spatial truss

	Optimal cross-sectional areas (in^2)								
	Rajeev and Krishnamoorthy	Schutte and Groenwold	Lee and Geem	Kaveh et al.	Kaveh and Talatahar		Camp	Present work [2]	
Element group	GA [8]	PSO [17]	HS [18]	IACS [4]	PSACO [15]	HPSACO [16]	BB–BC [14]	in^2	cm^2
1 A_1	0.10	0.010	0.047	0.010	0.010	0.010	0.010	0.010	0.065
2 A_2–A_5	1.80	2.121	2.022	2.042	2.052	2.054	2.092	1.993	12.856
3 A_6–A_9	2.30	2.893	2.950	3.001	3.001	3.008	2.964	3.056	19.717
4 A_{10}–A_{11}	0.20	0.010	0.010	0.010	0.010	0.010	0.010	0.010	0.065
5 A_{12}–A_{13}	0.10	0.010	0.014	0.010	0.010	0.010	0.010	0.010	0.065
6 A_{14}–A_{17}	0.80	0.671	0.688	0.684	0.684	0.679	0.689	0.665	4.293
7 A_{18}–A_{21}	1.80	1.611	1.657	1.625	1.616	1.611	1.601	1.642	10.594
8 A_{22}–A_{25}	3.0	2.717	2.663	2.672	2.673	2.678	2.686	2.679	17.281
Best weight (Ib)	546	545.21	544.38	545.03	545.04	544.99	545.38	545.16	2425N
Average weight (Ib)	N/A	546.84	N/A	545.74	N/A	545.52	545.78	545.66	
Std dev	N/A	1.478	N/A	0.620	N/A	0.315	0.491	0.367	
No. of analyses	N/A		9,596	15,000	3,520	28,850	9,875	20,566	12,500

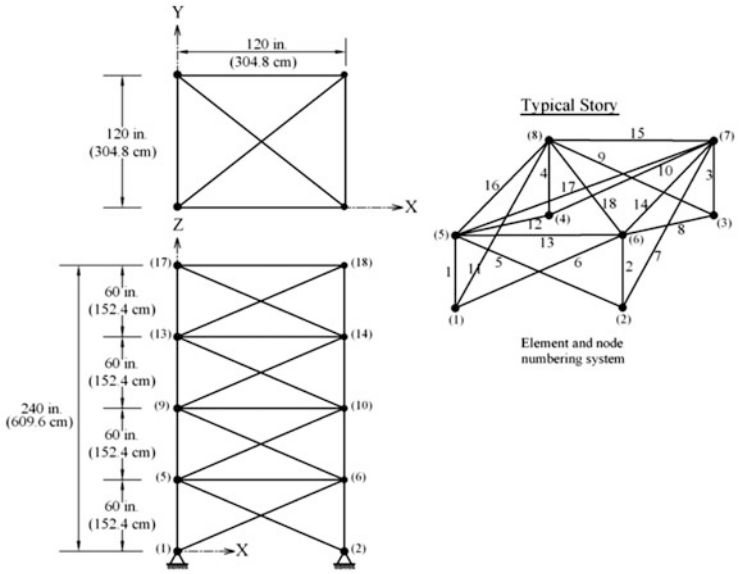

Fig. 9.4 Schematic of a 72-bar spatial truss

PSACO and HS [18]. Table 9.3 presents a comparison of the performance of the MBB–BC method and other heuristic algorithms.

9.3.2.2 Seventy Two-Bar Spatial Truss

For the 72-bar spatial truss structure shown in Fig. 9.4, the material density is 0.1 lb/in^3 (2,767.990 kg/m^3) and the modulus of elasticity is 10,000 ksi (68,950 MPa).

The members are subjected to the stress limits of ±25 ksi (±172.375 MPa). The uppermost nodes are subjected to the displacement limits of ±0.25 in (0.635) in both the x and y directions. The 72 structural members of this spatial truss are sorted into 16 groups using symmetry: (1) A_1–A_4, (2) A_5–A_{12}, (3) A_{13}–A_{16}, (4) A_{17}–A_{18}, (5) A_{19}–A_{22}, (6), A_{23}–A_{30}, (7) A_{31}–A_{34}, (8) A_{35}–A_{36}, (9) A_{37}–A_{40}, (10) A_{41}–A_{48}, (11) A_{49}–A_{52}, (12) A_{53}–A_{54}, (13) A_{55}–A_{58}, (14) A_{59}–A_{66} (15), A_{67}–A_{70}, (16) A_{71}–A_{72}. The minimum permitted cross-sectional area of each member is 0.10 in^2 (0.6452 cm^2), and the maximum cross-sectional area of each member is 4.00 in^2 (25.81 cm^2). Table 9.4 lists the values and directions of the two load cases applied to the 72-bar spatial truss.

The best weight of the MBB–BC optimization is 379.66 lb, while it is 379.85 lb, 380.24 lb, 381.91 and 385.76 lb for the BB–BC [14], ACO [19], PSO [20] and GA [21], respectively. Standard deviation in the MBB–BC is 1.201 lb while standard deviation of primary BB–BC algorithm has been reported 1.912 lb [14]. In addition, the required analyses for reaching a convergence is 13,200 analyses, which is 48 % and 40 % less than the BB–BC method and ACO, respectively. Table 9.5 compares

Table 9.4 Loading conditions for the 72-bar spatial truss

	Case 1			Case 2		
Node	P_X kips (kN)	P_Y kips (kN)	P_Z kips (kN)	P_X	P_Y	P_Z kips (kN)
17	5.0 (22.25)	5.0 (22.25)	−5.0 (22.25)	0.0	0.0	−5.0 (22.25)
18	0.0	0.0	0.0	0.0	0.0	−5.0 (22.25)
19	0.0	0.0	0.0	0.0	0.0	−5.0 (22.25)
20	0.0	0.0	0.0	0.0	0.0	−5.0 (22.25)

Table 9.5 Performance comparison for the 72-bar spatial truss

		Optimal cross-sectional areas (in^2)					
Element group		Erbatur et al.	Camp and Bichon	Perez and Behdinan	Camp	Kaveh and Talatahari [2]	
		GA [21]	ACO [19]	PSO [20]	BB–BC [14]	in^2	cm^2
1	A_1–A_4	1.755	1.948	1.7427	1.8577	1.9042	12.2851
2	A_5–A_{12}	0.505	0.508	0.5185	0.5059	0.5162	3.3303
3	A_{13}–A_{16}	0.105	0.101	0.1000	0.1000	0.1000	0.6452
4	A_{17}–A_{18}	0.155	0.102	0.1000	0.1000	0.1000	0.6452
5	A_{19}–A_{22}	1.155	1.303	1.3079	1.2476	1.2582	8.1176
6	A_{23}–A_{30}	0.585	0.511	0.5193	0.5269	0.5035	3.2488
7	A_{31}–A_{34}	0.100	0.101	0.1000	0.1000	0.1000	0.6452
8	A_{35}–A_{36}	0.100	0.100	0.1000	0.1012	0.1000	0.6452
9	A_{37}–A_{40}	0.460	0.561	0.5142	0.5209	0.5178	3.3409
10	A_{41}–A_{48}	0.530	0.492	0.5464	0.5172	0.5214	3.3639
11	A_{49}–A_{52}	0.120	0.100	0.1000	0.1004	0.1000	0.6452
12	A_{53}–A_{54}	0.165	0.107	0.1095	0.1005	0.1007	0.6497
13	A_{55}–A_{58}	0.155	0.156	0.1615	0.1565	0.1566	1.0104
14	A_{59}–A_{66}	0.535	0.550	0.5092	0.5507	0.5421	3.4973
15	A_{67}–A_{70}	0.480	0.390	0.4967	0.3922	0.4132	2.6658
16	A_{71}–A_{72}	0.520	0.592	0.5619	0.5922	0.5756	3.7133
Best weight (lb)		385.76	380.24	381.91	379.85	379.66	1,689 N
Average weight (lb)		N/A	383.16	N/A	382.08	381.85	
Std dev (lb)		N/A	3.66	N/A	1.912	1.201	
Number of analyses		N/A	18,500	N/A	19,621	13,200	

the performance of the improved BB–BC algorithm with those previously reported in the literature.

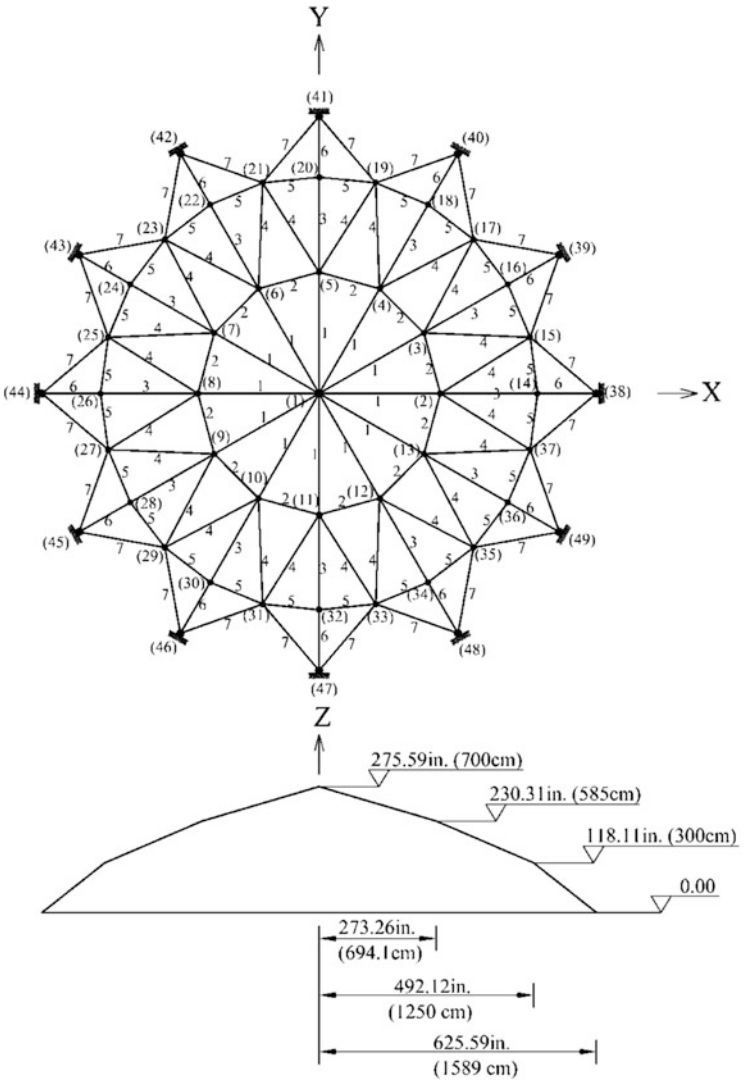

Fig. 9.5 Schematic of a 120-bar dome shaped truss

9.3.2.3 A 120-Bar Dome Truss

Figure 9.5 shows the topology and group numbering of a 120-bar dome truss. The modulus of elasticity is 30,450 ksi (210,000 MPa), and the material density is 0.288 lb/in^3 (7,971.810 kg/m^3). The yield stress of steel is taken as 58.0 ksi (400 MPa).

The dome is considered to be subjected to vertical loading at all the unsupported joints. These loads are taken as −13.49 kips (−60 kN) at node 1, −6.744 kips (−30

kN) at nodes 2 through 14, and −2.248 kips (−10 kN) at the rest of the nodes. The minimum cross-sectional area of all members is 0.775 in^2 (2 cm^2) and the maximum cross-sectional area is taken as 20.0 in^2 (129.03 cm^2). The constraints are considered as:

1. Stress constraints (according to the AISC ASD (1989) [22] code):

$$\begin{cases} \sigma_i^+ = 0.6F_y & for\ \ \sigma_i \geq 0 \\ \sigma_i^- & for\ \ \sigma_i < 0 \end{cases} \tag{9.15}$$

where σ_i^- is calculated according to the slenderness ratio:

$$\sigma_i^- = \begin{cases} \left[\left(1 - \dfrac{\lambda_i^2}{2C_C^2}\right)Fy\right] \Big/ \left(\dfrac{5}{3} + \dfrac{3\lambda_i}{8C_C} - \dfrac{\lambda_i^3}{8C_C^3}\right) & for\ \ \lambda_i < C_C \\ \dfrac{12\pi^2 E}{23\lambda_i^2} & for\ \ \lambda_i \geq C_C \end{cases} \tag{9.16}$$

where E is the modulus of elasticity; F_y is the yield stress of steel; C_c is the slenderness ratio (λ_i) dividing the elastic and inelastic buckling regions ($C_C = \sqrt{2\pi^2 E/F_y}$); λ_i is the slenderness ratio ($\lambda_i = kL_i/r_i$); k is the effective length factor; L_i is the member length; and r_i is the radius of gyration.

2. Displacement limitations of ±0.1969 in (5 mm) are imposed on all nodes in x, y and z directions.

Table 9.6 illustrates the best solution vectors, the corresponding weights and the required number for convergence in the present algorithm and some of other heuristic methods. Except IACS which uses two auxiliary mechanisms for searching, the MBB–BC optimization and HPSACO have best convergence rates.

9.3.2.4 A Square on Diagonal Double-Layer Grid

A double-layer grid of the type shown in Fig. 9.6 with a span of 21 m and the height of 1.5 m is chosen from [23]. The structure is simply supported at the corner nodes of the bottom-layer.

The loading is assumed as a uniformly distributed load on the top-layer of intensity of 155.5 kg/m^2 and it is transmitted to the joints acting as concentrated vertical loads only. The structure is assumed as pin jointed with elastic modulus of 210,000 MPa and the material density is assumed as 0.008 kg/cm^3 for all the members. Member areas are linked to maintain symmetry about the four lines of symmetry axes in the plane of the grid. Thus the problem has 47 design variables. The maximum allowable area is considered as 22 cm^2 with a lower limit of 0.1 cm^2.

Stress, Euler buckling and displacement constraints are considered in this problem. All the elements are subjected to the following stress constraints:

Table 9.6 Performance comparison for the 120-bar dome truss

Element group		Optimal cross-sectional areas (in^2)						
		Kaveh et al.	Kaveh and Talatahari				Present work [2]	
		IACS [4]	PSOPC [15]	PSACO [15]	HPSACO [16]	BB–BC	in^2	cm^2
1	A_1	3.026	3.040	3.026	3.095	3.026	3.037	19.596
2	A_2	15.06	13.149	15.222	14.405	14.276	14.431	93.010
3	A_3	4.707	5.646	4.904	5.020	4.986	5.130	33.094
4	A_4	3.100	3.143	3.123	3.352	3.175	3.134	20.217
5	A_5	8.513	8.759	8.341	8.631	8.617	8.591	55.427
6	A_6	3.694	3.758	3.418	3.432	3.558	3.377	21.785
7	A_7	2.503	2.502	2.498	2.499	2.510	2.500	16.129
Best weight (Ib)		33,320.52	33,481.2	33,263.9	33,248.9	33,340.7	33,287.9	148,064 N
No. of analyses		3,250	150,000	32,600	10,000	22,000	10,000	

$$-1,000 \leq \sigma_i \leq 1,400 \ \text{kg/cm}^2 \quad i = 1, 2, \ldots, 47 \tag{9.17}$$

where i is the element number. Tubular members are considered with a diameter to thickness ratio of 10. Thus Euler buckling is considered as:

$$\sigma_i^b = -10.1EA_i/8L_i^2 \quad i = 1, 2, \ldots, 47 \tag{9.18}$$

In addition, displacement constraints are imposed on the vertical component of the three central joints along the diagonal of the grid (joints 19, 20 and 22):

$$-1.5 \leq \delta_i \leq 1.5\,\text{cm} \quad i = 1, 2, 3 \tag{9.19}$$

This example is solved by GA, Standard PSO, PSOPC, BB–BC and the MBB–BC algorithm. The number of required iterations for the proposed algorithm is determined by using (9.13) (250 iterations in average), while it is considered as 500 iterations for other examples. The results are presented in Table 9.7.

The efficiency of the proposed algorithm in terms of the required optimization time and standard deviation is better than that of other approaches. The optimization time in the MBB–BC algorithm is 631 s while in primary BB–BC algorithm, it is 1,249 s on a core™ 2 Duo 3.0GHz CPU. Also, the MBB–BC algorithm can find the best result in comparison to other algorithms. Figure 9.7 shows the convergence rate of the best and average of 50 runs for the proposed algorithm.

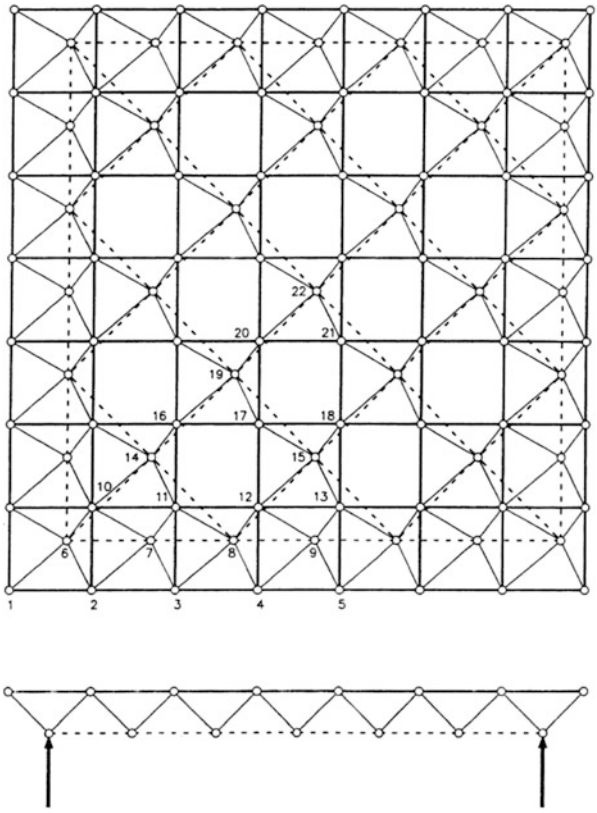

Fig. 9.6 Schematic of a square on diagonal double-layer grid [23]

9.3.2.5 A 26-Story Tower Spatial Truss

The 26-story tower space truss containing 942 elements and 244 nods is considered. Fifty nine design variables are used to represent the cross-sectional areas of 59 element groups in this structure, employing the symmetry of the structure. Figure 9.8 shows the geometry and the 59 element groups. The material density is 0.1 lb/in^3 (2767.990 kg/m^3) and the modulus of elasticity is 10,000 ksi (68,950 MPa). The members are subjected to the stress limits of $\pm$25 ksi (172.375 MPa) and the four nodes of the top level in the x, y, and z directions are subjected to the displacement limits of $\pm$15.0 in (38.10 cm) (about 1/250 of the total height of the tower).

The allowable cross-sectional areas in this example are selected from 0.1 to 20.0 in^2 (from 0.6452 cm^2 to 129.032 cm^2). The loading on the structure consists of:

1. The vertical load at each node in the first section is equal to -3 kips (-13.344 kN);
2. The vertical load at each node in the second section is equal to -6 kips (-26.688 kN);

Table 9.7 Performance comparison for the square on diagonal double-layer grid

		Optimal cross-sectional areas (cm^2)				
Group	Members	GA	PSO	PSOPC	BB–BC	Present work [2]
1	1-2	0.308854	1.791978	1.012706	0.285600	0.433754
2	2-3	10.83306	3.607462	10.32799	13.55730	5.584617
3	3-4	12.16883	7.951644	11.49081	11.91084	9.799718
4	4-5	16.45846	9.300211	10.68151	9.476018	17.07508
5	10-11	15.26131	17.51948	15.02204	13.63725	16.31362
6	11-12	17.96606	20.36895	18.01115	17.20785	19.63048
7	12-13	20.41424	21.99344	19.85809	18.81188	21.28936
8	16-17	11.12362	12.35451	11.21260	13.52788	10.02678
9	17-18	12.32299	19.71894	20.56506	15.32646	12.81294
10	20-21	13.20768	1.191691	3.287622	2.815005	9.633889
11	2-10	1.041269	14.52528	1.386787	0.572246	0.609792
12	3-11	4.161487	6.035163	0.608871	0.516033	0.572243
13	4-12	2.683208	13.56488	2.498575	0.505283	7.470955
14	11-16	2.849718	4.147840	3.987492	7.615556	0.685628
15	12-17	5.767594	0.793823	1.167498	1.022668	1.935885
16	17-20	0.816791	5.981349	1.297827	0.712039	1.237232
17	6-7	8.397544	9.386567	10.21764	13.75949	9.245048
18	7-8	3.72534	0.115224	0.922781	2.307911	0.949586
19	8-9	12.42663	10.02391	11.95824	2.470798	3.547774
20	6-14	15.29086	11.51125	14.69415	11.44199	16.15166
21	14-8	4.202762	0.924454	3.749231	1.321159	0.390444
22	8-15	1.410931	0.313266	0.564762	0.944948	5.009982
23	14-19	5.476267	14.30610	0.823906	0.731927	0.805348
24	19-15	4.34482	0.100715	0.780927	0.598549	4.229839
25	19-22	8.591895	15.97170	8.698821	8.818147	6.403876
26	6-1	7.833766	17.20812	8.625590	0.674610	6.961359
27	6-2	7.909819	4.294630	6.957233	13.27894	5.523857
28	6-10	18.56878	21.23205	19.22719	20.42001	19.36144
29	7-2	10.39279	4.382740	8.955598	3.643809	4.942896
30	7-3	4.534203	11.74380	7.007269	5.77340	7.867227
31	7-10	5.458530	5.204881	4.226522	7.61358	4.030943
32	7-11	5.847516	10.25399	4.42828	10.10760	3.746393
33	8-3	5.462611	3.141240	4.759653	3.036577	6.408331
34	8-4	10.16044	10.12301	6.047255	1.659517	3.18843
35	8-11	2.732264	2.647940	2.705861	2.513062	2.657439
36	8-12	2.957776	2.515398	7.098940	2.603133	2.932186
37	9-4	3.832699	1.520112	1.755671	1.313180	3.347062
38	9-12	10.44930	2.155439	0.299187	12.73675	6.036277
39	14-11	1.526541	1.002402	6.212577	4.481129	0.319025
40	14-16	10.24427	9.794119	11.67664	13.48525	10.07837
41	14-10	16.04097	8.867614	10.55834	3.083517	21.97723
42	15-17	0.782389	3.801597	16.12512	5.875162	0.505746
43	15-12	0.469413	12.66615	0.964569	0.115837	0.354663
44	19-17	2.830117	3.049450	5.495865	3.872755	3.969591
45	19-20	9.576797	18.10949	11.43763	10.27249	3.8124

(continued)

Table 9.7 (continued)

		Optimal cross-sectional areas (cm^2)				
Group	Members	GA	PSO	PSOPC	BB–BC	Present work [2]
46	19-16	9.393793	20.48772	6.014988	10.83278	9.327422
47	20-22	1.971953	17.67174	9.354127	14.32975	4.513447
Best weight (kg)		5236	5814	4951	4636	4413
Average weight (kg)		5614	6917	5162	4762	4508
Std dev (kg)		512.6	810.3	352.5	189.5	108.3
No. of analyses		50,000	50,000	50,000	50,000	25,000
Optimization time (s)		1,854	1,420	1,420	1,249	631

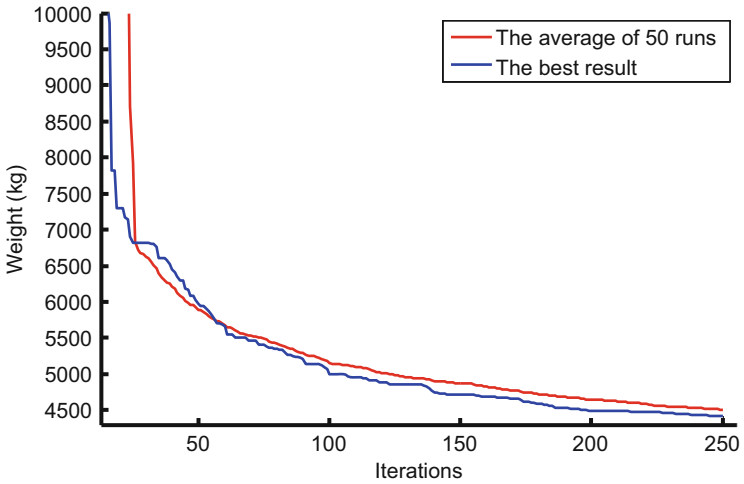

Fig. 9.7 Convergence history of the square on diagonal double-layer grid for the MBB–BC algorithm [2]

3. The vertical load at each node in the third section is equal to −9 kips (−40.032 kN);
4. The horizontal load at each node on the right side in the x direction is equal to −1 kips (−4.448 kN);
5. The horizontal load at each node on the left side in the x direction is equal to 1.5 kips (6.672 kN);
6. The horizontal load at each node on the front side in the y direction is equal to −1 kips (−4.448 kN);
7. The horizontal load at each node on the back side in the x direction is equal to 1 kips (4.448 kN).

The MBB–BC method achieved a good solution after 30,000 analyses and found an optimum weight of 52,401 lb. The best weights for the GA, Standard PSO and BB–BC are 56,343 lb, 60,385 lb and 53,201 lb, respectively. In addition, MBB–BC

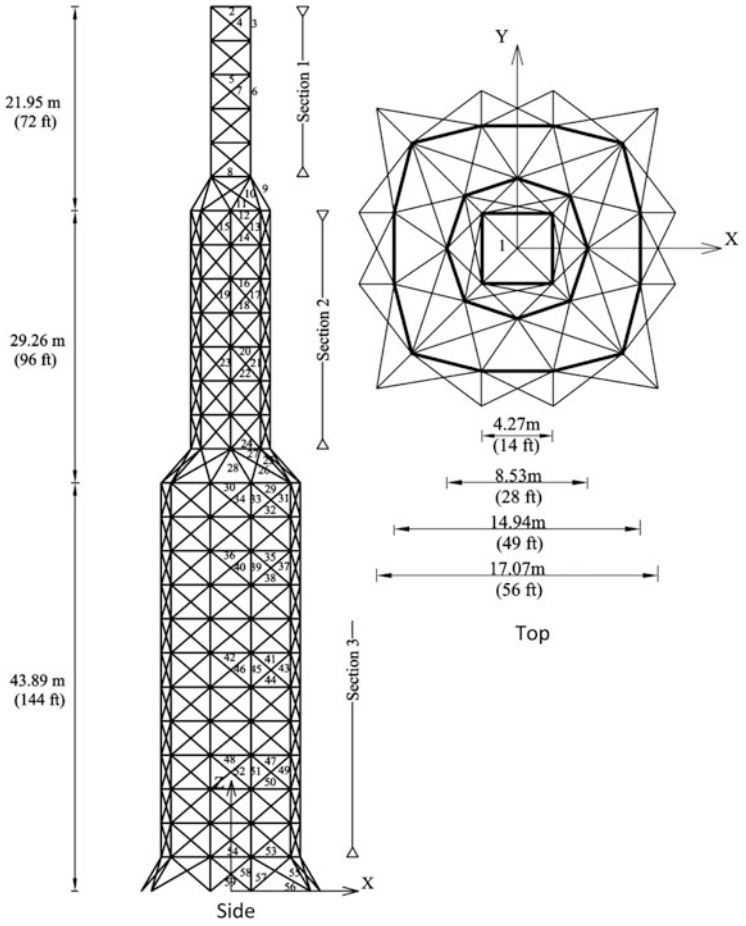

Fig. 9.8 Schematic of a 26-story-truss tower

has better performance in terms of the optimization time, standard deviation and the average weight. Table 9.8 provides the statistic information for this example.

Figure 9.9 compares the scattering of the particles for the 8th, 26th, 32nd, and 37th design variables in the 1st, 180th and 300th iterations (end of the optimization) for this example. It can be seen that particles can be given any value in the allowable space in the first iteration (Fig. 9.9a); while after 180 iterations, the particles are concentrated on a little space of search domain (Fig. 9.9b). At the end of optimization (Fig 9.9c), almost all candidates are concentrated around a specific value. Figure 9.10 shows the best and average of 20 runs convergence history for the proposed algorithm.

Table 9.8 Performance comparison for the 26-story-tower spatial truss

	GA	PSO	BB–BC	Kaveh and Talatahari [2]
Best weight (lb)	56,343	60,385	53,201	52,401 (233,081 N)
Average weight (lb)	63,223	75,242	55,206	53,532
Std dev (lb)	6,640.6	9,906.6	2,621.3	1,420.5
No. of analyses	50,000	50,000	50,000	30,000
Optimization time (s)	4,450	3,640	3,162	1,926

9.3.2.6 Discussion

The comparisons of numerical results of various trusses using the MBB–BC method with the results obtained by other heuristic approaches are performed to demonstrate the robustness of the present algorithm. With respect to the BB–BC approach, MBB–BC has better solutions and standard deviations. Also, MBB–BC has low computational time and high convergence speed compared to BB–BC. Specially, when the number of design variables increases the modified BB–BC shows better performance. By adding the PSO principle to the BB–BC algorithm, we increase the exploration by raising the search ability of the algorithm. As a result contrary to the other metaheuristic techniques which present convergence difficulty or get trapped at a local optimum in large size structures, MBB–BC performs well in large size structures. On the other hand, increasing the exploration often causes increasing the number of analyses. This problem is solved by using SOM which works as a search space updating rule and reduces the number analyses for convergence.

9.4 Optimal Design of Schwedler and Ribbed Domes Using MBB–BC Algorithm

9.4.1 Introduction

Covering large areas without intermediate supports has always been an attractive problem for architects and a challenging task for structural engineers. Dome structures are lightweight and elegant structures that provide economical solutions for covering large areas with their splendid aesthetic appearance. The joints of dome structures are considered to be rigidly connected and the members are exposed to both axial forces and bending moments. Therefore, bending moments of members affect the axial stiffness of these elements because of being slender members. Consequently, consideration of geometric nonlinearity in the analysis of these structures becomes important if the real behavior of these structures is intended to be obtained [24]. Furthermore, the instability of domes is also required to be checked during the nonlinear analysis [25, 26]. Some recent researches by Saka have shown that consideration of nonlinear behavior in the optimum design of

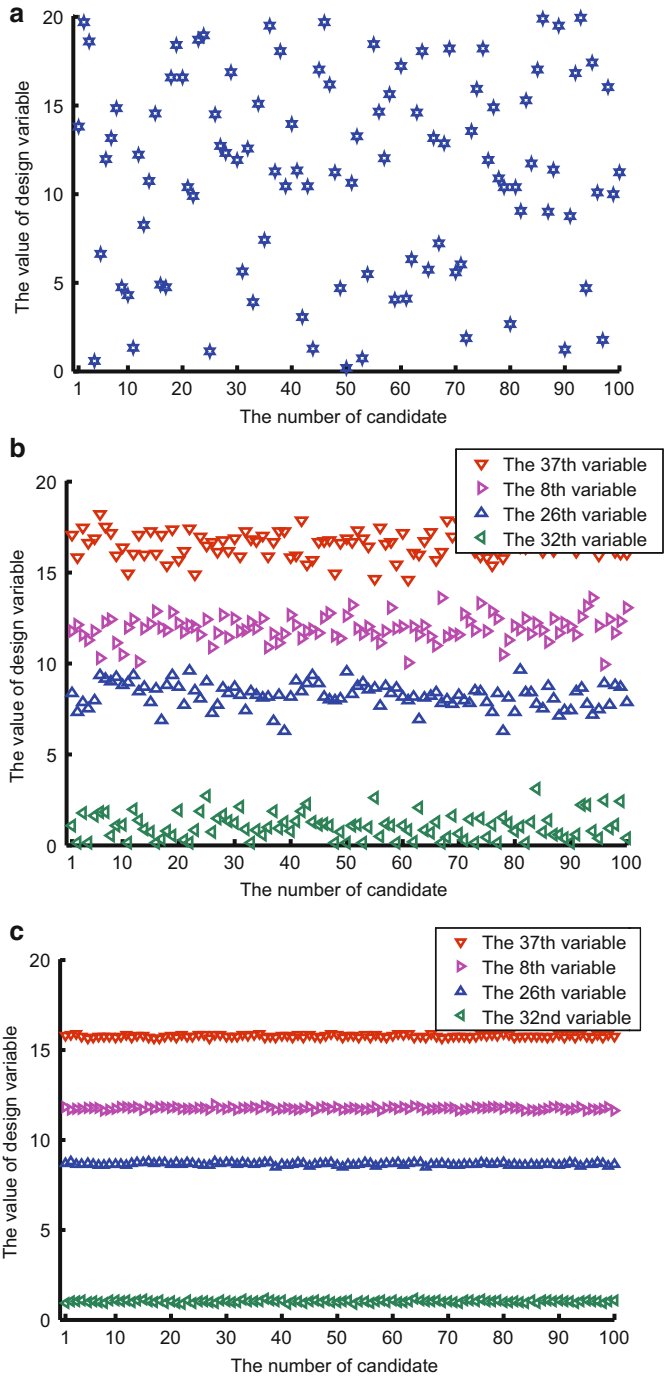

Fig. 9.9 The value of design variable (**a**) in the first iteration (for the 8th variable) (**b**) in the 180th iteration (for the 8th, 26th, 32nd, 37th and variables) (**c**) in the 300th iteration (for the 8th, 26th, 32nd, 37th and variables) [2]

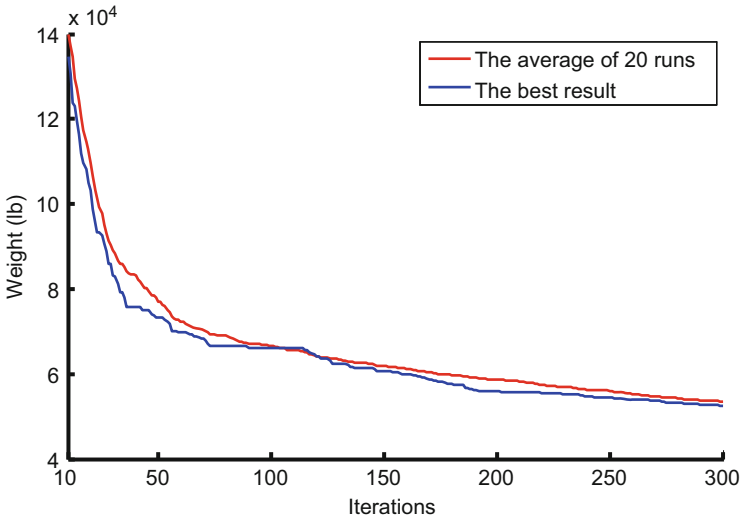

Fig. 9.10 Convergence history of the 26-story-tower truss for the MBB–BC algorithm [2]

domes does not only provide more realistic results, it also produces lighter structures [27, 28].

In this part, optimum topology design algorithm based on the MBB–BC method is developed for the *Schwedler and ribbed domes*. The algorithm determines the optimum number of rings, the optimum height of crown, and sectional designations for the members of the Schwedler domes under the external loads. Due to the selection of the number of rings as the design variable, a simple procedure is necessary to determine the dome configuration. In order to fulfill this aim, a simple methodology is introduced in here. This procedure consists of calculating the joint coordinates and the element constructions. Diagonal members are considered in the Schwedler domes to stiffen the structure. The effect of these members on the results of the optimization is investigated. The serviceability and the strength requirements are considered in the design problem as specified in LRFD–AISC [29]. The steel pipe sections list of LRFD–AISC is adopted for the cross sections of dome members and the nonlinear response of the dome is considered during the optimization process.

9.4.2 Dome Structure Optimization Problems

Optimal design of Schwedler and ribbed domes consists of finding optimal sections for elements, optimal height for the crown, and the optimum number of rings, under the determined loading conditions. The allowable cross sections are considered as 37 steel pipe sections, as shown in Table 9.9, where abbreviations ST, EST, and

Table 9.9 The allowable steel pipe sections taken from LRFD-AISC

		Nominal diameter	Weight per ft	Area	I	S	J	Z
	Type	in	(lb)	in^2	in^4	in^3	in^4	in^3
1	ST	1/2	0.85	0.250	0.017	0.041	0.082	0.059
2	EST	1/2	1.09	0.320	0.020	0.048	0.096	0.072
3	ST	3/4	1.13	0.333	0.037	0.071	0.142	0.100
4	EST	3/4	1.47	0.433	0.045	0.085	0.170	0.125
5	ST	1	1.68	0.494	0.087	0.133	0.266	0.187
6	EST	1	2.17	0.639	0.106	0.161	0.322	0.233
7	ST	$1^1/_4$	2.27	0.669	0.195	0.235	0.470	0.324
8	ST	$1^1/_2$	2.72	0.799	0.310	0.326	0.652	0.448
9	EST	$1^1/_4$	3.00	0.881	0.242	0.291	0.582	0.414
10	EST	$1^1/_2$	3.63	1.07	0.666	0.561	1.122	0.761
11	ST	2	3.65	1.07	0.391	0.412	0.824	0.581
12	EST	2	5.02	1.48	0.868	0.731	1.462	1.02
13	ST	$2^1/_2$	5.79	1.70	1.53	1.06	2.12	1.45
14	ST	3	7.58	2.23	3.02	1.72	3.44	2.33
15	EST	$2^1/_2$	7.66	2.25	1.92	1.34	2.68	1.87
16	DEST	2	9.03	2.66	1.31	1.10	2.2	1.67
17	ST	$3^1/_2$	9.11	2.68	4.79	2.39	4.78	3.22
18	EST	3	10.25	3.02	3.89	2.23	4.46	3.08
19	ST	4	10.79	3.17	7.23	3.21	6.42	4.31
20	EST	$3^1/_2$	12.50	3.68	6.28	3.14	6.28	4.32
21	DEST	$2^1/_2$	13.69	4.03	2.87	2.00	4.00	3.04
22	ST	5	14.62	4.30	15.2	5.45	10.9	7.27
23	EST	4	14.98	4.41	9.61	4.27	8.54	5.85
24	DEST	3	18.58	5.47	5.99	3.42	6.84	5.12
25	ST	6	18.97	5.58	28.1	8.50	17.0	11.2
26	EST	5	20.78	6.11	20.7	7.43	14.86	10.1
27	DEST	4	27.54	8.10	15.3	6.79	13.58	9.97
28	ST	8	28.55	8.40	72.5	16.8	33.6	22.2
29	EST	6	28.57	8.40	40.5	12.2	24.4	16.6
30	DEST	5	38.59	11.3	33.6	12.1	24.2	17.5
31	ST	10	40.48	11.9	161	29.9	59.8	39.4
32	EST	8	43.39	12.8	106	24.5	49.0	33.0
33	ST	12	49.56	14.6	279	43.8	87.6	57.4
34	DEST	6	53.16	15.6	66.3	20.0	40.0	28.9
35	EST	10	54.74	16.1	212	39.4	78.8	52.6
36	EST	12	65.42	19.2	362	56.7	113.4	75.1
37	DEST	8	72.42	21.3	162	37.6	75.2	52.8

ST standard weight, *EST* extra strong, *DEST* double- extra strong

DEST stand for standard weight, extra strong, and double-extra strong, respectively.

These sections are taken from LRFD–AISC [29] which is also utilized as the code of practice. The process of the optimum design of the dome structures can be summarized as

$$\begin{aligned} \text{Find} \quad & \mathbf{X} = \left[x_1, x_2, \ldots, x_{ng}\right], h, Nr \\ & x_i \in \{d_1, d_2, \ldots, d_{37}\} \text{ to minimize } V(\mathbf{X}) = \sum_{i=1}^{nm} x_i \cdot L_i \\ & h_i \in \{h_{\min}, h_{\min} + h^*, \ldots, h_{\max}\} \end{aligned} \tag{9.20}$$

subjected to the following constraints:

Displacement constraint

$$\delta_i \leq \delta_i^{\max} \quad i = 1, 2, \ldots, nn \tag{9.21}$$

Interaction formula constraints

$$\frac{P_u}{2\phi_c P_n} + \left(\frac{M_{ux}}{\phi_b M_{nx}} + \frac{M_{uy}}{\phi_b M_{ny}}\right) \leq 1 \quad \text{For} \quad \frac{P_u}{\phi_c P_n} < 0.2 \tag{9.22}$$

$$\frac{P_u}{\phi_c P_n} + \frac{8}{9}\left(\frac{M_{ux}}{\phi_b M_{nx}} + \frac{M_{uy}}{\phi_b M_{ny}}\right) \leq 1 \quad \text{For} \quad \frac{P_u}{\phi_c P_n} \geq 0.2 \tag{9.23}$$

Shear constraint

$$V_u \leq \phi_v V_n \tag{9.24}$$

where $\mathbf{X}$ is the vector containing the design variables of the elements; h is the variable of the crown height; Nr is the total number of rings; d_j is the jth allowable discrete value for the design variables; $h_{\min}$, $h_{\max}$ and h^* are the permitted minimum, maximum and increased amounts of the crown height which in this part are taken as $D/20$, $D/2$ and 0.25 m, respectively in which D is the diameter of the dome; ng is the number of design variables or the number of groups; $V(\mathbf{X})$ is the volume of the structure; L_i is the length of member i; δ_i is the displacement of node i; $\delta_i^{\max}$ is the permitted displacement for the ith node; nn is the total number of nodes; ϕ_c is the resistance factor ($\phi_c = 0.9$ for tension, $\phi_c = 0.85$ for compression); ϕ_b is the flexural resistance reduction factor ($\phi_b = 0.90$); M_{ux} and M_{uy} are the required flexural strengths in the x and y directions, respectively; M_{nx} and M_{ny} are the nominal flexural strengths in the x and y directions, respectively; P_u is the required strength; and P_n denotes the nominal axial strength which is computed as

$$P_n = A_g F_{cr} \tag{9.25}$$

where A_g is the gross area of a member; and F_{cr} is calculated as following

$$F_{\text{cr}} = \left(0.658^{\lambda_c{}^2}\right) \cdot f_y \qquad \text{For} \quad \lambda_c \leq 1.5 \tag{9.26}$$

$$F_{\mathrm{cr}} = \left(\frac{0.877}{{\lambda_{\mathrm{c}}}^2}\right) \cdot f_y \qquad \text{For} \quad \lambda_{\mathrm{c}} > 1.5 \tag{9.27}$$

Here, f_y is the specified yield stress; and λ_{c} is obtained from

$$\lambda_{\mathrm{c}} = \frac{kl}{\pi r}\sqrt{\frac{f_y}{E}} \tag{9.28}$$

where k is the effective length factor taken as 1; l is the length of a dome member; r is governing radius of gyration about the axis of buckling; and E is the modulus of elasticity.

In (9.24), V_u is the factored service load shear; V_n is the nominal strength in shear; and ϕ_v represents the resistance factor for shear ($\phi_v = 0.90$).

In order to handle the constraints, the objective function for a set of design variables can be penalized to reflect any violation of the design constraints. In utilizing the penalty functions, if the constraints are satisfied, the penalty will be zero; otherwise, the amount of penalty is obtained by dividing the violation of allowable limit to the limit itself. After analyzing the structure and determining the penalty functions for each constraint, the merit function is defined as

$$Mer^k = \varepsilon_1 \cdot V^k + \varepsilon_2 \cdot \left(\Phi^k\right)^{\varepsilon_3} \tag{9.29}$$

where Mer^k = merit function for the kth candidate; ε_1, ε_2 and ε_3 = coefficients of merit function. Φ^k = summation of penalty functions for the candidate k. The main objective of optimizing structures is to find the minimum amount of the merit function. In this part, ε_1 is set to 1. The coefficient ε_2 is taken as the volume of the structure and the coefficient ε_3 is set to 1.5 but gradually, it is increased to 3 [4].

9.4.3 *Pseudo-Code of the Modified Big Bang–Big Crunch Algorithm*

The pseudo-code of the MBB–BC algorithm can be summarized as follows:

Step 1: Generate initial candidates in a random manner (considering allowable set).

Step 2: Calculate the merit function values of all the candidate solutions (Eq. 9.29).

Step 3: Find the center of the mass. The term mass refers to the inverse of the merit function value for the dome structures. The point representing the center of mass that is denoted by $A_i^{c(k)}$, is calculated according to

$$A_i^{c(k)} = \frac{\sum_{j=1}^{N} \frac{1}{Mer^j} \cdot A_i^{(k,j)}}{\sum_{j=1}^{N} \frac{1}{Mer^j}} \quad i = 1, 2, \ldots, ng \tag{9.30}$$

where $A_i^{(k,j)}$ is the ith component of the jth solution generated in the kth iteration; N is the population size in Big Bang phase.

Step 4: Calculate new candidates around the center of the mass. The modified BB–BC approach uses the center of mass, the best position of each candidate ($A_i^{lbest(k,j)}$) and the best global position ($A_i^{gbest(k)}$) to generate a new solution as:

$$A_i^{(k+1,j)} = \alpha_2 A_i^{c(k)} + (1-\alpha_2)\left(\alpha_3 A_i^{gbest(k)} + (1-\alpha_3) A_i^{lbest(k,j)}\right) + \frac{r_j \alpha_1 (A_{\max} - A_{\min})}{k+1} \quad \left\langle \begin{matrix} i = 1, 2, \ldots, ng \\ j = 1, 2, \ldots, N \end{matrix} \right. \tag{9.31}$$

where r_j is a random number from a standard normal distribution which changes for each candidate; α_1 is a parameter for limiting the size of the search space; $A_{\min} = 0.250$ in. ^{2}; $A_{\max} = 21.3$ in. ^{2}; $A_i^{lbest(k,j)}$ is the best position of the jth particle up to the iteration k and $A_i^{gbest(k)}$ is the best position among all candidates up to the iteration k; α_2 and α_3 are adjustable parameters controlling the influence of the global best and local best on the new position of the candidates, respectively.

In order to reach a discrete solution, the new position of each agent is redefined as the following

$$A_i^{(k+1,j)} = Fix\left(\alpha_2 A_i^{c(k)} + (1-\alpha_2)\left(\alpha_3 A_i^{gbest(k)} + (1-\alpha_3) A_i^{lbest(k,j)}\right) + \frac{r_j \alpha_1 (A_{\max} - A_{\min})}{k+1}\right) \tag{9.32}$$

where $Fix(\mathbf{X})$ is a function which rounds each elements of $\mathbf{X}$ to the nearest permissible discrete value. Using this position updating formula, the agents will be permitted to select discrete values [12].

Step 5: Return to Step 2 and repeat the process until the condition for the stopping criterion is fulfilled.

9.4.4 Elastic Critical Load Analysis of Spatial Structures

The dome structures are rigid structures for which the overall loss of stability might take place when these structures are subjected to equipment loading concentrated at the apex. Therefore, stability check is necessary during the analysis to ensure that the structure does not lose its load carrying capacity due to instability [24] and furthermore, considering the nonlinear behaviour in the design of domes is necessary because of the change in geometry under external loads.

Details of the elastic instability analysis of a dome with rigid connections are carried out as the following [24]:

Step 1: Set the load factor to a pre-selected initial value and assume the axial forces in members are equal to zero.

Step 2: Compute the stability functions using the current values of axial forces in members, as in [30].

Step 3: Set up the nonlinear stiffness matrix for each member.

Step 4: Transform the member stiffness matrices from local coordinates into the global coordinate and assemble the overall stiffness matrix.

Step 5: Check the stability of the dome. Calculate the determinant of the overall stiffness matrix. If it becomes negative, then the dome becomes instable and the design process is terminated; otherwise, go to the next step.

Step 6: Analyze the dome under the factored external loads and obtain joint displacements.

Step 7: Find the member forces.

Step 8: Replace the previous axial forces in members with the new ones.

Step 9: Repeat the steps from 2 until differences between two successive sets of axial forces are smaller than a specific tolerance.

Step 10: Increase the load factor by pre-selected increment. If the load factor has reached the specified ultimate value, terminate the elastic critical load analysis; otherwise, go to Step 2.

9.4.5 Configuration of Schwedler and Ribbed Domes

The configuration of a Schwedler dome is shown in Fig. 9.11. Schwedler, a German engineer, who introduced this type of dome in 1863, built numerous braced domes during his lifetime. A Schwedler dome, one of the most popular types of braced domes, consists of meridional *ribs* connected together to a number of horizontal polygonal *rings*. To stiffen the resulting structure, each trapezium formed by intersecting meridional ribs with horizontal rings is subdivided into two triangles by introducing a *diagonal* member.

The number of nodes in each ring for the Schwedler domes is considered constant and it is equal to ten in this part. The distances between the rings in the dome on the meridian line are generally of equal length. The structural data for the

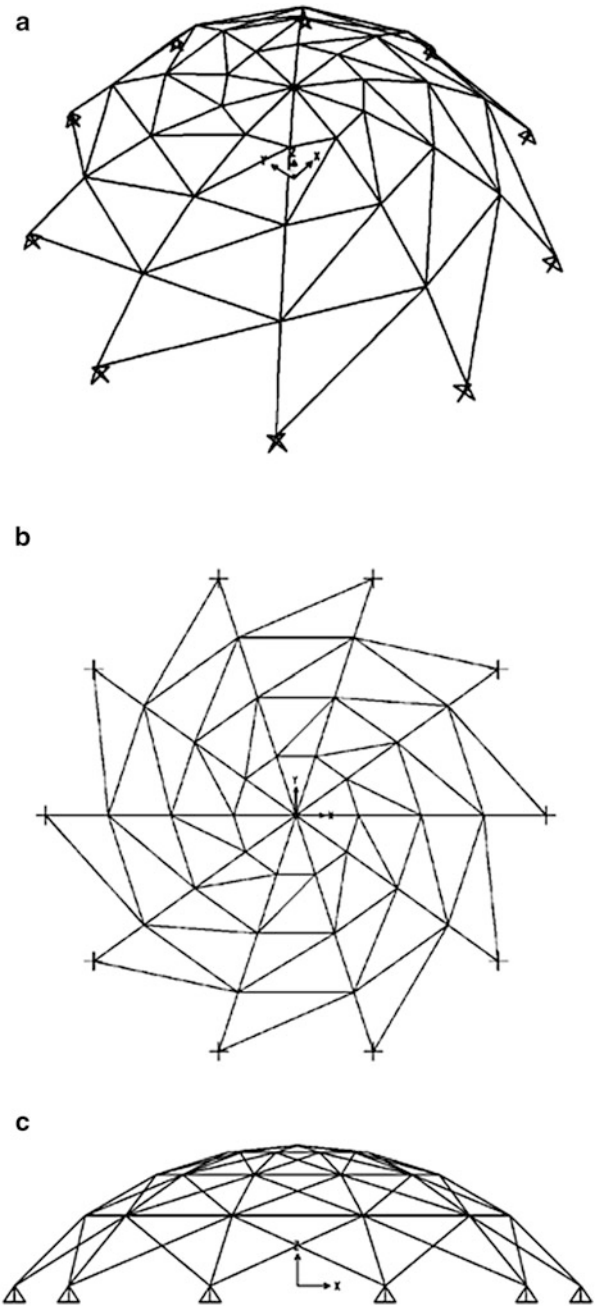

Fig. 9.11 Schematic of a Schwedler dome [3]. (**a**) 3-D view, (**b**) top view, (**c**) side view

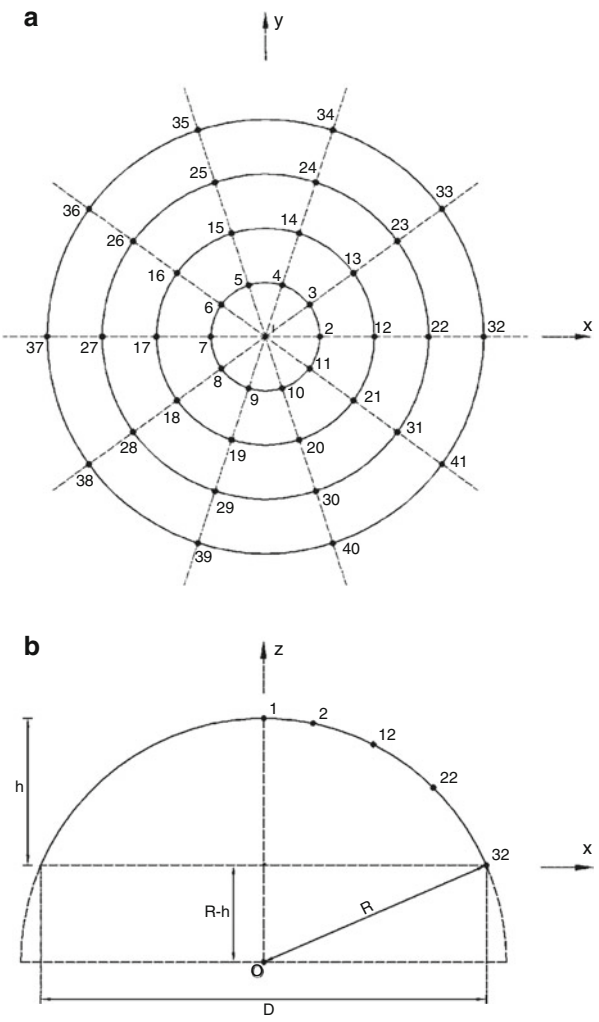

Fig. 9.12 Nodal numbering and the corresponding coordinate system [3]. (**a**) Top view of the dome and (**b**) section of the dome

geometry of this form of the Schwedler domes is a function of the diameter of the dome (D), the total number of rings (N), and the height of the crown (h). The total number of rings can be selected 3, 4 or 5. The top joint at the crown is numbered as first joint as shown in Fig. 9.12a (joint number 1) which is located in the centre of the coordinate system in x-y plane. The coordinates of other joints in each ring are obtained as

$$\begin{cases} x_i = \dfrac{D}{2N}\cos\left(\dfrac{360}{4n_i}\left(i - \sum_{j=1}^{i-1} 4n_j - 1\right)\right) \\ y_i = \dfrac{D}{2N}\sin\left(\dfrac{360}{4n_i}\left(i - \sum_{j=1}^{i-1} 4n_j - 1\right)\right) \\ z_i = \sqrt{\left(R^2 - \dfrac{n_i^2 D^2}{4N^2}\right)} - (R - h) \end{cases} \tag{9.33}$$

where n_i is the number of ring corresponding to the node i; $R = (D^2 + 4h^2)/(8h)$ is the radius of the hemisphere as shown in Fig. 9.12b.

The member of grouping is determined in a way that rib members between each consecutive pair of rings belong to one group, diagonal members belong to one group and the members on each ring form another group. Therefore, the total number of groups is equal to $3n_i - 2$. Figure 9.13 shows the number of groups corresponding to rib, diagonal and ring members. The configuration of elements contains determining the start and end nodes of each element.For the first group, the start node for all elements is the joint number 1 and the end nodes are those on the first ring. The start and end nodes of ring groups can be obtained using following equations:

$$\begin{cases} I = 10 \times (n_i - 1) + j + 1 \\ J = 10 \times (n_i - 1) + j + 2 \end{cases} \quad \begin{matrix} j = 1, 2, 3, \ldots, 9 \\ n_i = 1, 2, \ldots, Nr - 1 \end{matrix} \tag{9.34}$$

$$\begin{cases} I = 10 \times (n_i - 1) + 2 \\ J = 10 \times n_i + 1 \end{cases} \quad n_i = 1, 2, \ldots, Nr - 1 \tag{9.35}$$

Also for the rib and diagonal groups, we have

$$\begin{cases} I = 10 \times \left(n_i - {}^{i}1\right) + 2 + Fix\left(\dfrac{j-1}{2}\right) \\ J = 10 \times n_i + {}^{i}j - Fix\left(\dfrac{j-1}{2}\right) \end{cases} \quad \begin{matrix} j = 2, 3, \ldots, 20 \\ n_i = 1, 2, \ldots, Nr - 1 \end{matrix} \tag{9.36}$$

$$\begin{cases} I = 10 \times (n_i - 1) + 2 \\ J = 10 \times (n_i + 1) + 1 \end{cases} \quad n_i = 1, 2, \ldots, Nr - 1 \tag{9.37}$$

where I and J are the start and end nodal numbers of the elements, respectively. Equation (9.34) determines the elements of ring groups where each element is made up of two consecutive nodes on each ring. The element with the lower and upper numbers on each ring also corresponds to that group, according to (9.35). Equations

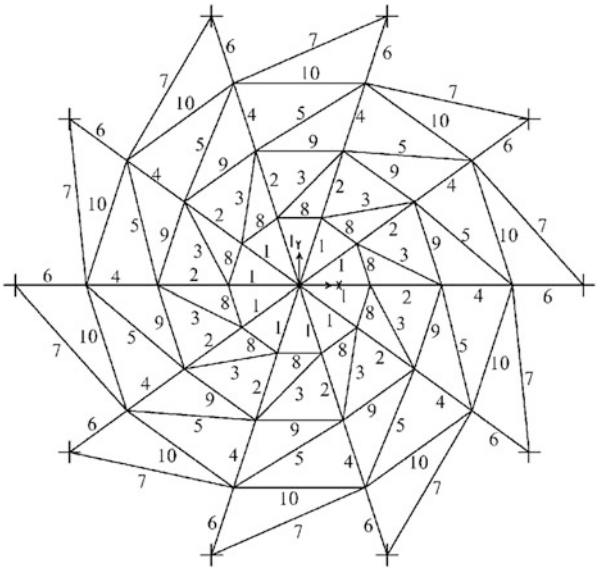

Fig. 9.13 The Schwedler dome with the related member grouping [3]

(9.36) and (9.37) present the total elements of the rib and diagonal groups located between the rings n_i and $n_i + 1$. Equation (9.37) presents only one element which connects the first node on the ring n_i to the last node on the ring $n_i + 1$.

A dome without the diagonal members is called the *ribbed dome*, as shown in Fig. 9.14. For these domes Eqs. (9.33), (9.34) and (9.35) are also valid to determine the joint coordinates and the ring member constructions. However, the rib members are assigned using the following relationship:

$$\begin{cases} I = 10 \times (n_i - 1) + j + 1 \\ J = 10 \times n_i + j + 1 \end{cases} \quad \begin{cases} j = 1, 2, \ldots, 10 \\ n_i = 1, 2, \ldots, Nr - 1 \end{cases} \tag{9.38}$$

9.4.6 Results and Discussion

This section presents the optimum design of the Schwedler and ribbed domes using the MBB–BC algorithm. The modulus of elasticity for the steel is taken as 205 kN/mm^2. The limitations imposed on the joint displacements are 28 mm in the z direction and 33 mm in the x and y directions for the 1st, 2nd and 3rd nodes, respectively.

For the proposed algorithm, a population of 50 individuals is used. Using $\alpha_1 = 1.0$ allows the initial search of the full range of values for each design variable. Previous investigations show that $\alpha_2 = 0.40$ and $\alpha_3 = 0.80$ are suitable values for MBB–BC algorithm. Here, first a comparison is made between the Schwedler and ribbed domes, and then the efficiency of the Schwedler domes for various diameters is investigated.

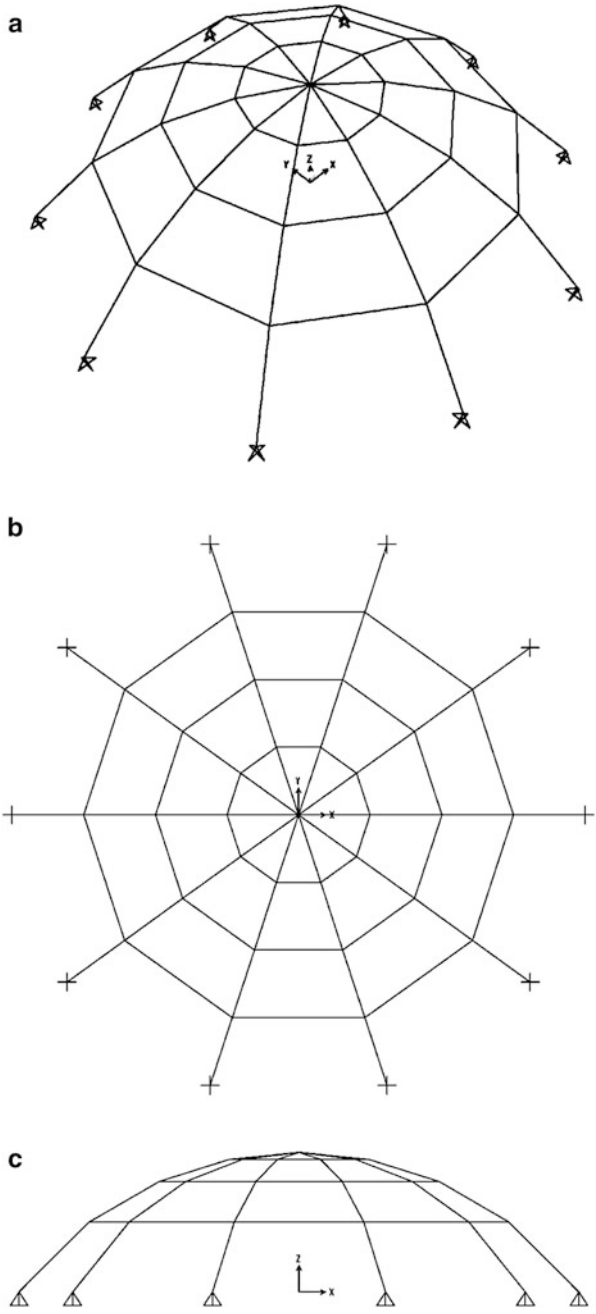

Fig. 9.14 Schematic of a ribbed dome [3]. (**a**) 3-D view, (**b**) top view, (**c**) side view

9.4.6.1 Comparison of the Schwedler and Ribbed Domes

The diameter of the dome is selected as 40 m. The dome is considered to be subjected to equipment loading at its crown. The three loading conditions are considered:

Case 1. The vertical downward load of −500 kN;

Case 2. The two horizontal loads of 100 kN in the x and y directions;

Case 3. The vertical downward load of −500 kN and two horizontal loads of 100 kN in the x and y directions.

Table 9.10 presents the results for the Schwedler and ribbed domes. In all loading cases, the optimum number of rings for both domes is three. The volume of the dome structures can be considered as a function of the average cross-sectional area of the elements ($\overline{A}$) and the sum of the element lengths, written as

$$V(\mathbf{X}) = \overline{A} \cdot \sum_{i=1}^{nm} L_i \tag{9.20}$$

In Case 1, $\overline{A}$ for the ribbed dome is 60 % more than the Schwedler one. Both domes have approximately the same height; therefore, because of having less number of elements, the ribbed dome has smaller value (64 %) for the sum of the element lengths than the Schwedler dome. Therefore, the difference of the volume for the domes is small and increasing the sum of element lengths for the Schwedler dome is compensated by reduction of the average cross-sectional areas of the elements.

Because of existing only horizontal forces in Case (2), the angles of elements with the horizontal line in the optimum design must have the minimum value; therefore, both domes have the minimum allowable heights. When comparing the optimum sections for these two types of domes, it can be shown that the rib members in the ribbed dome have much heavier sections than the rings elements, while almost all members in the Schwedler dome are not so much different. In addition, contrary to Case 1 and Case 3, the neighboring elements to supports in both domes have the stronger sections than the others, while in two other cases the elements near to the apex have the heavier members. Another interesting point is that the stress constraints are dominant for the Schwedler dome while for the ribbed dome, the displacement constraints are dominant. Therefore, the Schwedler dome has better performance against the external lateral forces and has the smaller volume.

The Schwedler dome contains more appropriate sections and lighter weight than the ribbed dome for Case 3. In order to provide lateral stiffness, all rib members in the ribbed domes have very strong sections, and $\overline{A}$ has a very large value; whereas the Schwedler dome has small $\overline{A}$ because of existing diagonal elements which provide the necessary lateral stiffness against the horizontal external loads. The height of the ribbed dome must be selected small because of existing the horizontal loads in one hand and in the other hand, it must have a large value to provide the

Table 9.10 Optimum design of the Ribbed and Schwedler domes

	Optimum section (Designations)					
	Case 1		Case 2		Case 3	
Group number	Ribbed dome	Schwedler dome	Ribbed dome	Schwedler dome	Ribbed dome	Schwedler dome
1	PIPST (8)	PIPST (8)	PIPST (6)	PIPST (3)	PIPST (12)	PIPST (10)
2	PIPST (5)	PIPST ($^1/_2$)	PIPST (6)	PIPST (3)	PIPST (12)	PIPST ($3^1/_2$)
3	PIPST (5)	PIPST (5)	PIPST (10)	PIPST ($2^1/_2$)	PIPST (10)	PIPST (6)
4	PIPST (8)	PIPST ($^1/_2$)	PIPST ($^1/_2$)	PIPST ($3^1/_2$)	PIPST (8)	PIPST (4)
5	PIPST (5)	PIPST (5)	PIPST ($1^1/_4$)	PIPST ($2^1/_2$)	PIPST (6)	PIPST (5)
6	N/A	PIPST (8)	N/A	PIPEST (2)	N/A	PIPST (8)
7	N/A	PIPST (5)	N/A	PIPST (3)	N/A	PIPST (5)
Height (m)	13.5	13.5	2.00	2.00	7.25	10.75
Max. displacement (cm)	2.80	2.80	3.29	1.79	3.30	2.73
Max. strength ratio	0.79	0.81	0.63	0.95	0.82	0.92
Volume (m^3)	1.33	1.38	1.16	0.74	2.42	1.94
$\sum l_i$ (m)	377.75	623.25	324.90	535.70	340.20	591.10
$\overline{A}$ (cm^2)	35.35	22.06	35.15	13.81	71.20	32.83

necessary strength against the vertical load and to avoid instability. Thus, the optimum height of the ribbed dome is constrained to a small range. It is obtained 7.25 m which is between the optimum heights in two previous cases. For the Schwedler dome, the diagonal and rib elements provide the lateral and vertical strengths, respectively. Therefore, the height of the dome can be selected from a broad range and the algorithm has a large space to find the optimum design. To sum up, the Schwedler domes are more appropriate than the ribbed ones against vertical and horizontal loads.

9.4.6.2 Schwedler Domes with Different Diameters

In order to investigate the efficiency of the Schwedler domes, two other domes with different diameters are considered: one with smaller diameter (20 m) and another with larger diameter (60 m). The loading condition is the same as the Case 3 in previous section. Figure 9.15 shows the normalized optimum volume of these domes when the number of rings is altered. For all three cases, a dome with three rings is lighter. Optimum designs for domes with five and four rings are approximately 18 % and 8 % heavier than the one with three rings in average, respectively. Therefore, it seems that selecting a minimum number for the rings leads the better

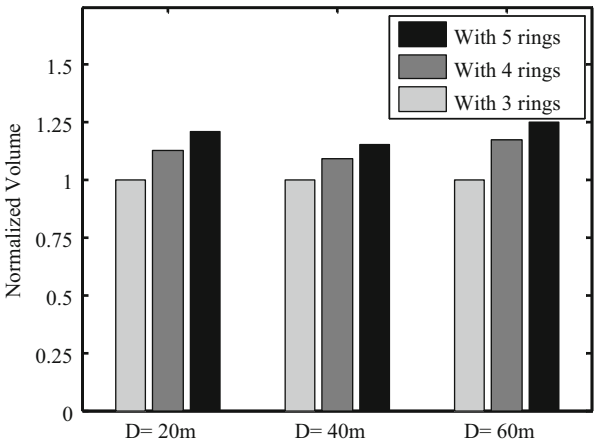

Fig. 9.15 The normalized optimum volume for the Schwedler domes with different number of rings [3]

results unless a dome has very large diameter in which case some elements will buckle if the number of rings is selected small.

The optimum height of crown is 5.25 m, 10.75 m and 18.5 m for domes with 20 m, 40 m and 60 m diameters and the ratio of the height to the diameter is equal to 0.26, 0.27, and 0.31, respectively. Thus, when the diameter of the dome increases, the ratio of the height to the diameter raises slightly. It seems that the range of 0.2 to 0.4 can be utilized as a good search space for the ratio of the height to the diameter.

The convergence history for the studied Schwedler domes are shown in Fig. 9.16, and the comparison of the optimal design of Schwedler domes with different diameters is made in Table 9.11. In this table, the mean of the required materials to cover the space is obtained by dividing the optimum volume of each dome to the covered area by the dome ($\pi D^2/4$).

In other words, this ratio can be considered as the cost of required structural material to the space being covered. Almost for all domes, the required structural material is the same and this shows the suitability of the Schwedler domes to cover large areas.

9.4.7 Discussion

A Modified Big Bang–Big Crunch optimization is developed for optimal design of geometrically nonlinear Schwedler and ribbed domes. This method consists of a Big Bang phase where candidate solutions are randomly distributed over the search space, and a Big Crunch phase working as a convergence operator where the center of mass is generated. The Particle Swarm Optimization capacities are added to improve the exploration ability of the algorithm. A simple procedure is developed to determine the configuration of the ribbed and Schwedler domes. Using this procedure, the joint coordinates are calculated and the elements are constructed.

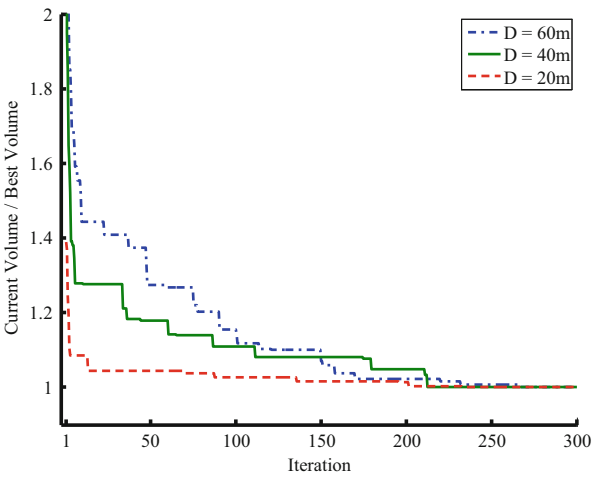

Fig. 9.16 The convergence history for the Schwedler domes [3]

Table 9.11 Comparison of optimal design of the Schwedler domes with different diameters

	D = 20 m	D = 40 m	D = 60 m
Height (m)	5.25	10.75	18.50
Max. displacement (cm)	2.73	2.73	2.80
Max. strength ratio	0.96	0.92	0.99
Volume (m^3)	0.53	1.94	4.11
$\sum l_i$ (m)	294.25	591.10	913.27
$\overline{A}$ (cm^2)	18.16	32.83	45.08
Required materials to cover the space	0.170	0.154	0.145

The domes with the diagonal elements (Schwedler domes) and without them (ribbed domes) are optimized using the MBB–BC algorithm.

The three considered loading conditions consist of the vertical downward load, the two horizontal loads and both of these loads acting simultaneously. In Case 1, the volume of the ribbed dome is smaller than the Schwedler one because of having less number of elements. In Case 2, both domes have the minimum height and the stress constraints are dominant for the Schwedler dome while for the ribbed one, the displacement constraints are dominant. In Case 3, the Schwedler dome has lighter weight. Despite the fact that diagonal elements increase the sum of the element lengths, they have efficient influences against vertical and horizontal loads and therefore, the MBB–BC algorithm is allowed to select some lighter sections for other elements in the Schwedler domes. In addition, the efficiency of the Schwedler domes to cover various areas is investigated. The results show that a minimum number for rings is the best choice and selecting a ratio of the height to the diameter from the range of [0.2, 0.4] can improve the performance of the dome. Finally, the results reveal that the normalized required material for Schwedler domes is approximately identical for small or large areas. As a result, this type of domes can be considered as a good selection to cover large areas without intermediate columns.

References

1. Erol OK, Eksin I (2006) New optimization method: Big Bang–Big Crunch. Adv Eng Softw 37:106–111
2. Kaveh A, Talatahari S (2009) Size optimization of space trusses using Big Bang–Big Crunch algorithm. Comput Struct 87:1129–1140
3. Kaveh A, Talatahari S (2010) Optimal design of Schwedler and ribbed domes; hybrid Big Bang-Big Crunch algorithm. J Construct Steel Res 66:412–419
4. Kaveh A, Farahmand Azar B, Talatahari S (2008) Ant colony optimization for design of space trusses. Int J Space Struct 23(3):167–181
5. Kennedy J, Eberhart R, Shi Y (2001) Swarm intelligence. Morgan Kaufmann Publishers, Edinburgh
6. Kaveh A, Talatahari S (2010) An improved ant colony optimization for constrained engineering design problems. Eng Comput 27(1):155–182
7. He S, Wu QH, Wen JY, Saunders JR, Paton RC (2004) A particle swarm optimizer with passive congregation. Biosystem 78:135–147
8. Rajeev S, Krishnamoorthy CS (1992) Discrete optimization of structures using genetic algorithms. J Struct Eng ASCE 118(5):1233–1250
9. Camp CV, Bichon J (2005) Design of steel frames using ant colony optimization. J Struct Eng ASCE 131:369–379
10. Kaveh A, Shojaee S (2007) Optimal design of skeletal structures using ant colony optimisation. Int J Numer Methods Eng 70(5):563–581
11. Van Laarhoven PJM, Aarts EHL (1998) Simulated annealing, theory and applications. Kluwer Academic, Boston, MA
12. Dorigo M (1992) Optimization, Learning and Natural Algorithms. Ph.D thesis. Dipartimento di Elettronica e Informazione, Politecnico di Milano, IT (in Italian)
13. Dorigo M, Caro GD, Gambardella LM (1999) An algorithm for discrete optimization. Artif Life 5:137–172
14. Camp CV (2007) Design of space trusses using Big Bang–Big Crunch optimization. J Struct Eng ASCE 133:999–1008
15. Kaveh A, Talatahari S (2008) A hybrid particle swarm and ant colony optimization for design of truss structures. Asian J Civil Eng 9(4):329–348
16. Kaveh A, Talatahari S (2009) Particle swarm optimizer, ant colony strategy and harmony search scheme hybridized for optimization of truss structures. Comput Struct 87:267–283
17. Schutte JJ, Groenwold AA (2003) Sizing design of truss structures using particle swarms. Struct Multidiscip Optim 25:261–269
18. Lee KS, Geem ZW (2004) A new structural optimization method based on the harmony search algorithm. Comput Struct 82:781–798
19. Camp CV, Bichon J (2004) Design of space trusses using ant colony optimization. J Struct Eng ASCE 130(5):741–751
20. Perez RE, Behdinan K (2007) Particle swarm approach for structural design optimization. Comput Struct 85:1579–1588
21. Erbatur F, Hasancebi O, Tutuncil I, Kihc H (2000) Optimal design of planar and space structures with genetic algorithms. Comput Struct 75:209–224
22. American Institute of Steel Construction (AISC) (1989) Manual of steel construction-allowable stress design, 9th edn. American Institute of Steel Construction (AISC), Chicago, IL
23. Salajegheh E, Vanderplaats GN (1986/87) An efficient approximation method for structural synthesis with reference to space structures. Int J Space Struct 2:165–175
24. Saka MP, Kameshki ES (1998) Optimum design of nonlinear elastic framed domes. Adv Eng Softw 29(7–9):519–528
25. Makowski ZS (1984) Analysis, design and construction of braced domes. Granada Publishing Ltd, London

26. Coates RC, Coutie MG, Kong FK (1972) Structural analysis. Thomas Nelson & Sons Ltd., London
27. Saka MP (2007) Optimum geometry design of geodesic domes using harmony search algorithm. Adv Struct Eng 10:595–606
28. Saka MP (2007) Optimum topological design of geometrically nonlinear single layer latticed domes using coupled genetic algorithm. Comput Struct 85:1635–1646
29. American Institute of Steel Construction (AISC) (1991) Manual of steel construction-load resistance factor design, 3rd edn. AISC, Chicago, IL
30. Ekhande SG, Selvappalam M, Madugula KS (1989) Stability functions for three-dimensional beam-columns. J Struct Eng ASCE 115:467–479

Chapter 10
Cuckoo Search Optimization

10.1 Introduction

In this chapter, a metaheuristic method so-called Cuckoo Search (CS) algorithm is utilized to determine optimum design of structures for both discrete and continuous variables. This algorithm is recently developed by Yang [1], Yang and Deb [2, 3], and it is based on the obligate brood parasitic behavior of some cuckoo species together with the Lévy flight behavior of some birds and fruit flies. The CS is a population based optimization algorithm and similar to many others metaheuristic algorithms starts with a random initial population which is taken as host nests or eggs. The CS algorithm essentially works with three components: selection of the best by keeping the best nests or solutions; replacement of the host eggs with respect to the quality of the new solutions or Cuckoo eggs produced based randomization via Lévy flights globally (exploration); and discovering of some cuckoo eggs by the host birds and replacing according to the quality of the local random walks (exploitation) [2].

This chapter consists of two parts. In part 1, optimum design of the truss structures is presented for both discrete and continuous variables, based on the Cuckoo Search (CS) algorithm [4]. In order to demonstrate the effectiveness and robustness of the present method, minimum weight design of truss structures is performed and the results of the CS and some well-known metaheuristic algorithms are compared for some benchmark truss structures.

In part 2, optimum design of two dimensional steel frames for discrete variables based on the Cuckoo search (CS) algorithm is presented [5].

A. Kaveh, *Advances in Metaheuristic Algorithms for Optimal Design of Structures*,
DOI 10.1007/978-3-319-05549-7_10, © Springer International Publishing Switzerland 2014

10.2 Optimum Design of Truss Structures Using Cuckoo Search Algorithm with Lévy Flights

10.2.1 Formulation

The aim of optimizing a truss structure is to find a set of design variables corresponding to the minimum weight satisfying certain constraints. This can be expressed as:

$$\begin{aligned} &\text{Find} \qquad \{X\} = \left[x_1, x_2, \ldots, x_{ng}\right], \;\; x_i \in D_i \\ &\text{To minimize} \qquad W(\{X\}) = \sum_{\mathrm{i}=1}^{ng} x_{\mathrm{i}} \sum_{j=1}^{nm(i)} \rho_j . L_j \\ &\text{Subject to :} \qquad g_j(\{X\}) \leq 0 \quad j = 1, 2, \ldots, n \end{aligned} \tag{10.1}$$

where $\{X\}$ is the set of design variables; ng is the number of member groups in structure (number of design variables); D_i is the allowable set of values for the design variable x_i ; $W(\{\mathrm{X}\})$ presents weight of the structure; $nm\,(i)$ is the number of members for the ith design variable; ρ_j and L_j denotes the material density and the length of the member j, respectively; $g_j(\{X\})$ denotes design constraints; and n is the number of the constraints. D_i can be considered either as a continuous set or as a discrete one. In the continuous problems, the design variables can vary continuously in the optimization process.

$$D_i = \left\{ x_i \middle| x_i \in [x_{i,min}, x_{i,max}] \right\} \tag{10.2}$$

where $x_{i,min}$ and $x_{i,max}$ are minimum and maximum allowable values for the design variables x_i, respectively. If the design variables represent a selection from a set of parts as

$$D_i = \left(d_{i,1}, d_{i,2}, \ldots, d_{i,nm(i)}\right) \tag{10.3}$$

then the problem can be considered as a discrete one.

In order to handle the constraints, a penalty approach is utilized. In this method, the aim of the optimization is redefined by introducing the cost function as:

$$f_{\cos t}(\{X\}) = (1 + \varepsilon_1 . \upsilon)^{\varepsilon_2} \times W(\{X\}), \quad \upsilon = \sum_{j=1}^{n} \max\left[0, g_j(\{X\})\right] \tag{10.4}$$

where n represents the number of evaluated constraints for each individual design, and υ denotes the sum of the violations of the design. The constants ε_1 and ε_2 are selected considering the exploration and the exploitation rate of the search space.

Here, ε_1 is set to unity, ε_2 is selected in a way that it decreases the penalties and reduces the cross-sectional areas. Thus, in the first steps of the search process, ε_2 is set to 1.5 and ultimately increased to 3.

The constraint conditions for truss structures are briefly explained in the following. The stress limitations of the members are imposed according to the provisions of ASD-AISC [6] as follows:

$$\begin{cases} \sigma_i^+ = 0.6\,\mathrm{F}_y & \text{for } \sigma_i \geq 0 \\ \sigma_i^- & \text{for } \sigma_i < 0 \end{cases} \tag{10.5}$$

$$\sigma_i^- = \begin{cases} \left[\left(1 - \dfrac{\lambda_i^2}{2\,\mathrm{c}_c^2}\right) F_y\right] \Big/ \left(\dfrac{5}{3} + \dfrac{3\lambda_i}{8\,c_c} + \dfrac{\lambda_i^3}{8\,\mathrm{c}_c^3}\right) & \textit{for } \lambda_i \geq c_c \\ \dfrac{12\,\pi^2\,E}{23\,\lambda_i^2} & \textit{for } \lambda_i \geq c_c \end{cases} \tag{10.6}$$

where, E is the modulus of elasticity; F_y is the yield stress of steel; c_c denotes the slenderness ratio (λ_i) dividing the elastic and inelastic buckling regions ($c_c = \sqrt{2\pi^2 E/F_y}$); λ_i = the slenderness ratio ($\lambda_i = kl_i/r_i$); k = the effective length factor; L_i = the member length; and r_i = the radius of gyration. The radius of gyration (r_i) can be expressed in terms of cross-sectional areas as $r_i = a\,A_i^b$. Here, a and b are the constants depending on the types of sections adopted for the members such as pipes, angles, and tees. In this study, pipe sections ($a = 0.4993$ and $b = 0.6777$) are adopted for bars.

The other constraint corresponds to the limitation of the nodal displacements:

$$\delta_i - \delta_i^u \leq 0 \qquad i = 1, 2, \ldots, nn \tag{10.7}$$

where δ_i is the nodal deflection; δ_i^u is the allowable deflection of node i; and nn is the number of nodes.

10.2.2 Lévy Flights as Random Walks

The randomization plays an important role in both exploration and exploitation in metaheuristic algorithms. The Lévy flights as random walks can be described as follows [2]:

A random walk is a random process which consists of taking a series of consecutive random steps. A random walk can be expressed as:

$$S_n = \sum_{i=1}^{n} X_i = X_1 + X_2 + \ldots + X_n = \sum_{i=1}^{n-1} X_i + X_n = S_{n-1} + X_n \tag{10.8}$$

where S_n presents the random walk with n random steps and X_i is the ith random step with predefined length. The last statement means that the next state will only depend on the current existing state and the motion or transition X_n. In fact the step size or length can vary according to a known distribution. A very special case is when the step length obeys the Lévy distribution; such a random walk is called a Lévy flight or Lévy walk. In fact, Lévy flights have been observed among foraging patterns of albatrosses, fruit flies and spider monkeys.

From the implementation point of view, the generation of random numbers with Lévy flights consists of two steps: the choice of a random direction and the generation of steps which obey the chosen Lévy distribution. While the generation of steps is quite tricky, there are a few ways of achieving this. One of the most efficient and yet straightforward ways is to use the so-called Mantegna algorithm. In the Mantegna's algorithm, the step length S can be calculated by:

$$S = \frac{u}{|v|^{1/\beta}} \tag{10.9}$$

where β is a parameter between [1, 2] interval and considered to be 1.5; u and v are drawn from normal distribution as

$$u \sim N\left(0, \sigma_u^2\right), \qquad v \sim N\left(0, \sigma_v^2\right) \tag{10.10}$$

where

$$\sigma_u = \left\{ \frac{\Gamma(1+\beta)\sin(\pi\beta/2)}{\Gamma[(1+\beta)/2]\,\beta\,2^{(\beta-1)/2}} \right\}^{1/\beta}, \qquad \sigma_v = 1 \tag{10.11}$$

Studies show that the Lévy fights can maximize the efficiency of the resource searches in uncertain environments. In fact, Lévy flights have been observed among foraging patterns of albatrosses, fruit flies and spider monkeys.

10.2.3 Cuckoo Search Algorithm

This algorithm is inspired by some species of a bird family called cuckoo because of their special lifestyle and aggressive reproduction strategy. These species lay their eggs in the nests of other host birds (almost other species) with amazing abilities such as selecting the recently spawned nests, and removing the existing eggs that increase the hatching probability of their eggs. On the other hand, some of host

birds are able to combat this parasites behavior of cuckoos, and throw out the discovered alien eggs or build their new nests in new locations.

This algorithm contains a population of nests or eggs. For simplicity, following representations is used; where each egg in a nest represents a solution and a Cuckoo egg represents a new one. If the Cuckoo egg is very similar to the host's egg, then this Cuckoo's egg is less likely to be discovered, thus the fitness should be related to the difference in solutions. The aim is to employ the new and potentially better solutions (Cuckoos') to replace a not-so-good solution in the nests [2].

For simplicity in describing the CS, the following three idealized rules are utilized [3]:

1. Each Cuckoo lays one egg at a time, and dumps it in a randomly chosen nest;
2. The best nests with high quality of eggs are carried over to the next generations;
3. The number of available host nests is constant, and the egg which is laid by a Cuckoo is discovered by the host bird with a probability of *pa* in the range of [0, 1]. The later assumption can be approximated by the fraction *pa* of the *n* nests is replaced by new ones (with new random solutions).

Based on the above three rules, the basic steps of the CS can be summarized as the pseudo code shown in Fig. 10.1.

This pseudo code, provided in the book entitled Nature-Inspired metaheuristic algorithms, by Yang [1], is a sequential version and each iteration of the algorithm consisting of two main steps, but another version of the CS which is supposed to be different and more efficient is provided by Yang and Deb [3]. This new version has some differences with the book version as follows:

In the first step according to the pseudo code, one of the randomly selected nests (except the best one) is replaced by a new solution, produced by random walk with Lévy flight around the so far best nest, considering the quality. But in the new version, all of the nests except the best one are replaced in one step, by new solutions. When generating new solutions $x_i^{(t+1)}$ for the *i*th Cuckoo, a Lévy flight is performed using the following equation:

$$x_i^{(t+1)} = x_i^{(t)} + \alpha.S \tag{10.12}$$

where $\alpha > 0$ is the step size parameter and should be chosen considering the scale of the problem and is set to unity in the CS [2], and decreases function as the number of generations increases in the modified CS. It should be noted that in this new version, the solutions' current positions are used instead of the best solution so far as the origin of the Lévy flight. The step size is considered as 0.1 in this work because it results in efficient performance of algorithm in our examples. The parameter S is the length of random walk with Lévy flights according to the Mantegna's algorithm as described in (10.9).

In the second step, the *pa* fraction of the worst nests are discovered and replaced by new ones. However, in the new version, the parameter *pa* is considered as the

```
Objective function f(x), x = (x_1, x_2, ..., x_d) ;
Generate initial population of n host nests
x_i (i =1 , 2, .., n) ;
while (stop criterion)
  Get a Cuckoo randomly by Lévy flights;
  Evaluate its quality/fitness F_i ;
  Choose a nest among n (say j ) randomly;
  if F_i ≥ F_j
     replace j by the new solution;
  end
  Abandon a fraction (pa) of worse nests
  [and build new ones at new locations via Lévy flights]
  Keep the best solutions (or nests with quality solutions);
  Rank the solutions and find the current best;
end while
Post process results and visualization;
```

Fig. 10.1 Pseudo code of the CS [4]

probability of a solution's component to be discovered. Therefore, a probability matrix is produced as:

$$P_{ij} = \begin{cases} 1 & if\ rand < pa \\ 0 & if\ rand \geq pa \end{cases} \tag{10.13}$$

where *rand* is a random number in [0, 1] interval and P_{ij} is discovering probability for *j*th variable of *i*th nest. Then all of the nests are replaced by new ones produced by random walks (point wise multiplication of random step sizes with probability matrix) from their current positions according to quality. In this study the later version of the CS algorithm is used for optimum design of truss structures.

10.2.4 Optimum Design of Truss Structures Using Cuckoo Search Algorithm

The pseudo code of optimum design algorithm is as follows:

10.2.4.1 Initialize the Cuckoo Search Algorithm Parameters

The CS parameters are set in the first step. These parameters are number of nests (n), step size parameter (α), discovering probability (pa) and maximum number of analyses as the stopping criterion.

10.2.4.2 Generate Initial Nests or Eggs of Host Birds

The initial locations of the nests are determined by the set of values assigned to each decision variable randomly as

$$nest_{i,j}^{(0)} = x_{j,\min} + rand.\left(x_{j,\max} - x_{j,\min}\right) \tag{10.14a}$$

where $nest_{i,j}{}^{(0)}$ determines the initial value of the jth variable for the ith nest; $x_{j,min}$ and $x_{j,max}$ are the minimum and the maximum allowable values for the jth variable; *rand* is a random number in the interval [0, 1]. For problems with discrete design variables it is necessary to use a rounding function as

$$nest_{i,j}^{(0)} = ROUND\left(x_{j,\min} + rand.\left(x_{j,\max} - x_{j,\min}\right)\right) \tag{10.14b}$$

10.2.4.3 Generate New Cuckoos by Lévy Flights

In this step all of the nests except for the best so far are replaced in order of quality by new Cuckoo eggs produced with Lévy flights from their positions as

$$nest_i^{(t+1)} = nest_i^{(t)} + \alpha.S.\left(nest_i^{(t)} - nest_{best}^{(t)}\right).r \tag{10.15}$$

where $nest_i{}^t$ is the ith nest current position; α is the step size parameter which is considered to be 0.1; S is the Lévy flights vector as in Mantegna's algorithm; r is a random number from a standard normal distribution and $nest^t{}_{best}$ is the position of the best nest so far.

10.2.4.4 Alien Eggs Discovery

The alien eggs discovery is preformed for all of eggs using of the probability matrix for each component of each solution. Existing eggs are replaced considering quality by newly generated ones from their current position by random walks with step size such as [7]:

$$S = rand.(nests[permute1[i][j]] - nests[permute2[i][j]])$$
$$nest^{(t+1)} = nest^{(t)} + S.^{*}P \tag{10.16}$$

where *permute1* and *permute2* are different rows permutation functions applied to the nests matrix and P is the probability matrix which was mentioned in (10.13).

10.2.4.5 Termination Criterion

The generation of new Cuckoos and the discovering of the alien eggs steps are performed alternately until a termination criterion is satisfied. The maximum number of structure analyses is considered as algorithm's termination criterion.

10.2.5 Design Examples

In this section, common truss optimization examples as benchmark problems are optimized with the CS algorithm. The final results are compared to the solutions of other methods to demonstrate the efficiency of the CS. We have tried to vary the number of host nests (or the population size of n) and the probability pa. From our simulations, we found that $n = 7$–20 and $pa = 0.15$–0.35 are efficient for design examples. The examples contain a 25-bar transmission tower and a 72-bar spatial truss with both discrete and continuous design variables and a dome shaped space truss with continuous search space.

10.2.5.1 A 25-Bar Space Truss

The 25-bar transmission tower is used widely in structural optimization to verify various metaheuristic algorithms. The topology and nodal numbering of a 25-bar space truss structure is shown in Fig. 10.2. The material density is considered as 0.1 lb/in^3 (2,767.990 kg/m^3) and the modulus of elasticity is taken as 10^7 psi (68,950 MPa). Twenty-five members are categorized into eight groups, as follows: (1) A_1, (2) A_2–A_5, (3) A_6–A_9, (4) A_{10}–A_{11}, (5) A_{12}–A_{13}, (6) A_{14}–A_{17}, (7) A_{18}–A_{21}, and (8) A_{22}–A_{25}. In this example, designs for both a single and multiple load cases using both discrete and continuous design variables are performed. The parameters of the CS algorithm are considered to be pa = 0.15, number of nests = 10 and the maximum number of analyses = 14,000 as the stopping criterion.

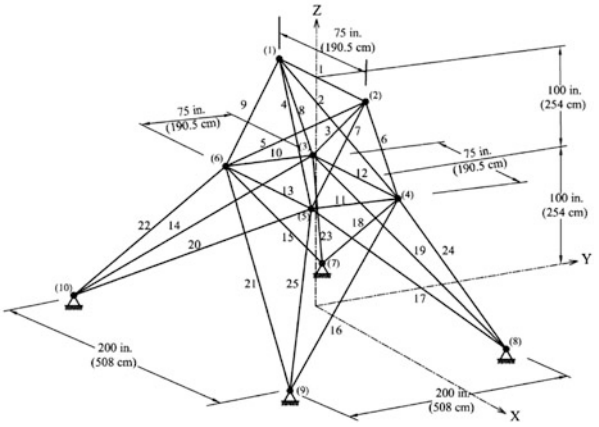

Fig. 10.2 Schematic of a 25-bar space truss

10.2.5.2 Design of a 25-Bar Truss Utilizing Discrete Variables

In the first design of a 25-bar truss, a single load case {(kips) (kN)} is applied to the structure, at nodes 1, 2, 3 and 4 as follows: 1{(0, −10, −10) (0, −44.5, −44.5)}, 2{(1, −10, −10) (4.45, −44.5, −44.5)}, 3{(0.6, 0, 0) (2.67, 0, 0)} and 4{(0.5, 0, 0) (2.225, 0, 0)}. The allowable stresses and displacements are respectively ±40 ksi (275.80 MPa) for each member and ±0.35 in (±8.89 mm) for each node in the x, y and z directions. The range of discrete cross-sectional areas is from 0.1 to 3.4 in^2 (0.6452 to 21.94 cm^2) with 0.1 in^2 (0.6452 cm^2) increment (resulting in 34 discrete cross sections) for each of the eight element groups [8].

The CS algorithm achieves the best solution weighted by 484.85 lb (2,157.58 N), after 2,000 analyses. Although, this is identical to the best design developed using BB-BC algorithm [8] and a multiphase ACO procedure [9], it performs better than others when the number of analyses and average weight for 100 runs are compared. Table 10.1 presents the performance of the CS and other heuristic algorithms.

10.2.5.3 Design of a 25-Bar Truss Utilizing Continuous Variables

In the second design of a 25-bar truss, the structure is subjected to two load cases listed in Table 10.2. Maximum displacement limitations of ±0.35 in (±8.89 mm) are imposed on every node in every direction and the axial stress constraints vary for each group as shown in Table 10.3. The range of cross-sectional areas varies from 0.01 to 3.4 in^2 (0.06452 to 21.94 cm^2) [10].

Table 10.4 shows the best solution vectors, the corresponding weights, average weights and the required number of analyses for present algorithm and some other metaheuristic algorithms. The best result obtained by IACS algorithm [12] in the aspects of low weight and number of analyses. The CS-based algorithm needs 6,100 analyses to find the best solution while this number is equal to 9,596, 15,000, 9,875, 12,500 and 7,000 analyses for a PSO-based algorithm, HS algorithm [11], a

Table 10.1 Performance comparison for the 25-bar spatial truss under single load case

		Optimal cross-sectional areas (in^2)					
		GA	GA	ACO	BB-BC phase 1, 2	Present work [4]	
Element group		[8]	[8]	[9]	[8]	in^2	cm^2
1	A_1	0.10	0.10	0.10	0.10	0.10	0.645
2	A_2–A_5	1.80	0.50	0.30	0.30	0.30	1.935
3	A_6–A_9	2.30	3.40	3.40	3.40	3.40	21.935
4	A_{10}–A_{11}	0.20	0.10	0.10	0.10	0.10	0.645
5	A_{12}–A_{13}	0.10	1.90	2.10	2.10	2.10	13.548
6	A_{14}–A_{17}	0.80	0.90	1.00	1.00	1.00	6.452
7	A_{18}–A_{21}	1.80	0.50	0.50	0.50	0.50	3.226
8	A_{22}–A_{25}	3.00	3.40	3.40	3.40	3.40	21.935
Best weight (lb)		546.01	485.05	484.85	484.85	484.85	2,157.58 (N)
Average weight (lb)		N/A	N/A	486.46	485.10	485.01	2,158.29 (N)
Number of analyses		800	15,000	7,700	9,000	2,000	

Table 10.2 Loading conditions for the 25-bar spatial truss

Case	Node	F_x kips (kN)	F_y kips (kN)	Fz kips (kN)
1	1	1.0 (4.45)	10.0 (44.5)	−5.0 (−22.25)
	2	0.0	10.0	−5.0 (−22.25)
	3	0.5 (2.225)	0.0	0.0
	6	0.5 (2.225)	0.0	0.0
2	1	0.0	20.0 (89)	−5.0 (−22.25)
	2	0.0	−20.0 (−89)	−5.0 (−22.25)

Table 10.3 Member stress limitation for the 25-bar space truss

Element group		Compression ksi (MPa)	Tension ksi (MPa)
1	A_1	35.092 (241.96)	40.0 (275.80)
2	A_2–A_5	11.590 (79.913)	40.0 (275.80)
3	A_6–A_9	17.305 (119.31)	40.0 (275.80)
4	A_{10}–A_{11}	35.092 (241.96)	40.0 (275.80)
5	A_{12}–A_{13}	35.092 (241.96)	40.0 (275.80)
6	A_{14}–A_{17}	6.759 (46.603)	40.0 (275.80)
7	A_{18}–A_{21}	6.959 (47.982)	40.0 (275.80)
8	A_{22}–A_{25}	11.082 (76.410)	40.0 (275.80)

combination algorithm based on PSO, ACO and HS [13], an improved BB–BC method using PSO properties [14] and the CSS algorithm [10], respectively. The difference between the result of the CS and these algorithms are very small, but the average weight obtained by the CS algorithm for 100 runs is better than others. The convergence history for best result and average weight of 100 runs are shown in Fig. 10.3. The important point is that although the CS requires 6,100 analyses to achieve the 545.17 lb (2,426.02 N), it can achieve the 545.76 lb (2,428.63 N) after 2,700 analyses, because CS uses the exploration step in terms of Lévy flights. If the search space is large, Lévy flights are usually more efficient.

Table 10.4 Performance comparison for the 25-bar spatial truss under multiple load cases

Element group	Optimal cross-sectional areas (in^2)							
	PSO [10]	HS [11]	IACS [12]	HPSACO [13]	HBB–BC [14]	CSS [10]	Present work [4]	
							in^2	cm^2
1 A_1	0.010	0.047	0.010	0.010	0.010	0.010	0.01	0.065
2 A_2–A_5	2.121	2.022	2.042	2.054	1.993	2.003	1.979	12.765
3 A_6–A_9	2.893	2.950	3.001	3.008	3.056	3.007	3.005	19.386
4 A_{10}–A_{11}	0.010	0.010	0.010	0.010	0.010	0.010	0.01	0.065
5 A_{12}–A_{13}	0.010	0.014	0.010	0.010	0.010	0.010	0.01	0.065
6 A_{14}–A_{17}	0.671	0.688	0.684	0.679	0.665	0.687	0.686	4.428
7 A_{18}–A_{21}	1.611	1.657	1.625	1.611	1.642	1.655	1.679	10.830
8 A_{22}–A_{25}	2.717	2.663	2.672	2.678	2.679	2.660	2.656	17.134
Best weight (lb)	545.21	544.38	545.03	544.99	545.16	545.10	545.17	2,426.02 (N)
Average weight (lb)	546.84	N/A	545.74	545.52	545.66	545.58	545.18	2,426.05 (N)
Number of analyses	9,596	15,000	3,254	9,875	12,500	7,000	6,100	

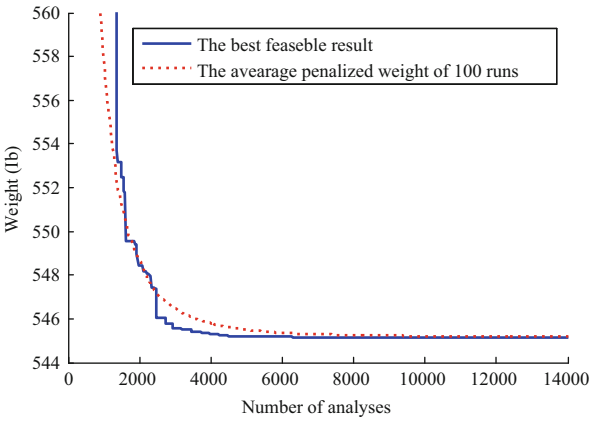

Fig. 10.3 Convergence history of the 25-bar space truss under multiple load cases [4]

10.2.5.4 A 72-Bar Space Truss

For the 72-bar spatial truss structure shown in Fig. 10.4 taken from [14], the material density is 0.1 lb/in^3 (2,767.990 kg/m^3) and the modulus of elasticity is 10^7 psi (68,950 MPa). The 72 structural members of this spatial truss are categorized into 16 groups using symmetry: (1) A_1–A_4, (2) A_5–A_{12}, (3) A_{13}–A_{16}, (4) A_{17}–A_{18}, (5) A_{19}– A_{22}, (6) A_{23}–A_{30}, (7) A_{31}–A_{34}, (8) A_{35}–A_{36}, (9) A_{37}–A_{40}, (10) A_{41}–A_{48}, (11) A_{49}–A_{52}, (12) A_{53}–A_{54}, (13) A_{55}–A_{58}, (14) A_{59}–A_{66} (15), A_{67}– A_{70}, and (16) A_{71}–A_{72}. In this example, designs for a multiple load cases using both discrete and continuous design variables are performed. The values and directions of the

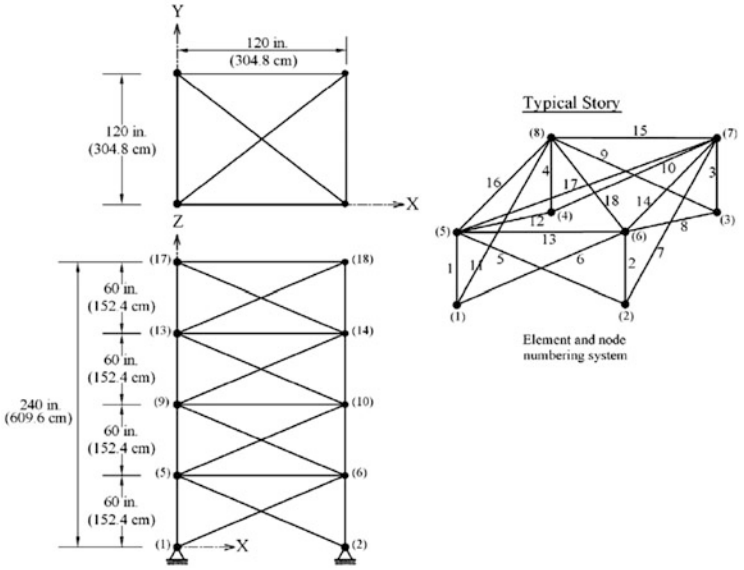

Fig. 10.4 Schematic of a 72-bar space truss

two load cases applied to the 72-bar spatial truss for both discrete and continuous designs are listed in Table 10.5. The members are subjected to the stress limits of ±25 ksi (±172.375 MPa) for both discrete and continuous designs. Maximum displacement limitations of ±0.25 in (±6.35 mm), are imposed on every node in every direction and on the uppermost nodes in both x and y directions respectively for discrete and continuous cases. In this example, the parameters of the CS algorithm are considered to be pa = 0.15 and number of nests = 7, maximum number of analyses = 21,000.

10.2.5.5 Design of a 72-Bar Truss Using Discrete Variables

In this case, the discrete variables are selected from 64 discrete values from 0.111 to 33.5 in^2 (71.613 to 21,612.860 mm^2). For more information, the reader can refer to Table 10.2 in Kaveh and Talatahari [15].

Table 10.6 shows the best solution vectors, the corresponding weights and the required number of analyses for present algorithm and some other metaheuristic algorithms. The CS algorithm can find the best design among the other existing studies. Although the number of required analyses by the CS algorithm is slightly more than ICA algorithm, but the best weight of the CS algorithm is 389.87 lb (1,734.93 N) that is 2.97 lb (13.22 N) lighter than the best result obtained by ICA algorithm [15].

Table 10.5 Multiple loading conditions for the 72-bar truss

Case	Node	F_x kips (kN)	F_y kips (kN)	F_z kips (kN)
1	17	0.0	0.0	−5.0 (−22.25)
	18	0.0	0.0	−5.0 (−22.25)
	19	0.0	0.0	−5.0 (−22.25)
	20	0.0	0.0	−5.0 (−22.25)
2	17	5.0 (22.25)	5.0 (22.25)	−5.0 (−22.25)

Table 10.6 Performance comparison for the 72-bar spatial truss with discrete variables

		Optimal cross-sectional areas (in^2)						
		GA	PSOPC	HPSO	HPSACO	ICA	Present work [4]	
Element group		[15]	[15]	[15]	[16]	[15]	in^2	cm^2
1	A_1–A_4	0.196	4.490	4.970	1.800	1.99	1.800	11.613
2	A_5–A_{12}	0.602	1.457	1.228	0.442	0.442	0.563	3.632
3	A_{13}–A_{16}	0.307	0.111	0.111	0.141	0.111	0.111	0.716
4	A_{17}–A_{18}	0.766	0.111	0.111	0.111	0.141	0.111	0.716
5	A_{19}–A_{22}	0.391	2.620	2.880	1.228	1.228	1.266	8.168
6	A_{23}–A_{30}	0.391	1.130	1.457	0.563	0.602	0.563	3.632
7	A_{31}–A_{34}	0.141	0.196	0.141	0.111	0.111	0.111	0.716
8	A_{35}–A_{36}	0.111	0.111	0.111	0.111	0.141	0.111	0.716
9	A_{37}–A_{40}	1.800	1.266	1.563	0.563	0.563	0.563	3.632
10	A_{41}–A_{48}	0.602	1.457	1.228	0.563	0.563	0.442	2.852
11	A_{49}–A_{52}	0.141	0.111	0.111	0.111	0.111	0.111	0.716
12	A_{53}–A_{54}	0.307	0.111	0.196	0.250	0.111	0.111	0.716
13	A_{55}–A_{58}	1.563	0.442	0.391	0.196	0.196	0.196	1.265
14	A_{59}–A_{66}	0.766	1.457	1.457	0.563	0.563	0.602	3.884
15	A_{67}–A_{70}	0.141	1.228	0.766	0.442	0.307	0.391	2.523
16	A_{71}–A_{72}	0.111	1.457	1.563	0.563	0.602	0.563	3.632
Weight (lb)		427.203	941.82	933.09	393.380	392.84	389.87	1,734.93 (N)
Number of analyses		N/A	150,000	50,000	5,330	4,500	4,840	

10.2.5.6 Design of a 72-Bar Truss Using Continuous Variables

In this case the minimum value for the cross-sectional areas is 0.1 in^2 (0.6452 cm^2) and the maximum value is limited to 4.00 in^2 (25.81 cm^2).

The CS algorithm achieves the best result among other algorithms in the aspects of weight, number of required analyses and the average weight of 100 runs. The convergence history of the best result and the average weight of 100 runs are shown in Fig. 10.5. Notice that as shown in this figure, although the CS requires 10,600 analyses to achieve 379.63 lb (1,689.37 N), but achieves the 380 lb (1,691 N) possible design after 4,900 analyses. Table 10.7 compares the results of the CS to those of the previously reported methods in the literature.

For Further studies of one of two CS parameters we have tried this example alternatively for constant number of nests as 7 and various amounts of *pa* from the [0, 1] interval with 21,000 as the maximum number of analyses. The convergence history of the average weight for 100 runs is shown in Fig. 10.6. According to this

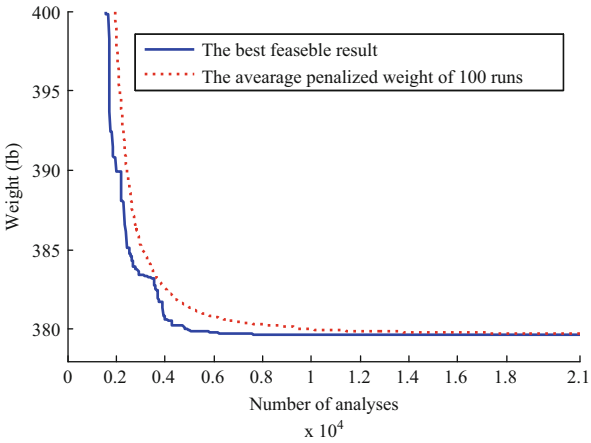

Fig. 10.5 Convergence history of the 72-bar space truss with continious variables [4]

Table 10.7 Performance comparison for the 72-bar spatial truss with continuous variables

		Optimal cross-sectional areas (in^2)						
		GA	ACO	PSO	BB–BC	HBB–BC	Present work [4]	
Element group		[14]	[9]	[14]	[8]	[14]	in^2	cm^2
1	A_1–A_4	1.755	1.948	1.7427	1.8577	1.9042	1.9122	12.055
2	A_5–A_{12}	0.505	0.508	0.5185	0.5059	0.5162	0.5101	3.267
3	A_{13}–A_{16}	0.105	0.101	0.1000	0.1000	0.1000	0.1000	0.646
4	A_{17}–A_{18}	0.155	0.102	0.1000	0.1000	0.1000	0.1000	0.645
5	A_{19}–A_{22}	1.155	1.303	1.3079	1.2476	1.2582	1.2577	8.487
6	A_{23}–A_{30}	0.585	0.511	0.5193	0.5269	0.5035	0.5128	3.343
7	A_{31}–A_{34}	0.100	0.101	0.1000	0.1000	0.1000	0.1000	0.645
8	A_{35}–A_{36}	0.100	0.100	0.1000	0.1012	0.1000	0.1000	0.646
9	A_{37}–A_{40}	0.460	0.561	0.5142	0.5209	0.5178	0.5229	3.197
10	A_{41}–A_{48}	0.530	0.492	0.5464	0.5172	0.5214	0.5177	3.345
11	A_{49}–A_{52}	0.120	0.100	0.1000	0.1004	0.1000	0.1000	0.648
12	A_{53}–A_{54}	0.165	0.107	0.1095	0.1005	0.1007	0.1000	0.645
13	A_{55}–A_{58}	0.155	0.156	0.1615	0.1565	0.1566	0.1566	1.013
14	A_{59}–A_{66}	0.535	0.550	0.5092	0.5507	0.5421	0.5406	3.492
15	A_{67}–A_{70}	0.480	0.390	0.4967	0.3922	0.4132	0.4152	2.839
16	A_{71}–A_{72}	0.520	0.592	0.5619	0.5922	0.5756	0.5701	3.486
Weight (lb)		385.76	380.24	381.91	379.85	379.66	379.63	1,689.37 (N)
Average weight (lb)		N/A	383.16	N/A	382.08	381.85	379.73	1,689.80 (N)
Number of analyses		N/A	18,500	N/A	19,621	13,200	10,600	

figure, the values from [0.15, 0.35] are more efficient for the performance of the algorithm and 0.15 gives the best result among others.

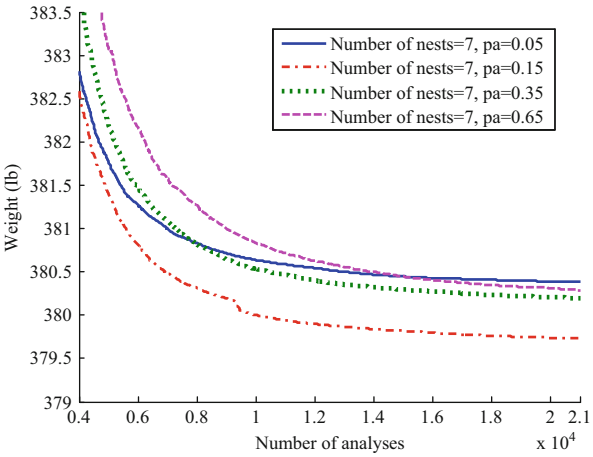

Fig. 10.6 Convergence history for the average weight of 100 runs, with constant number of nests and different values of *pa* [4]

10.2.5.7 Design of a 120-Bar Dome Shaped Truss

The topology, nodal numbering and element grouping of a 120-bar dome truss are shown in Fig. 10.7. For clarity, not all the element groups are numbered in this figure. The 120 members are categorized into seven groups, because of symmetry. Other conditions of problem are as follows [8], the modulus of elasticity is 30,450 ksi (210,000 MPa) and the material density is 0.288 lb/in^3 (7,971.810 kg/m^3). The yield stress of steel is taken as 58.0 ksi (400 MPa). The dome is considered to be subjected to vertical loading at all the unsupported joints. These loads are taken as −13.49 kips (−60 kN) at node 1, −6.744 kips (−30 kN) at nodes 2 through 14, and −2.248 kips (−10 kN) at the rest of the nodes. The minimum cross-sectional area of all members is 0.775 in^2 (5 cm^2) and the maximum cross-sectional area is taken as 20.0 in^2 (129.032 cm^2). The constraints are stress constraints [as defined by (10.5) and (10.6)] and displacement limitations of ±0.1969 in (±5 mm), imposed on all nodes in x, y and z directions.

In this example, the parameters of the CS algorithm are considered to be $pa = 0.15$, the number of nests = 7 and the maximum number of analyses = 21,000. Table 10.8 shows the best solution vectors, the corresponding weights and the required number of analyses for convergence of the present algorithm and some other metaheuristic algorithms. The CS-based algorithm needs 6,300 analyses to find the best solution while this number is equal to 150,000, 32,600, 10,000, 10,000, 7,000 and 6,000 analyses for a PSO-based algorithm [13], a PSO and ACO hybrid algorithm [13], a combination algorithm based on PSO, ACO and HS [13], an improved BB–BC method using PSO properties [14], the CSS algorithm [10] and the ICA algorithm [17], respectively. As a result, the CS optimization algorithm has second best convergence rates among the considered metaheuristics and its difference with the ICA is only 300 analyses. Comparing the final results of the CS and those of the other metaheuristics shows that CS finds the second best result while the difference between the result of the CS and that obtained by the HPSACO

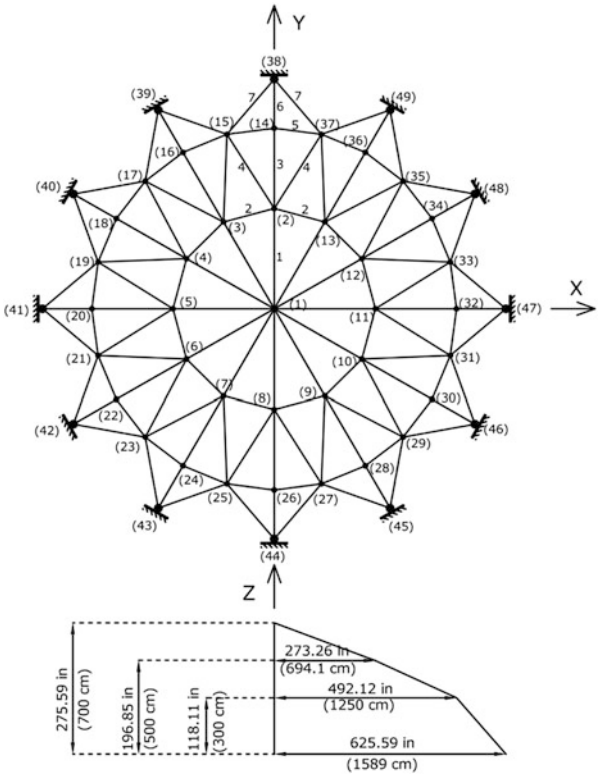

Fig. 10.7 Schematic of a 120-bar dome shaped truss

[13], as the first best result, is very small. A comparison of the allowable and existing stresses and displacements of the 120-bar dome truss structure using CS is shown in Fig. 10.8. The maximum value for displacement is equal to 0.1969 in (5 mm) and the maximum stress ratio is equal to 99.99 %.

10.2.6 Discussions

A version of cuckoo search algorithm via Lévy flights is applied to optimum design of truss structures using both discrete and continuous design variables. Looking at the CS algorithm carefully, one can observe essentially three components: selection of the best, exploitation by local random walk, and exploration by randomization via Lévy flights globally. In order to sample the search space effectively so that the newly generated solutions be diverse enough, the CS uses the exploration step in terms of Lévy flights. In contrast, most metaheuristic algorithms use either uniform distributions or Gaussian to generate new explorative moves. For large search spaces the Lévy flights are usually more efficient.

Table 10.8 Performance comparison for the 120-bar dome shaped truss with continuous variables

		Optimal cross-sectional areas (in^2)							
		PSOPC	PSACO	HPSACO	HBB–BC	CSS	ICA	Present work [4]	
Element group		[13]	[13]	[13]	[14]	[10]	[17]	in^2	cm^2
1	A1	3.040	3.026	3.095	3.037	3.027	3.0275	3.0244	19.512
2	A_2	13.149	15.222	14.405	14.431	14.606	14.4596	14.7168	94.947
3	A_3	5.646	4.904	5.020	5.130	5.044	5.2446	5.0800	32.774
4	A_4	3.143	3.123	3.352	3.134	3.139	3.1413	3.1374	20.241
5	A_5	8.759	8.341	8.631	8.591	8.543	8.4541	8.5012	54.847
6	A_6	3.758	3.418	3.432	3.377	3.367	3.3567	3.3019	21.303
7	A_7	2.502	2.498	2.499	2.500	2.497	2.4947	2.4965	16.106
Best weight (lb)		33,481.2	33,263.9	33,248.9	33,287.9	33,251.9	33,256.2	33,250.42	147,964.37 (N)
Average weight(lb)		N/A	N/A	N/A	N/A	N/A	N/A	33,253.28	147,977.10 (N)
Number of analyses		150,000	32,600	10,000	10,000	7,000	6,000	6,300	

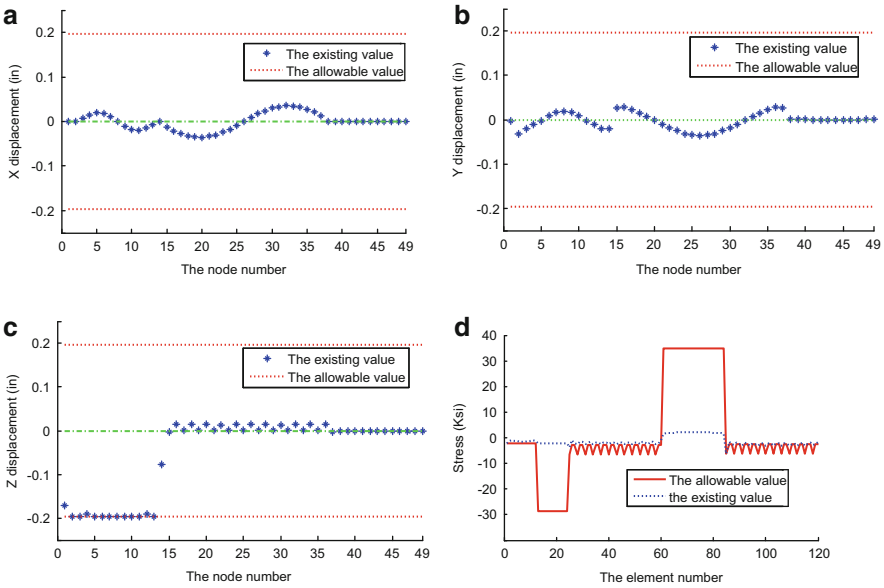

Fig. 10.8 Comparison of the allowable and existing constraints for the 120-bar dome shaped truss using the CS [4] (**a**) Displacement in the x direction, (**b**) Displacement in the y direction, (**c**) Displacement in the z direction, (**d**) Stresses

Unique characteristics of the CS algorithm over other metaheuristic methods are its simplified numerical structure and its dependency on a relatively small number of parameters to define and determine - or limit- the algorithm's performance. In fact, apart from the step size parameter α and the population size n, there is essentially one parameter pa.

Three design examples consisting of two space trusses with continuous and discrete design variables and a dome-shaped truss with continuous search space are studied to illustrate the efficiency of the present algorithm. The comparisons of the numerical results of these structures utilizing the CS and those obtained by other optimization methods are carried out to demonstrate the robustness of the present algorithm in terms of good results and number of analyses together. The most noticeable result obtained by the CS is that the average weight of 100 runs is better than other algorithms.

10.3 Optimum Design of Steel Frames

In this section, optimum design of two dimensional steel frames for discrete variables based on the Cuckoo search algorithm is developed. The design algorithm is supposed to obtain minimum weight frame through suitable selection of sections from a standard set of steel sections such as American Institute of Steel

Construction (AISC) wide-flange (W) shapes. Strength constraints of AISC load and resistance factor design specification and displacement constraints are imposed on frames.

10.3.1 Optimum Design of Planar Frames

The aim of optimizing the frame weight is to find a set of design variables that has the minimum weight satisfying certain constraints. This can be expressed as:

$$\begin{aligned} &\text{Find} \quad \{x\} = \left[x_1, x_2, ..., x_{ng}\right], x_i \in D_i \\ &\text{to} \quad \text{minimize} \quad \mathrm{w}\left(\{\mathrm{x}\}\right) = \sum_{\mathrm{i}=1}^{\mathrm{nm}} p_{\mathrm{i}}.x_i.L_i \\ &\text{subject to :} \quad \mathrm{g_i}\left(\{x\}\right) \leq 0, \;\; j = 1, 2, \ldots, \mathrm{n} \end{aligned} \tag{10.17}$$

where $\{x\}$ is the set of design variables; ng is the number of member groups in structure (number of design variables); D_i is the allowable set of values for the design variable x_i; $w(\{x\})$ presents weight of the structure; nm is the number of members of the structure; p_i denotes the material density of member i; L_i and x_i are the length and the cross sectional area of member i, respectively; $g_i(\{x\})$ denotes design constraints include strength constraints of the American Institute of Steel Construction load and resistance factor design (AISC [18]) and displacement constraints; and n is the number of the constraints. D_i can be considered either as a continuous set or as a discrete one. If the design variables represent a selection from a set of parts as

$$D_i = \left(d_{i,1}, d_{i,2}, \ldots, d_{i,r(i)}\right) \tag{10.18}$$

Then the problem can be considered as a discrete one, where $r(i)$ is the number of available discrete values for the ith design variable.

In this study, an implementation of penalty approach is used to account for constraints. In this method, the aim of the optimization is redefined by introducing the cost function as:

$$f_{\cos t}(\{X\}) = (1 + \varepsilon_1.\upsilon)^{\varepsilon_2} \times W(\{X\}), \quad \upsilon = \sum_{j=1}^{n} \max[0, g_i(\{x\})] \tag{10.19}$$

where n represents the number of evaluated constraints for each individual design, and υ denotes the sum of the violations of the design. The constant ε_1 and ε_2 are selected considering the exploration and the exploitation rate of the search space. Here, ε_1 is set to unity; ε_2 is selected in a way that it decreases the penalties and reduces the cross-sectional areas. Thus, in the first steps of the search process, ε_2 is set to 1.5 and ultimately increased to 3.

Design constraints according to LRFD-AISC requirements can be summarized as follows:

Maximum lateral displacement:

$$\frac{\Delta_T}{H} - R \leq 0 \tag{10.20}$$

Inter-story drift constraints:

$$\frac{d_i}{h_i} - R_I \leq 0, i = 1, 2, \ldots, ns \tag{10.21}$$

Strength constraints:

$$\begin{aligned} &\frac{P_u}{2\phi P_n} + \left(\frac{M_{ux}}{\phi_b M_{nx}} + \frac{M_{uy}}{\phi_b M_{ny}}\right) - 1 \leq 0, \quad for \frac{P_u}{\phi_c P_n} \langle 0.2 \\ &\frac{P_u}{\phi_c P_n} + \frac{8}{9}\left(\frac{M_{ux}}{\phi_b M_{nx}} + \frac{M_{uy}}{\phi_b M_{ny}}\right) - 1 \leq 0, \quad for \frac{P_u}{\phi_c P_n} \langle 0.2 \end{aligned} \tag{10.22}$$

where Δ_T is the maximum lateral displacement; H is the height of the frame structure; R is the maximum drift index (1/300); d_i is the inter-story drift; h_i is the story height of the ith floor; ns is the total number of stories; R_I presents the inter-story drift index permitted by the code of the practice (1/300); P_u is the required strength (tension or compression); P_n is the nominal axial strength (tension or compression); ϕ_c is the resistance factor ($\phi_c = 0.9$ for tension, $\phi_c = 0.85$ for compression); M_{ux} and M_{uy} are the required flexural strengths in the x and y directions, respectively; M_{nx} and M_{ny} are the nominal flexural strengths in the x and y directions (for two-dimensional structures, $M_{ny} = 0$); and ϕ_b denotes the flexural resistance reduction factor ($\phi_c = 0.90$) The nominal tensile strength for yielding in the gross section is computed as

$$P_n = A_g.F_y \tag{10.23}$$

and the nominal compressive strength of a member is computed as

$$P_n = A_g.F_{cr} \tag{10.24}$$

$$F_{cr} = \left(0.658^{\lambda_c^2}\right)F_y, \quad for \lambda_c \leq 1.5$$
$$F_{cr} = \left(\frac{0.877}{\lambda_c^2}\right)F_y, \quad for \lambda_c \rangle 1.5 \tag{10.25}$$

$$\lambda_c = \frac{kl}{r\pi}\sqrt{\frac{F_y}{E}} \tag{10.26}$$

where A_g is the cross-sectional area of a member and k is the effective length factor determined by the approximated formula based on Dumonteil [19].

10.3.2 Optimum Design of Steel Frames Using Cuckoo Search Algorithm

Before initiating optimization process, it is necessary to set the search space. The steel members used for the design of steel frames, consist of 267 W-shaped sections from the AISC-LRFD database starting from W44 × 335 to W4 × 13. These sections with their properties are used to prepare a design pool. The sequence numbers assigned to this pool that sorted with respect to area of sections are considered as design variables. In other words the design variables represent a selection from a set of integer numbers between 1 and the number of sections. The pseudo code of optimum design algorithm is identical to that of Sect. 10.2.4.

10.3.3 Design Examples

In this section, three steel frames are optimized using the CS algorithm as benchmark problems. To investigate the effect of the initial solution on the final results, each example is solved independently several times with random initial designs due to the stochastic nature of the algorithm. The proposed algorithm is coded in MATLAB and structures are analyzed using the direct stiffness method. The final results are compared to the solutions of other methods to demonstrate the efficiency of the present approach. First example is also used for adjusting algorithm parameters and the obtained results are used in other examples.

10.3.3.1 A One-Bay Ten-Story Frame

Figure 10.9 shows the topology, the service loading conditions and the numbering of member groups for a one-bay ten-story frame. The element grouping results in four beam sections and five column sections for a total of nine design variables. Beam element groups were chosen from 267 W-sections, and column groups were selected from only W14 and W12 sections.

The AISC-LRFD combined strength constraints and a displacement constraint of inter-story drift $<$ story height/300 are the performance constraints of this frame. The material has a modulus of elasticity equal to $E = 29{,}000$ ksi (200 GPa) and a

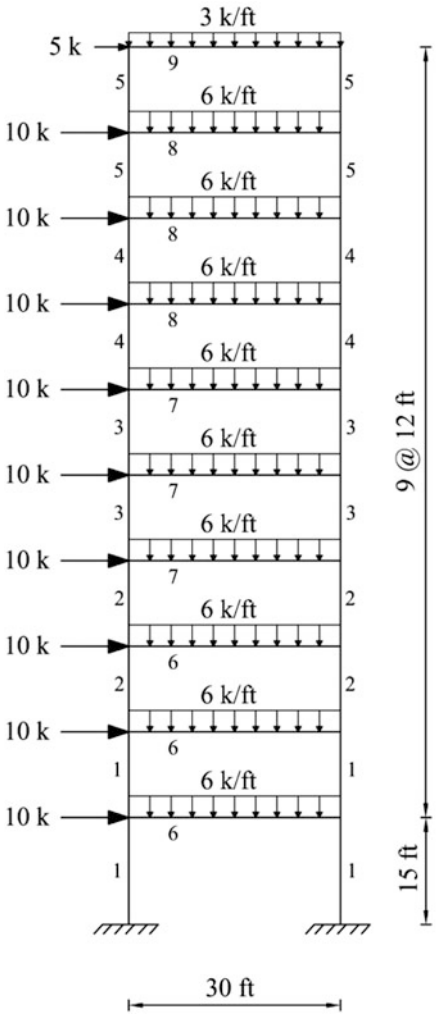

Fig. 10.9 Schematic of a one-bay ten-story frame

yield stress of $f_y = 36$ ksi (248.2 MPa). The effective length factors of the members are calculated as $K_x \geq 1.0$ using the approximate equation proposed by Dumonteil [19], for a sway-permitted frame and the out-of-plane effective length factor is specified as $K_y = 1.0$. Each column is considered as non-braced along its length, and the non-braced length for each beam member is specified as one-fifth of the span length.

Based on Yang's simulations [1], considering algorithm parameters (population size or number of host nests (n) and probability pa) such as $n = 15$–25 and $pa = 0.15$–0.3 is efficient for most optimization problems. In order to adjust probability pa for the two-dimensional steel frame optimization problem, we solve this example alternatively with a constant n equal to10 and various amounts

Table 10.9 Performance of the CS for one-bay ten-story frame with various amounts of *pa*

	Best run		100 runs	
CS parameters	Min weight (lb)	Min *Noa*	Weight (lb)	*Noa*
Pa = 0.1, n = 10	62,074.368	9,320	62,195.35 ± 188.77 (45 %)	17,400 ± 5,900 (45 %)
Pa = 0.2, n = 10	62,074.368	6,040	62,133.05 ± 149.07 (74 %)	16,480 ± 5,600 (74 %)
Pa = 0.25, n = 10	62,074.368	6,980	62,111.01 ± 130.74 (81 %)	14,360 ± 5,360 (81 %)
Pa = 0.3, n = 10	62,074.368	7,100	62,088.25 ± 41.66 (90 %)	14,640 ± 4,520 (90 %)
Pa = 0.4, n = 10	62,074.368	6,280	62,110.47 ± 60.9 (74 %)	13,780 ± 3,800 (74 %)
Pa = 0.5, n = 10	62,074.368	8,820	62,156.84 ± 173.54 (63 %)	15,120 ± 4,580 (63 %)

of *pa* within the [0, 1] interval with 30,000 as the maximum number of analyses. The results are summarized in Table 10.9, where the second and third columns contain minimum weight and minimum number of analyses (Min *Noa*) for best runs, respectively. Two other columns show the results of 100 runs for the obtained optimal weight and the number of analyses in the format: average ± one standard deviation (success rate) (Yang [1]). Therefore, 62,088.25 ± 41.66 (90 %) means that the average optimal weight is 62,088.25 lb with a standard deviation of 41.66 lb, and the success rate of all runs in finding the best obtained weight is 90 %. As it is shown, the amounts from 0.1 to 0.3 are efficient for algorithm and the $pa = 0.3$ $pa = 0.3$ gives the best result.

In order to adjust the population size, we design this frame with constant *pa* equal to 0.3 and various *n* values within the [5] interval with 30,000 as the maximum number of analyses. Results are shown in Table 10.10 with the previous table's format for 100 independent runs. As it is demonstrated, considering the number of nests from 7 to 20 is sufficient, and $n = 7$ is the most efficient value which results the minimum number of analyses despite poor performance with respect to average optimal weight and standard deviation. The table also indicates that CS results the same best design in all cases. Overall, it seems that choosing $Pa = 0.3$ and $n = 7$ can give efficient performance of the CS for the two-dimensional steel frame optimization problem. Thus, we used these values for the present example and two subsequent ones.

This frame was studied for discrete design variables by Pezeshk et al. [20] using GA, Camp et al. [17] using ACO and Kaveh and Talatahari [21] using IACO. Table 10.11 lists the designs developed by these algorithms and the CS. The lighter design with minimum number of analyses obtained by IACO algorithm. The best design developed by CS weighted 62,074 (lb) with 4,438 as required number of analyses that is 0.4 % heavier than the lighter design obtained by IACO. The average weight and standard deviation of 100 runs (lb) are 63,308 ± 684 and 63,279 ± 618 for ACO and IACO algorithms, respectively, and 62,923 ± 1.74 for 30 runs by HS, CS results in 62,186.96 ± 240.12 for 100 runs that is better than others.

Table 10.10 Performance of the CS for one-bay ten-story frame with various amounts of n

	Best run		100 runs	
CS parameters	Min weight (lb)	Min *Noa*	Weight (lb)	*Noa*
n = 5, Pa = 0.3	62,074.368	4,560	62,672.28 ± 854.16 (41 %)	8,140 ± 4,640 (41 %)
n = 7, Pa = 0.3	62,074.368	4,438	62,186.96 ± 240.12 (58 %)	10,528 ± 3,920 (58 %)
n = 10, Pa = 0.3	62,074.368	7,100	62,088.25 ± 41.66 (90 %)	14,640 ± 4,520 (90 %)
n = 15, Pa = 0.3	62,074.368	11,610	62,109.71 ± 66.48 (76 %)	19,920 ± 4,860 (76 %)
n = 20, Pa = 0.3	62,074.368	13,280	62,116.00 ± 66.89 (71 %)	23,320 ± 3,800 (71 %)
n = 25, Pa = 0.3	62,074.368	16,400	62,177.38 ± 137.74 (45 %)	24,900 ± 3,250 (45 %)

10.3.3.2 A Three-Bay Fifteen-Story Frame

The configuration, the service loading conditions and the numbering of member groups for a three-bay fifteen-story frame is shown in Fig. 10.10. The loads are W = 6.75 kips and w_1 = 3.42 kips/ft. All 64 columns grouped into 9 groups and all 45 beams are considered as a beam element group. All element groups are chosen from 267 W-sections. Performance constraints, material properties and other conditions are the same as those of the first example. One additional constraint of displacement control is that the sway of the top story is limited to 9.25 in (23.5 cm). The parameters of algorithm are considered same as those of the first example. The maximum number of analyses is 19,600.

The frame was designed by Kaveh and Talatahari using PSO algorithm [15], hybrid PSO and BB-BC algorithm [22] and ICA algorithm [15]. Table 10.12 shows the optimal design developed by CS algorithm, a frame weighting 86,809 lb that is 7.4 % lighter than the best design obtained by ICA algorithm as best result of three other algorithms. The average optimal weight and standard deviation of 50 independent runs with random initial designs are 87,784 ± 942 lb. The convergence history for best result and penalized average weight of 50 runs are shown in Fig. 10.11, and for clarity the upper bound of y axis limited to 120 kips. It should be noted that although the CS requires 16,170 analyses to reach the lightest design, but achieves the 93,630 lb structure as a feasible design after 4,700 analyses. The maximum value of sway at the top story, stress ratio and inter-story drift are 5.39 in. 99.72 % for right corner column at 10th story and 0.46 in for 4th story, respectively.

Table 10.11 Performance comparison for the one-bay ten-story frame

Element group		Optimal W-shaped sections				
		GA	ACO	HS	IACO	Present work [5]
1	Column 1–2 S	W14 × 233	W14 × 233	W14× 211	W14 × 233	W14 × 233
2	Column 3–4 S	W14 × 176	W14 × 176	W14× 176	W14 × 176	W14 × 176
3	Column 5–6 S	W14 × 159	W14 × 145	W14× 145	W14 × 145	W14 × 132
4	Column 7–8 S	W14 × 99	W14 × 99	W14 × 90	W14 × 90	W14 × 109
5	Column 9–10 S	W12 × 79	W12 × 65	W14 × 61	W12 × 65	W14 × 61
6	Beam 1–3 S	W33 × 118	W30 × 108	W33× 118	W33 × 118	W33 × 118
7	Beam 4–6 S	W30 × 90	W30 × 90	W30 × 99	W30 × 90	W30 × 108
8	Beam 7–9 S	W27 × 84	W27 × 54	W24 × 76	W24 × 76	W24 × 55
9	Beam 10 S	W24 × 55	W21 × 44	W18 × 46	W14 × 30	W18 × 40
Best weight (lb)		65,136	62,610	61,864	61,796	62,074
No. of analyses		3,000	5,100	3,690	2,500	4,438

10.3.3.3 A Three-Bay Twenty Four-Story Frame

The last example is a three-bay twenty four-story frame shown in Fig. 10.12. Camp et al. [17], Degertekin [23], Kaveh and Talatahari [22] and Kaveh and Talatahari [15] utilized ant colony optimization, harmony search algorithm, a hybrid PSO and BB-BC, and Imperialist competitive algorithm to solve this problem, respectively The frame is designed following the LRFD specifications. The inter-story drift displacement constraint is same as the first example. The loads are W = 5,761.85 lb, $w_1 = 300$ lb/ft, $w_2 = 436$ lb/ft, $w_3 = 474$ lb/ft and $w_4 = 408$ lb/ft. The material's modulus of elasticity is E =29,732 ksi (205 GPa) and its yield stress is $f_y = 33.4$ ksi (230.3 MPa). The element grouping results in 16 column sections and 4 beam sections for a total of 20 design variables. In this example, each of the four beam element groups is chosen from all 267 W-shapes, while the 16 column element groups are limited to W14 sections (37 W-shapes). The effective length factors of the members are calculated as $K_x \geq 1.0$ for a sway-permitted frame and the out-of-plane effective length factor is specified as $K_y = 1.0$. All columns and beams are considered as non-braced along their lengths.

The optimum designs of the algorithms are listed in Table 10.13. The best design of previous works is due to ICA algorithm, weights 212,640 lb with 10,500 as the number of analyses. The lightest design of 20 independently runs for CS algorithm weights 201,451 lb that is 5.3 % lighter than developed by ICA.

The convergence history for best result and penalized average weight of 20 runs are shown in Fig. 10.13. For clarity the initial iterations of algorithm are eliminated. As depicted in this figure although the best design of the CS needs 15,918 analyses, it reaches to a feasible design of 211,979 lb with 3,528 analyses which is lighter than the design obtained by ICA. The average optimal weight for the 20 runs is 205,096 lb, and the standard deviation is 4,533 lb. The average number of analyses is 18,000. The optimal average weight and a standard deviation for 100 runs of ACO and HS algorithms is 229,555 ± 4,561 lb and 222,620 ± 5,800 lb with 15,500 and 14,651 as average number of analyses, respectively.

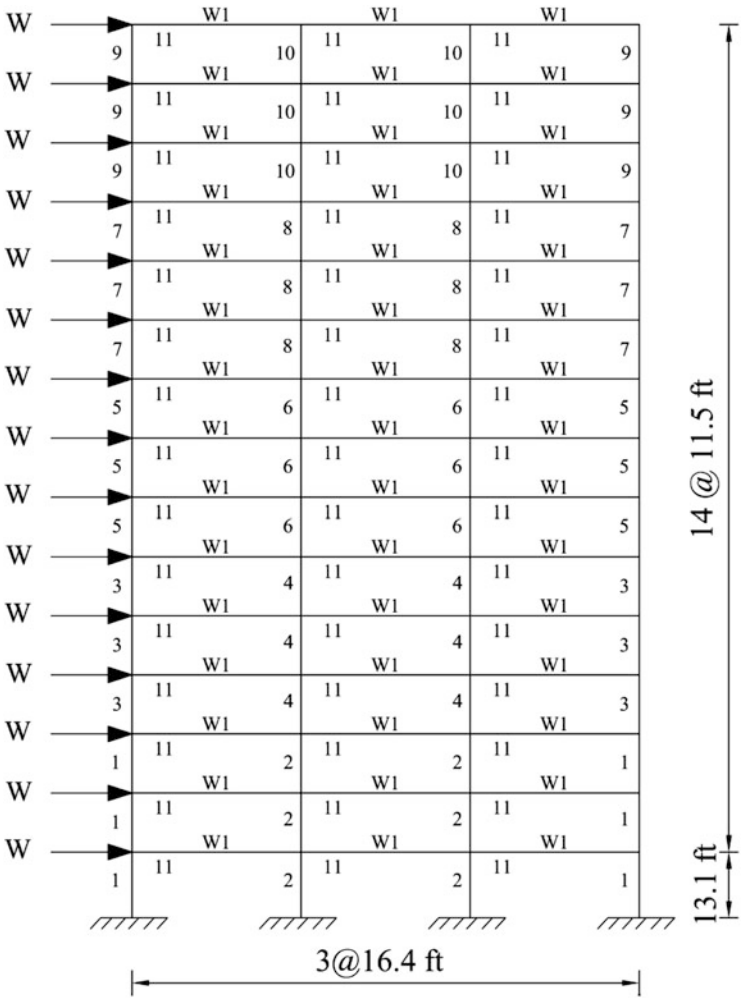

Fig. 10.10 Schematic of a three-bay fifteen-story frame [5]

The maximum value of sway at the top story, stress ratio and inter-story drift are 10.63 in. 80.18 % for right inner column at 2nd story and 0.48 in for 13th story, respectively. Figure 10.14a, b shows the inter-story drift for all the stories, stress ratio for all the members, and their upper bounds. Evidently, the inter-story drift is the dominant constraint in the frame design so that it is more than 90 % of the maximum drift between 2 to 17 stories.

Table 10.12 Performance comparison for the three-bay fifteen-story frame

	Optimal W-shaped sections			
Element group	PSO	HBB-BC	ICA	Present work [5]
1	W33 × 118	W24 × 117	W24 × 117	W 14 × 99
2	W33 × 263	W21 × 132	W21 × 147	W 27 × 161
3	W24 × 76	W12 × 95	W27 × 84	W14 × 82
4	W36 × 256	W18 × 119	W27 × 114	W 24 × 104
5	W21 × 73	W21 × 93	W14 × 74	W 12 × 65
6	W18 × 86	W18 × 97	W18 × 86	W 18 × 86
7	W18 × 65	W18 × 76	W12 × 96	W 18 × 50
8	W21 × 68	W18 × 65	W24 × 68	W 14 × 61
9	W18 × 60	W18 × 60	W10 × 39	W 8 × 24
10	W18 × 65	W10 × 39	W12 × 40	W 14 × 40
11	W21 × 44	W21 × 48	W21 × 44	W 21 × 44
Best weight (lb)	111,613	97,649	93,813	86,809
No. of analyses	50,000	9,500	6,000	16,170

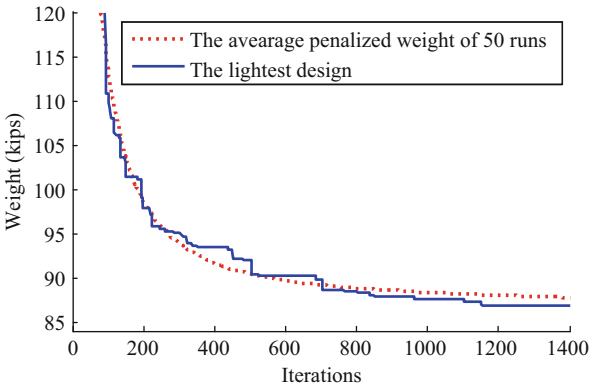

Fig. 10.11 The best and average design history of the three-bay fifteen-story frame [5]

10.3.4 Discussions

A version of Cuckoo Search algorithm via Lévy flights, which is proposed by Yang and Deb [3], is utilized to optimum design of two dimensional steel frames. The procedure of discrete design variables are performed according to AISC-LRFD specifications. The CS algorithm is comprised of three major components as following: selection of the best, exploitation by local random walk, and exploration by randomization based on Lévy flights. In order to sample the search space effectively so that the newly generated solutions to be diverse enough, the CS uses the exploration step in terms of Lévy flights. In contrast, most metaheuristic algorithms use either uniform distributions or Gaussian to generate new explorative moves. When the search space is large, Lévy flights are usually more efficient. As shown in convergence figures of examples, the ability of algorithm to local search is not very efficient as its exploration.

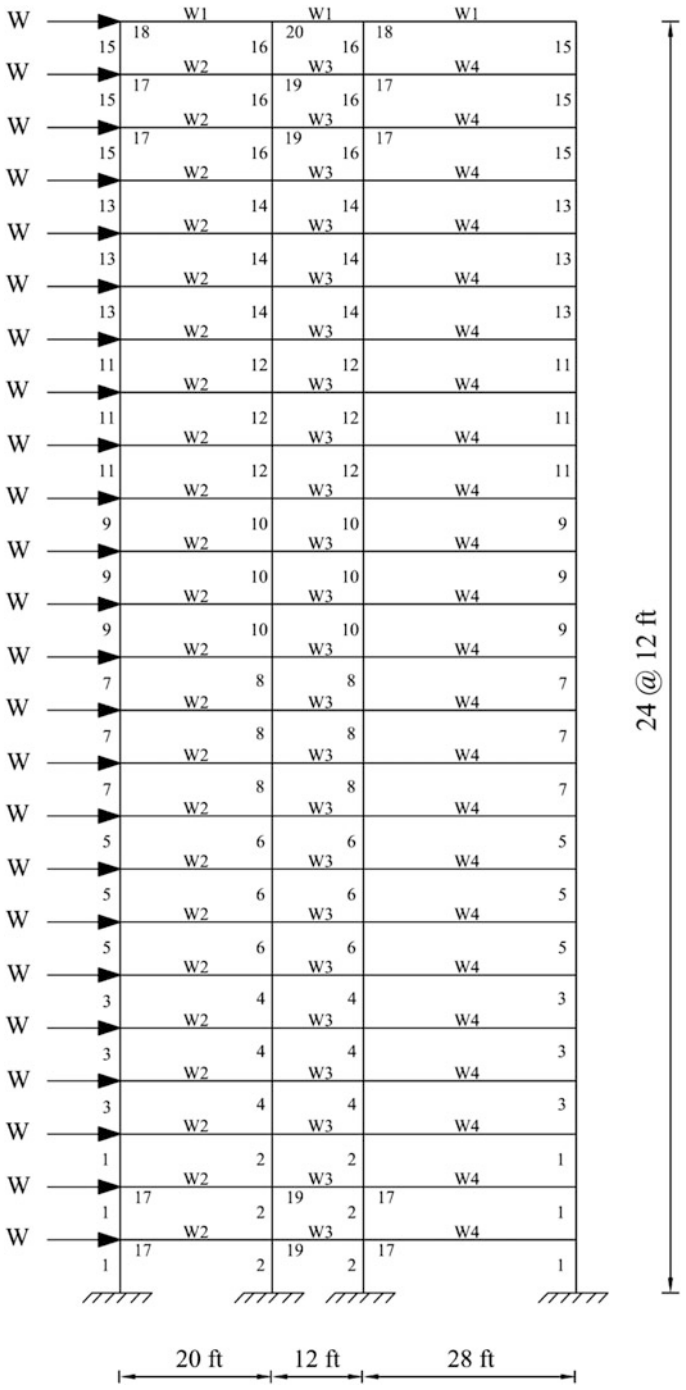

Fig. 10.12 Schematic of a three-bay twenty-four story frame

Table 10.13 Performance comparison for the three-bay twenty four-story frame

	Optimal W-shaped sections				
Element group	ACO	HS	HBB-BC	ICA	Present work [5]
1	W14× 145	W14 × 176	W14 × 176	W14 × 109	W14 × 159
2	W14 × 132	W14 × 176	W14 × 159	W14 × 159	W14 × 132
3	W14 × 132	W14 × 132	W14 × 109	W14 × 120	W14 × 99
4	W14 × 132	W14 × 109	W14 × 90	W14 × 90	W14 × 74
5	W14 × 68	W14 × 82	W14 × 82	W14 × 74	W14 × 61
6	W14 × 53	W14 × 74	W14 × 74	W14 × 68	W14 × 53
7	W14 × 43	W14 × 34	W14 × 38	W14 × 30	W14 × 34
8	W14 × 43	W14 × 22	W14 × 30	W14 × 38	W14 × 22
9	W14 × 145	W14 × 145	W14 × 159	W14 × 159	W14 × 90
10	W14 × 145	W14 × 132	W14 × 132	W14 × 132	W14 × 99
11	W14 × 120	W14 × 109	W14 × 109	W14 × 99	W14 × 99
12	W14 × 90	W14 × 82	W14 × 82	W14 × 82	W14 × 90
13	W14 × 90	W14 × 61	W14 × 68	W14 × 68	W14 × 74
14	W14 × 61	W14 × 48	W14 × 48	W14 × 48	W14 × 53
15	W14 × 30	W14 × 30	W14 × 34	W14 × 34	W14 × 34
16	W14 × 26	W14 × 22	W14 × 26	W14 × 22	W14 × 22
17	W30 × 90	W30 × 90	W30 × 90	W30 × 90	W 30× 90
18	W8 × 18	W10 × 22	W21 × 48	W21 × 50	W 6× 15
19	W24 × 55	W18 × 40	W18 × 46	W24 × 55	W 24× 55
20	W8 × 21	W12 × 16	W8 × 21	W8 × 28	W 6× 8.5
Best weight (lb)	220,465	214,860	215,933	212,640	201,451
No. of analyses	NA	9,924	10,500	7,500	15,918

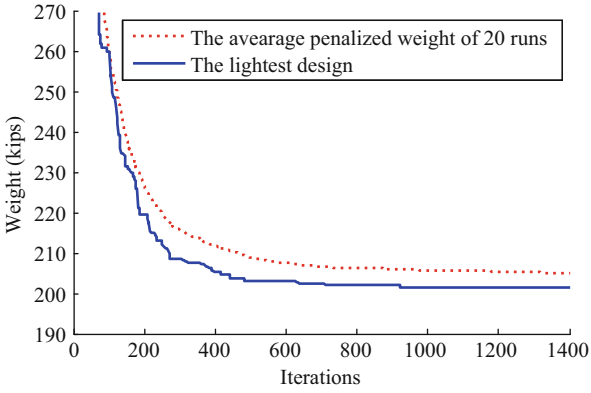

Fig. 10.13 The best and average design history of three-bay twenty four-story frame [5]

Unique characteristics of the CS algorithm over other metaheuristic methods are its simplified numerical structure and its dependency on a relatively small number of parameters, to define and determine the algorithm's performance. In fact, apart from the step size parameter α, and population size n, there is essentially one parameter pa. Simulations show that reaching the optimum designs via the later version of CS is insensitive to parameter tuning.

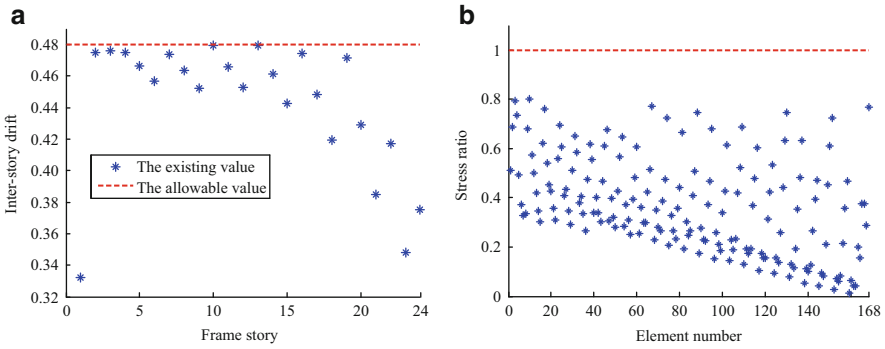

Fig. 10.14 Comparison of allowable and existing values for three-bay twenty four-bay frame using the CS [5] (**a**) Inter-story drift, (**b**) Stress ratio

Three steel frames with various number of stories and bays, are studied to illustrate the efficiency of the present algorithm. The comparisons of the numerical results obtained by CS with those by other optimization methods are carried out to demonstrate the robustness of the present algorithm in terms of reaching to best designs. According to what has been investigated, it can be interpreted that displacement constraints become dominant along the height of the structures. The most noticeable result obtained by the CS is that the performance of the algorithm in several independent runs is better than other algorithms.

References

1. Yang XS (2008) Nature-inspired metaheuristic algorithms. Luniver Press, Bristol
2. Yang XS, Deb S (2009) Cuckoo search via Lévy flights. In: Proceedings of world congress on nature and biologically inspired computing. IEEE Publications, USA, pp 210–214
3. Yang XS, Deb S (2010) Engineering optimisation by cuckoo search. Int J Math Model Numer Optim 1:330–343
4. Kaveh A, Bakhshpoori T (2013) Optimum design of space trusses using cuckoo search. Iranian J Sci Technol C1(37):1–15
5. Kaveh A, Bakhshpoori T (2013) Optimum design of steel frames using cuckoo search algorithm with Lévy flights. Struct Des Tall Build Spec Struct 22(13):1023–1036
6. American Institute of Steel Construction (AISC) (1989) Manual of steel construction–allowable stress design, 9th edn. AISC, Chicago
7. Tuba M, Subotic M, Stanarevic N (2011) Modified cuckoo search algorithm for unconstrained optimization problems. In: Proceedings of the 5th European computing conference (ECC'11), pp 263–268
8. Camp CV (2007) Design of space trusses using big bang-big crunch optimization. J Struct Eng 133:999–1008
9. Camp CV, Bichon BJ (2004) Design of space trusses using ant colony optimization. J Struct Eng 130:741–751
10. Kaveh A, Talatahari S (2010) Optimal design of skeletal structures via the charged system search algorithm. Struct Multidiscip Optim 41:893–911

11. Lee KS, Geem W (2004) A new structural optimization method based on the harmony search algorithm. Comput Struct 82:781–798
12. Kaveh A, Talatahari S (2008) Ant colony optimization for design of space trusses. Int J Space Struct 23:167–181
13. Kaveh A, Talatahari S (2009) Particle swarm optimizer, ant colony strategy and harmony search scheme hybridized for optimization of truss structures. Comput Struct 87:267–283
14. Kaveh A, Talatahari S (2009) Size optimization of space trusses using Big Bang-Big Crunch algorithm. Comput Struct 87:1129–1140
15. Kaveh A, Talatahari S (2010) Optimum design of skeletal structures using imperialist competitive algorithm. Comput Struct 88:1220–1229
16. Kaveh A, Talatahari S (2009) A particle swarm ant colony optimization for truss structures with discrete variables. J Construct Steel Res 65:1558–1568
17. Camp CV, Bichon BJ, Stovall SP (2005) Design of steel frames using ant colony optimization. J Struct Eng ASCE 131:369–379
18. AISC (2001) Manual of steel construction: load and resistance factor design. AISC, Chicago
19. Dumonteil P (1992) Simple equations for effective length factors. Eng J AISE 29(3):1115
20. Pezeshk S, Camp CV, Chen D (2000) Design of nonlinear framed structures using genetic optimization. J Struct Eng ASCE 126:382–388
21. Kaveh A, Talatahari S (2010) An improved ant colony optimization for the design of planar steel frames. Eng Struct 32:864–873
22. Kaveh A, Talatahari S (2010) A discrete Big Bang—Big Crunch algorithm for optimal design of skeletal structures. Asian J Civil Eng 11(1):103–122
23. Degertekin SO (2008) Optimum design of steel frames using harmony search algorithm. Struct Multidiscip Optim 36:393–401

Chapter 11
Imperialist Competitive Algorithm

11.1 Introduction

In this chapter an optimization method is presented based on a socio-politically motivated strategy, called Imperialist Competitive Algorithm (ICA). ICA is a multi-agent algorithm with each agent being a country, which is either a colony or an imperialist. These countries form some empires in the search space. Movement of the colonies toward their related imperialist, and imperialistic competition among the empires, form the basis of the ICA. During these movements, the powerful Imperialists are reinforced and the weak ones are weakened and gradually collapsed, directing the algorithm towards optimum points. Here, ICA is utilized to optimize the skeletal structures which is based on [1, 2].

This algorithm is proposed by Atashpaz et al. [3, 4] and is a socio-politically motivated optimization algorithm which similar to many other evolutionary algorithms starts with a random initial population. Each individual agent of an empire is called a *country*, and the countries are categorized into *colony* and *imperialist* states that collectively form *empires*. Imperialistic competitions among these empires form the basis of the ICA. During this competition, weak empires collapse and powerful ones take possession of their colonies. Imperialistic competitions direct the search process toward the powerful imperialist or the optimum points.

On the other hand, finding the optimum design of the skeletal structures is known as benchmark examples in the field of difficult optimization problems due to the presence of many design variables, large size of the search space, and many constraints. Thus, this chapter presents an ICA-based algorithm to solve optimization skeletal structures problems which can be considered as a suitable field to investigate the efficiency of the new algorithm. The chapter covers both the discrete and continuous structural design problems. Comparison of the results of the ICA with some well-known metaheuristics demonstrates the efficiency of the present algorithm.

A. Kaveh, *Advances in Metaheuristic Algorithms for Optimal Design of Structures*, 349
DOI 10.1007/978-3-319-05549-7_11, © Springer International Publishing Switzerland 2014

11.2 Optimum Design of Skeletal Structures

The aim of optimizing a structure is to find a set of design variables that has the minimum weight satisfying certain constraints. This can be expressed as

$$\begin{aligned} &\text{Find} && \{x\} = \left[x_1, x_2, \ldots, x_{ng}\right], \\ &&& x_i \in D_i \\ &\text{to minimize} && W\left(\{x\}\right) = \sum_{i=1}^{nm} \rho_i \cdot x_i \cdot L_i \\ &\text{subject to :} && g_j\left(\{x\}\right) \le 0 \qquad j = 1, 2, ..., n \end{aligned} \tag{11.1}$$

where $\{x\}$ is the set of design variables; ng is the number of member groups in structure (number of design variables); D_i is the allowable set of values for the design variable x_i; $W(\{x\})$ presents weight of the structure; nm is the number of members of the structure; ρ_i denotes the material density of member i; L_i and x_i are the length and the cross-sectional of member i, respectively; $g_j(\{x\})$ denotes design constraints; and n is the number of the constraints.

D_i can be considered either as a continuous set or as a discrete one [5]. In the continuous problems, the design variables can vary continuously in the optimization process

$$D_i = \left\{x_i \middle| x_i \in [x_{i,\min}, x_{i,\max}]\right\} \tag{11.2}$$

where $x_{i,min}$ and $x_{i,max}$ are minimum and maximum allowable values for the design variable i, respectively. If the design variables represent a selection from a set of parts as

$$D_i = \left\{d_{i,1}, d_{i,2}, \ldots, d_{i,r(i)}\right\} \tag{11.3}$$

Then the problem is considered as a discrete one, where $r(i)$ is the number of available discrete values for the ith design variable.

In order to handle the constraints, a penalty approach is utilized. In this method, the aim of the optimization is redefined by introducing the cost function as

$$f_{\text{cost}}\left(\{x\}\right) = (1 + \varepsilon_1 \cdot \upsilon)^{\varepsilon_2} \times W(\{x\}), \qquad \upsilon = \sum_{i=1}^{n} \max[0, \upsilon_i] \tag{11.4}$$

where n represents the number of evaluated constraints for each individual design. The constant ε_1 and ε_2 are selected considering the exploration and the exploitation rate of the search space. Here, ε_1 is set to unity, ε_2 is selected in a way that it decreases the penalties and reduces the cross-sectional areas. Thus, in the first steps of the search process, ε_2 is set to 1.5 and ultimately increased to 3.

This chapter investigates two types of skeletal structures consisting of trusses and frames. The constraint conditions for these structures are briefly explained in the following sections.

11.2.1 Constraint Conditions for Truss Structures

For truss structures, the stress limitations of the members are imposed according to the provisions of ASD-AISC [6] as follows:

$$\begin{cases} \sigma_i^+ = 0.6F_y & for \ \ \sigma_i \geq 0 \\ \sigma_i^- & for \ \ \sigma_i < 0 \end{cases} \tag{11.5}$$

where σ_i^- is calculated according to the slenderness ratio:

$$\sigma_i^- = \begin{cases} \left[\left(1 - \dfrac{\lambda_i^2}{2C_C^2}\right)F_y\right] \Big/ \left(\dfrac{5}{3} + \dfrac{3\lambda_i}{8C_C} - \dfrac{\lambda_i^3}{8C_C^3}\right) & for \ \ \lambda_i < C_C \\ \dfrac{12\pi^2 E}{23\lambda_i^2} & for \ \ \lambda_i \geq C_C \end{cases} \tag{11.6}$$

where E is the modulus of elasticity; F_y is the yield stress of steel; C_c denotes the slenderness ratio (λ_i) dividing the elastic and inelastic buckling regions; λ_i presents the slenderness ratio.

The other constraint is the limitation of the nodal displacements:

$$\delta_{\mathrm{i}} \leq \delta_i^u \qquad i = 1, 2, ..., nn \tag{11.7}$$

where δ_i is the nodal deflection; δ_i^u is the allowable deflection of node i; and nn is the number of nodes.

11.2.2 Constraint Conditions for Steel Frames

Optimal design of frame structures is subjected to the following constrains according to LRFD-AISC provisions [7]:

Maximum lateral displacement

$$\frac{\Delta_T}{H} \leq R \tag{11.8}$$

Inter-story displacements constraints

$$\frac{d_i}{h_i} \le R_I, \quad i = 1, 2, \ldots, ns \tag{11.9}$$

The strength constraints

$$\begin{aligned} &\frac{P_u}{2\phi_c P_n} + \left(\frac{M_{ux}}{\phi_b M_{nx}} + \frac{M_{uy}}{\phi_b M_{ny}}\right) \le 1, \quad \text{For} \quad \frac{P_u}{\phi_c P_n} < 0.2 \\ &\frac{P_u}{\phi_c P_n} + \frac{8}{9}\left(\frac{M_{ux}}{\phi_b M_{nx}} + \frac{M_{uy}}{\phi_b M_{ny}}\right) \le 1, \quad \text{For} \quad \frac{P_u}{\phi_c P_n} \ge 0.2 \end{aligned} \tag{11.10}$$

where Δ_T is the maximum lateral displacement; H is the height of the frame structure; R is the maximum drift index (1/300); d_i is the inter-story drift; h_i is the story height of the ith floor, ns is the total number of stories; R_I presents the inter-story drift index permitted by the code of the practice (1/300); P_u is the required strength (tension or compression); P_n is the nominal axial strength (tension or compression); ϕ_c is the resistance factor ($\phi_c = 0.9$ for tension, $\phi_c = 0.85$ for compression); M_{ux} and M_{uy} are the required flexural strengths in the x and y directions, respectively; M_{nx} and M_{ny} are the nominal flexural strengths in the x and y directions (for two-dimensional structures, $M_{ny} = 0$); and ϕ_b denotes the flexural resistance reduction factor ($\phi_b = 0.90$). The nominal tensile strength for yielding in the gross section is computed as

$$P_n = A_g \cdot F_y \tag{11.11}$$

and the nominal compressive strength of a member is computed as

$$P_n = A_g \cdot F_{cr} \tag{11.12}$$

$$F_{cr} = \left(0.658^{\lambda_c^2}\right) F_y, \quad \text{For} \quad \lambda_c \le 1.5 \tag{11.13}$$

$$F_{cr} = \left(\frac{0.877}{\lambda_c^2}\right) F_y, \quad \text{For} \quad \lambda_c > 1.5$$

$$\lambda_c = \frac{kl}{r\pi}\sqrt{\frac{F_y}{E}} \tag{11.14}$$

where A_g is the cross-sectional area of a member.

11.3 Imperialist Competitive Algorithm

ICA simulates the social-political process of imperialism and imperialistic competition. This algorithm contains a population of agents or countries. The pseudo-code of the algorithm is as follows:

Step 1: Initialization The primary locations of the agents or countries are determined by the set of values assigned to each decision variable randomly as

$$x_{i,j}^{(o)} = x_{i,\min} + rand \cdot (x_{i,\max} - x_{i,\min}) \tag{11.15}$$

where $x_{i,j}^{(o)}$ determines the initial value of the ith variable for the jth country; $x_{i,\min}$ and $x_{i,\max}$ are the minimum and the maximum allowable values for the ith variable; *rand* is a random number in the interval [0,1]. If the allowable search space is a discrete one, using a rounding function will also be necessary.

For each country, the cost identifies its usefulness. In the optimization process, the cost is proportional to the penalty function. When the values of cost for initial countries are calculated [as defined by (11.4)], some of the best countries (in optimization terminology, countries with the least costs) will be selected to be the imperialist states and the remaining countries will form the colonies of these imperialists. The total number of initial countries is set to $N_{country}$ and the number of the most powerful countries to form the empires is equal to N_{imp}. The remaining N_{col} of the initial countries will be the colonies each of which belongs to an empire. In this chapter, a population of 30 countries consisting of 3 empires and 27 colonies are used. All the colonies of initial countries are divided among the imperialists based on their power. The power of each country, the counterpart of fitness value, is inversely proportional to its cost value. That is, the number of colonies of an empire should be directly proportionate to its power. In order to proportionally divide the colonies among the imperialists, a normalized cost for an imperialist is defined as

$$C_j = f_{\text{cost}}^{(imp,j)} - \max_{i} \left(f_{\text{cost}}^{(imp,i)} \right) \tag{11.16}$$

where $f_{\text{cost}}^{(imp,j)}$ is the cost of the jth imperialist and C_j is its normalized cost. The colonies are divided among empires based on their power or normalized cost and for the jth empire it will be as follows:

$$NC_j = Round\left(\left| \frac{C_j}{\sum_{i=1}^{N_{imp}} C_i} \right| \cdot N_{col} \right) \tag{11.17}$$

where NC_j is the initial number of colonies associated to the jth empire which are

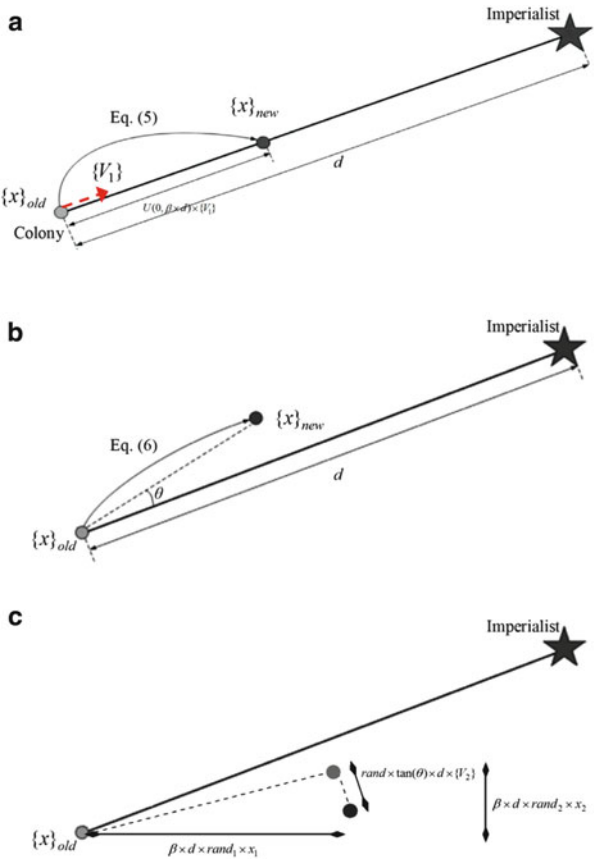

Fig. 11.1 Movement of colonies to its new location in the ICA [2] (**a**) toward their relevant imperialist, (**b**) in a deviated direction (**c**) using various random values

selected randomly among the colonies. These colonies together with the *j*th imperialist, form the empire number *j*.

Step 2: Colonies Movement In the ICA, the assimilation policy pursued by some of former imperialist states, is modeled by moving all the colonies toward the imperialist. This movement is shown in Fig. 11.1a in which a colony moves toward the imperialist by a random value that is uniformly distributed between 0 and $\beta \times d$ [3]:

$$\{x\}_{new} = \{x\}_{old} + U(0, \beta \times d) \times \{V_1\} \tag{11.18}$$

where β is a parameter with a value greater than one, and d is the distance between colony and imperialist. $\beta > 1$ peruseds the colonies to get closer to the imperialist state from both sides. $\beta \gg 1$ gradually results in a divergence of colonies from the imperialist state, while a very close value to 1 for β reduces the search ability of the algorithm. $\{V_1\}$ is a vector which its start point is the previous location of the

colony and its direction is toward the imperialist locations. The length of this vector is set to unity.

In order to increase the searching around the imperialist, a random amount of deviation is added to the direction of movement. Figure 11.1b shows the new direction which is obtained by deviating the previous location of the country as big as θ. In this figure θ is a random number with uniform distribution as

$$\theta = U(-\gamma, +\gamma) \tag{11.19}$$

where γ is a parameter that adjusts the deviation from the original direction. In most of the implementations, a value of about 2 for β [3] and about 0.1 (Rad) for γ, result in a good convergence of the countries to the global minimum.

In order to improve the performance of the ICA, we change the movement step as follow:

First: different random values are utilized for different components of the solution vector inplace of only one value (11.18) as

$$\{x\}_{new} = \{x\}_{old} + \beta \times d \times \{rand\} \otimes \{V_1\} \tag{11.20}$$

where $\{V_1\}$ is the base vector starting the previous location of colony and directing to the imperialistic; $\{rand\}$ is a random vector and the sign "$\otimes$" denotes an element-by-element multiplication. Since these random values are not necessarily the same, the colony is deviated automatically without using the definition of θ. However, for having a suitable exploration ability, the utilization of θ is modified by defining a new vector.

Second: From the above equation, it is possible to obtain the orthogonal colony-imperialistic contacting line (denoted by $\{V_2\}$). Then, deviation process is performed by using this vector in place of using θ as

$$\begin{aligned}\{x\}_{new} = \{x\}_{old} + \beta \times d \times \{rand\} \otimes \{V_1\} + U(-1, +1) \times \tan(\theta) \times d \\ \times \{V_2\}, \{V_1\} \cdot \{V_2\} = 0, \left|\left|\{V_2\}\right|\right| = 1\end{aligned} \tag{11.21}$$

Figure 11.1c describes the performance of this movement. In order to access the discrete results after performing the movement process, a rounding function is utilized which changes the magnitude of the results by the value of the nearest discrete value. Although this may reduce the exploration of the algorithm [8], as explained in the above, however we increase this ability by considering different random values and by defining a new deviation step.

Step 3: Imperialist Updating If the new position of the colony is better than that of its relevant imperialist (considering the cost function), the imperialist and the colony change their positions and the new location with a lower cost becomes the imperialist. Then the other colonies move toward this new position.

Step 4: Imperialistic Competition Imperialistic competition is another strategy utilized in the ICA methodology. All empires try to take the possession of colonies of other empires and control them. The imperialistic competition gradually reduces the power of weaker empires and increases the power of more powerful ones. The imperialistic competition is modeled by just picking some (usually one) of the weakest colonies of the weakest empires and making a competition among all empires to possess these (this) colonies. In this competition based on their total power, each of empires will have a likelihood of taking possession of the mentioned colonies.

Total power of an empire is mainly affected by the power of imperialist country. But the power of the colonies of an empire has an effect, though negligible, on the total power of that empire. This fact is modeled by defining the total cost as

$$TC_j = f_{\cos t}^{(imp,j)} + \xi \cdot \frac{\sum_{i=1}^{NC_j} f_{\cos t}^{(col,i)}}{NC_j} \tag{11.22}$$

where TC_n is the total cost of the jth empire and ξ is a positive number which is considered to be less than 1. A small value for ξ causes the total power of the empire to be determined by just the imperialist and increasing it will add to the role of the colonies in determining the total power of the corresponding empire. The value of 0.1 for ξ is found to be a suitable value in most of the implementations [3]. Similar to (11.16), the normalized total cost is defined as

$$NTC_j = TC_j - \max_{\mathrm{i}} \left(TC_i\right) \tag{11.23}$$

where NTC_j is the normalized total cost of the jth empire. Having the normalized total cost, the possession probability of each empire is evaluated by:

$$P_j = \left| \frac{NTC_j}{\sum_{i=1}^{N_{imp}} NTC_i} \right| \tag{11.24}$$

Step 5: Implementation When an empire loses all of its colonies, it is assumed to be collapsed. In this model implementation, where the powerless empires collapse in the imperialistic competition, the corresponding colonies will be divided among the other empires.

Step 6: Terminating Criterion Control Moving colonies toward imperialists are continued and imperialistic competition and implementations are performed during the search process. When the number of iterations reaches to a pre-defined value or

the amount of improvement in the best result reduces to a pre-defined value, the searching process is stopped.

The movement of colonies towards their relevant imperialist states along with competition among empires and also the collapse mechanism will hopefully cause all the countries to converge to a state in which there exist just one empire in the world and all the other countries are colonies of that empire. In this ideal new world, colonies will have the same position and power as the imperialist.

11.4 Design Examples

In this section, the optimal design of four steel structures is performed by the present algorithm. The final results are compared to the solutions of other methods to demonstrate the efficiency of the present approach. The examples contain a dome shaped truss example with continuous search space and a 72-bar spatial truss with the discrete variables. In addition, two benchmark frames are optimized by the ICA to find the optimum designs.

11.4.1 Design of a 120-Bar Dome Shaped Truss

The topology and elements group numbers of 120-bar dome truss are shown in Fig. 11.2. The modulus of elasticity is 30,450 ksi (210,000 MPa), and the material density is 0.288 lb/in^3 (7,971.810 kg/m^3). The yield stress of steel is taken as 58.0 ksi (400 MPa). The dome is considered to be subjected to vertical loading at all the unsupported joints. These loads are taken as −13.49 kips (−60 kN) at node 1, −6.744 kips (−30 kN) at nodes 2 through 14, and −2.248 kips (−10 kN) at the rest of the nodes. The minimum cross-sectional area of all members is 0.775 in^2 (2 cm^2) and the maximum cross-sectional area is taken as 20.0 in^2 (129.03 cm^2). The constraints are stress constraints [as defined by (11.5) and (11.6)] and displacement limitations of ±0.1969 in (±5 mm) imposed on all nodes in x, y and z directions.

Table 11.1 shows the best solution vectors, the corresponding weights and the required number of analyses for convergence of the present algorithm and some other metaheuristic algorithms. ICA-based algorithm needs 6,000 analyses to find the best solution while this number is equal to 150,000, 32,600, 10,000, 10,000 and 7,000 analyses for a PSO-based algorithm [11], a PSO and ACO hybrid algorithm [11], a combination algorithm based on PSO, ACO and HS [11], an improved BB–BC method using PSO properties [12] and the CSS algorithm [13], respectively. As a result, the ICA optimization algorithm has best convergence rates among the considered metaheuristics. Figure 11.3 shows the convergence history for the best results of the ICA. Comparing the final results of the ICA and those of the other metaheuristics, ICA finds the third best result while the difference between the

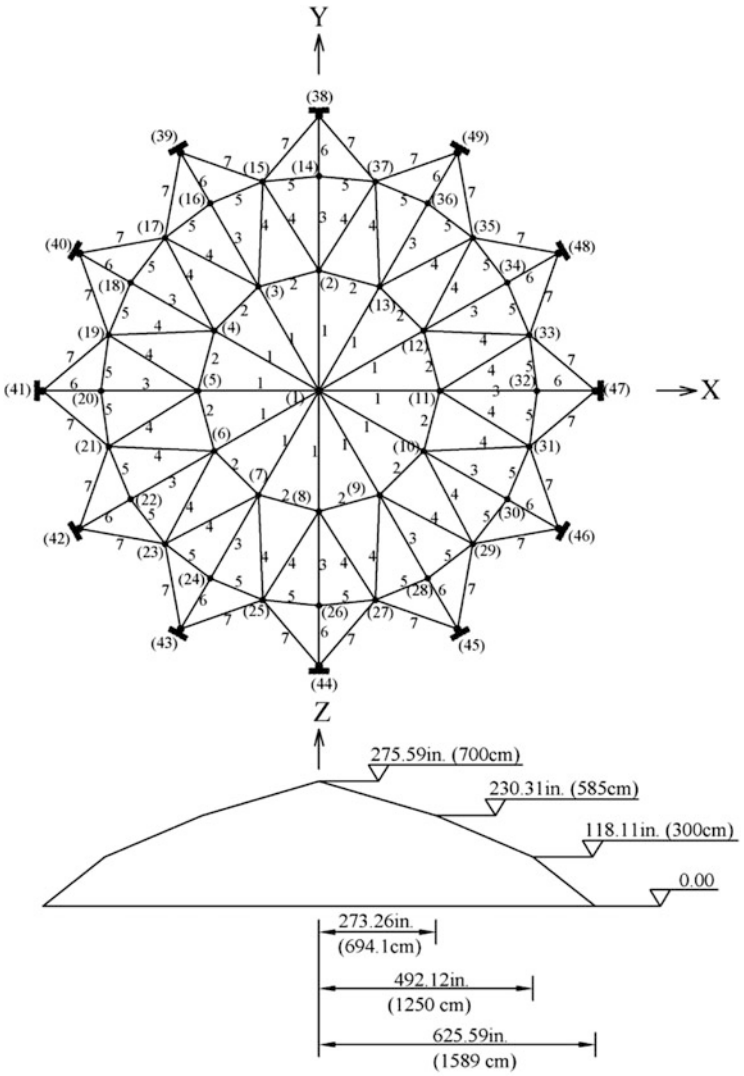

Fig. 11.2 Schematic of a 120-bar dome shaped truss

result of the ICA and those obtained by the HPSACO and the CSS methods, as the first and second best results, are very small. The maximum value for displacement is equal to 0.1969 in (5 mm) and the maximum stress ratio is equal to 99.999 %.

Table 11.1 Performance comparison for the 120-bar dome truss

Element group		Optimal cross-sectional areas (in^2)						
		PSOPC [10]	PSACO [10]	HPSACO [10]	HBB–BC [9]	CSS [6]	Present work [2]	
							in^2	cm^2
1	A_1	3.040	3.026	3.095	3.037	3.027	3.0275	19.532
2	A_2	13.149	15.222	14.405	14.431	14.606	14.4596	93.288
3	A_3	5.646	4.904	5.020	5.130	5.044	5.2446	33.836
4	A_4	3.143	3.123	3.352	3.134	3.139	3.1413	20.266
5	A_5	8.759	8.341	8.631	8.591	8.543	8.4541	54.543
6	A_6	3.758	3.418	3.432	3.377	3.367	3.3567	21.656
7	A_7	2.502	2.498	2.499	2.500	2.497	2.4947	16.095
Best weight (lb)		33,481.2	33,263.9	33,248.9	33,287.9	33,251.9	33,256.2	147,931 N
No. of required analyses		150,000	32,600	10,000	10,000	7,000	6,000	

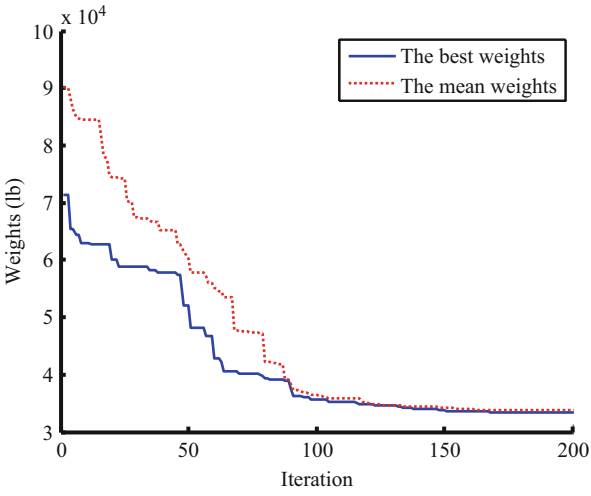

Fig. 11.3 The convergence for the dome shaped truss obtained by the ICA [2]

11.4.2 Design of a 72-Bar Spatial Truss

For the 72-bar spatial truss structure shown in Fig. 11.4, the material density is 0.1 lb/in^3 (2,767.990 kg/m^3) and the modulus of elasticity is 10,000 ksi (68,950 MPa). The members are subjected to the stress limits of ±25 ksi (±172.375 MPa). The nodes are subjected to the displacement limits of ±0.25 in (±0.635 cm). The 72 structural members of this spatial truss are categorized as 16 groups using symmetry: (1) A_1–A_4, (2) A_5–A_{12}, (3) A_{13}–A_{16}, (4) A_{17}–A_{18},

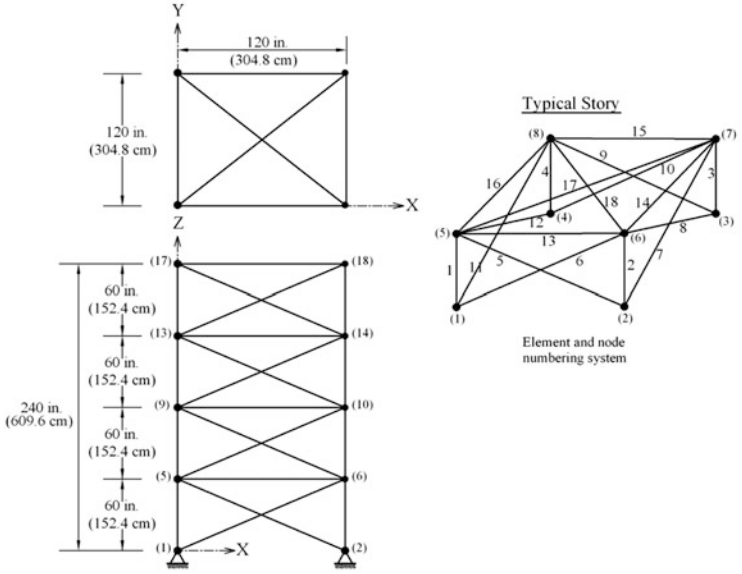

Fig. 11.4 Schematic of a 72-bar spatial truss

(5) A_{19}–A_{22}, (6) A_{23}–A_{30}, (7) A_{31}–A_{34}, (8) A_{35}–A_{36}, (9) A_{37}–A_{40}, (10) A_{41}–A_{48}, (11) A_{49}–A_{52}, (12) A_{53}–A_{54}, (13) A_{55}–A_{58}, (14) A_{59}–A_{66} (15), A_{67}–A_{70}, and (16) A_{71}–A_{72}. The discrete variables are selected from Table 11.2. The values and directions of the two load cases applied to the 72-bar spatial truss are listed in Table 11.3.

The ICA algorithm can find the best design among the other existing studies. The best weight of the ICA algorithm is 392.84 lb (178.19 kg), while it is 393.38 lb (178.43 kg), for the HPSACO [8]. The weight of the GA-based algorithm is equal to 427.203 lb (193.77 kg) [14]. The PSOPC and the standard PSO algorithms do not find optimal results when the maximum number of iterations is reached [10]. The HPSO and HPSACO algorithms get the optimal solution after 50,000 [10] and 5,330 [11] analyses while it takes only 4,500 analyses for the ICA. Table 11.4 compares the results of the CSS algorithm to those of the previously reported methods in the literature. In this example, stress constraints are not dominant while the maximum nodal displacement (0.2499 in or 0.635 cm) is close to its allowable value.

11.4.3 Design of a 3-Bay, 15-Story Frame

The configuration and applied loads of a three-bay fifty-story frame structure [5] is shown in Fig. 11.5. The displacement and AISC combined strength constraints are the performance constraint of this frame. The sway of the top story is limited to

Table 11.2 The available cross-section areas of the AISC code

No.	$in.^2$	mm^2	No.	$in.^2$	mm^2
1	0.111	(71.613)	33	3.840	(2,477.414)
2	0.141	(90.968)	34	3.870	(2,496.769)
3	0.196	(126.451)	35	3.880	(2,503.221)
4	0.250	(161.290)	36	4.180	(2,696.769)
5	0.307	(198.064)	37	4.220	(2,722.575)
6	0.391	(252.258)	38	4.490	(2,896.768)
7	0.442	(285.161)	39	4.590	(2,961.284)
8	0.563	(363.225)	40	4.800	(3,096.768)
9	0.602	(388.386)	41	4.970	(3,206.445)
10	0.766	(494.193)	42	5.120	(3,303.219)
11	0.785	(506.451)	43	5.740	(3,703.218)
12	0.994	(641.289)	44	7.220	(4,658.055)
13	1.000	(645.160)	45	7.970	(5,141.925)
14	1.228	(792.256)	46	8.530	(5,503.215)
15	1.266	(816.773)	47	9.300	(5,999.988)
16	1.457	(939.998)	48	10.850	(6,999.986)
17	1.563	(1,008.385)	49	11.500	(7,419.430)
18	1.620	(1,045.159)	50	13.500	(8,709.660)
19	1.800	(1,161.288)	51	13.900	(8,967.724)
20	1.990	(1,283.868)	52	14.200	(9,161.272)
21	2.130	(1,374.191)	53	15.500	(9,999.980)
22	2.380	(1,535.481)	54	16.000	(10,322.560)
23	2.620	(1,690.319)	55	16.900	(10,903.204)
24	2.630	(1,696.771)	56	18.800	(12,129.008)
25	2.880	(1,858.061)	57	19.900	(12,838.684)
26	2.930	(1,890.319)	58	22.000	(14,193.520)
27	3.090	(1,993.544)	59	22.900	(14,774.164)
28	1.130	(729.031)	60	24.500	(15,806.420)
29	3.380	(2,180.641)	61	26.500	(17,096.740)
30	3.470	(2,238.705)	62	28.000	(18,064.480)
31	3.550	(2,290.318)	63	30.000	(19,354.800)
32	3.630	(2,341.931)	64	33.500	(21,612.860)

Table 11.3 Loading conditions for the 72-bar spatial truss

	Case 1			Case 2		
Node	P_X *kips (kN)*	P_Y *kips (kN)*	P_Z *kips (kN)*	P_X	P_Y	P_Z *kips (kN)*
17	5.0 (22.25)	5.0 (22.25)	−5.0 (22.25)	0.0	0.0	−5.0 (22.25)
18	0.0	0.0	0.0	0.0	0.0	−5.0 (22.25)
19	0.0	0.0	0.0	0.0	0.0	−5.0 (22.25)
20	0.0	0.0	0.0	0.0	0.0	−5.0 (22.25)

23.5 cm (9.25 in.). The material has a modulus of elasticity equal to $E = 200$ GPa (29,000 ksi) and a yield stress of $F_y = 248.2$ MPa (36 ksi). The effective length factors of the members are calculated as $K_x \geq 0$ for a sway-permitted frame and the out-of-plane effective length factor is specified as $K_y = 1.0$. Each column is

Table 11.4 Optimal design comparison for the 72-bar spatial truss

Element group		Optimal cross-sectional areas (in^2)				
		GA [14]	PSOPC [10]	HPSO [10]	HPSACO [11]	Present work [2]
1	$A_1 \sim A_4$	0.196	4.490	4.970	1.800	1.99
2	$A_5 \sim A_{12}$	0.602	1.457	1.228	0.442	0.442
3	$A_{13} \sim A_{16}$	0.307	0.111	0.111	0.141	0.111
4	$A_{17} \sim A_{18}$	0.766	0.111	0.111	0.111	0.141
5	$A_{19} \sim A_{22}$	0.391	2.620	2.880	1.228	1.228
6	$A_{23} \sim A_{30}$	0.391	1.130	1.457	0.563	0.602
7	$A_{31} \sim A_{34}$	0.141	0.196	0.141	0.111	0.111
8	$A_{35} \sim A_{36}$	0.111	0.111	0.111	0.111	0.141
9	$A_{37} \sim A_{40}$	1.800	1.266	1.563	0.563	0.563
10	$A_{41} \sim A_{48}$	0.602	1.457	1.228	0.563	0.563
11	$A_{49} \sim A_{52}$	0.141	0.111	0.111	0.111	0.111
12	$A_{53} \sim A_{54}$	0.307	0.111	0.196	0.250	0.111
13	$A_{55} \sim A_{58}$	1.563	0.442	0.391	0.196	0.196
14	$A_{59} \sim A_{66}$	0.766	1.457	1.457	0.563	0.563
15	$A_{67} \sim A_{70}$	0.141	1.228	0.766	0.442	0.307
16	$A_{71} \sim A_{72}$	0.111	1.457	1.563	0.563	0.602
Weight (lb)		427.203	941.82	933.09	393.380	392.84
No. of required analyses		–	150,000	50,000	5,330	4,500

considered as non-braced along its length, and the non-braced length for each beam member is specified as one-fifth of the span length.

The optimum design of the frame is obtained after 6,000 analyses by using the ICA, having the minimum weight of 417.46 kN (93.85 kips). The optimum designs for HBB–BC [9], HPSACO, PSOPC and PSO [5] has the weights of 434.54 (97.65kN), 426.36 (95.85), 452.34 kN (101.69 kips) and 496.68 kN (111.66 kips), respectively. Table 11.5 summarizes the optimal designs for these algorithms. The HBB–BC approach could find the result after 9,900 analyses [9] and the HSPACO needs 6,800 analyses to reach a solution [5].

Figure 11.6 shows the convergence history for the result of the ICA method. The global sway at the top story is 11.52 cm, which is less than the maximum sway. The maximum value for the stress ratio is equal to 98.45 %. Also, the maximum drift story is equal to 1.04 cm.

11.4.4 Design of a 3-Bay 24-Story Frame

Figure 11.7 shows the topology and the service loading conditions of a three-bay twenty four-story frame consisting of 168 members originally designed by Davison and Adams [15]. Camp et al. utilized ant colony optimization [16], Degertekin developed least-weight frame designs for this structure using a harmony search [17]

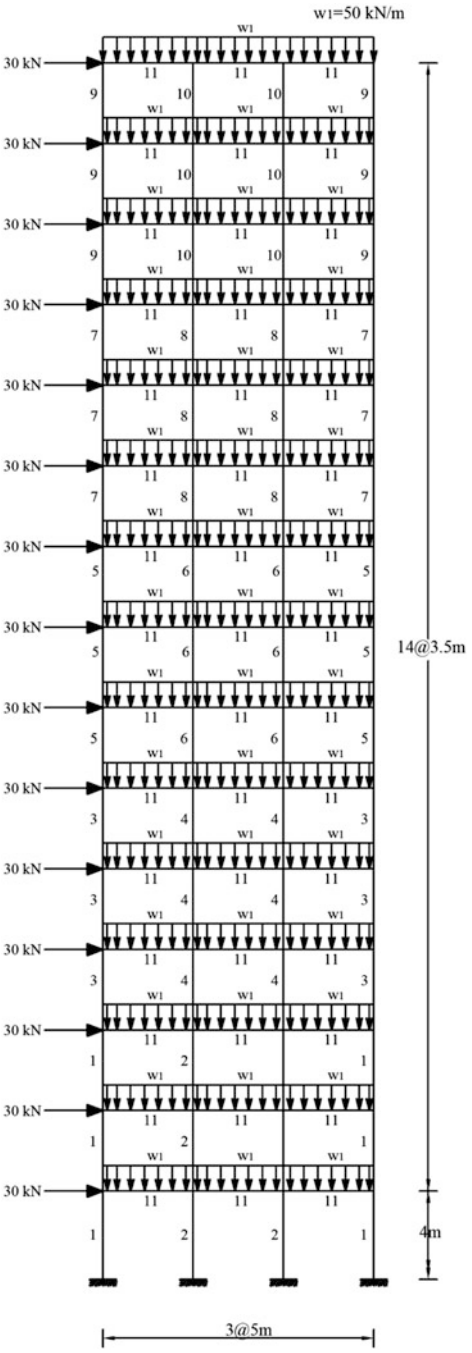

Fig. 11.5 Schematic of a three-bay fifteen-story frame

Table 11.5 Optimal design comparison for the 3-bay 15-story frame

	Optimal W-shaped sections				
Element group	PSO [5]	PSOPC [5]	HPSACO [5]	HBB–BC [9]	Present work [2]
1	W33X118	W26X129	W21X111	W24X117	W24X117
2	W33X263	W24X131	W18X158	W21X132	W21X147
3	W24X76	W24X103	W10X88	W12X95	W27X84
4	W36X256	W33X141	W30X116	W18X119	W27X114
5	W21X73	W24X104	W21X83	W21X93	W14X74
6	W18X86	W10X88	W24X103	W18X97	W18X86
7	W18X65	W14X74	W21X55	W18X76	W12X96
8	W21X68	W26X94	W26X114	W18X65	W24X68
9	W18X60	W21X57	W10X33	W18X60	W10X39
10	W18X65	W18X71	W18X46	W10X39	W12X40
11	W21X44	W21X44	W21X44	W21X48	W21X44
Weight (kN)	496.68	452.34	426.36	434.54	417.466
No. of required analyses	50,000	50,000	6,800	9,900	6,000

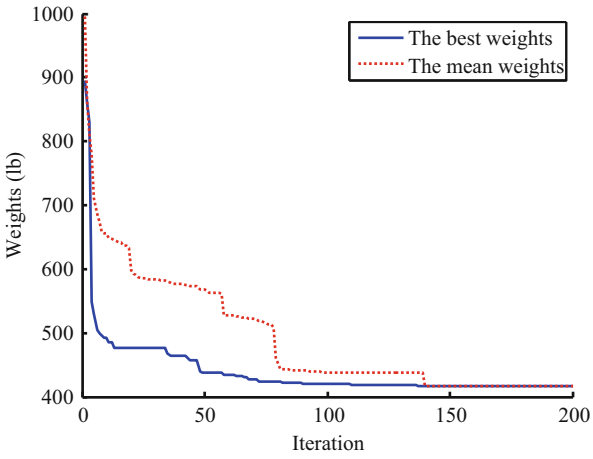

Fig. 11.6 The convergence for the three-bay fifteen-story frame obtained by the ICA [2]

and the authors utilized a hybrid PSO and BB–BC algorithm to solve this example [9].

The frame is designed following the LRFD specification and uses an inter-story drift displacement constraint. The material properties are: the modulus of elasticity $E = 205$ GPa (29,732 ksi) and a yield stress of $F_y = 230.3$ MPa (33.4 ksi). The detailed information is available in [9].

Table 11.6 lists the designs developed by: the ICA, the HBB–BC algorithm [9], the ant colony algorithm [16] and harmony search [17]. The ICA algorithm required 7,500 frame analyses to converge to a solution, while the 10,500 analyses were required by HBB–BC [9], 15,500 analyses by ACO [16] and 13,924 analyses by HS [17]. In this example, ICA can find the best results with 946.25 kN which is 3.67 %, 1.01 % and 1.60 % lighter than the results of the ACO [16], HS [17], and HBB–BC

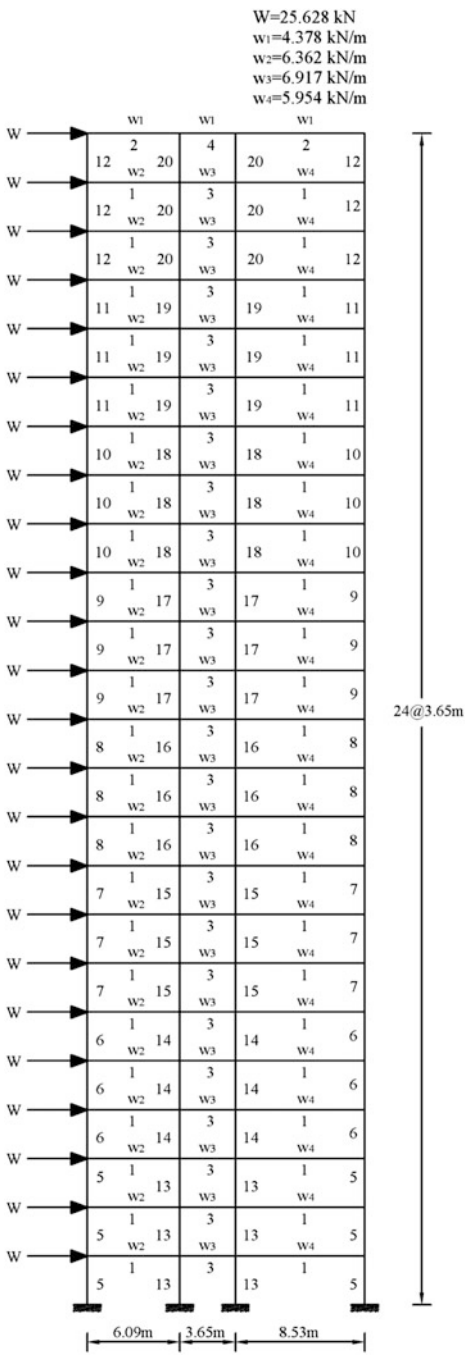

Fig. 11.7 Schematic of a three-bay twenty four-story frame

Table 11.6 Optimal design comparison for the 3-bay 24-story frame

	Optimal W-shaped sections			
	Camp et al. [16]	Degertekin [17]		
Element group	ACO	HS	HBB–BC [9]	Present work [2]
1	W30X90	W30X90	W30X90	W30X90
2	W8X18	W10X22	W21X48	W21X50
3	W24X55	W18X40	W18X46	W24X55
4	W8X21	W12X16	W8X21	W8X28
5	W14X145	W14X176	W14X176	W14X109
6	W14X132	W14X176	W14X159	W14X159
7	W14X132	W14X132	W14X109	W14X120
8	W14X132	W14X109	W14X90	W14X90
9	W14X68	W14X82	W14X82	W14X74
10	W14X53	W14X74	W14X74	W14X68
11	W14X43	W14X34	W14X38	W14X30
12	W14X43	W14X22	W14X30	W14X38
13	W14X145	W14X145	W14X159	W14X159
14	W14X145	W14X132	W14X132	W14X132
15	W14X120	W14X109	W14X109	W14X99
16	W14X90	W14X82	W14X82	W14X82
17	W14X90	W14X61	W14X68	W14X68
18	W14X61	W14X48	W14X48	W14X48
19	W14X30	W14X30	W14X34	W14X34
20	W14X26	W14X22	W14X26	W14X22
Weight (kN)	980.63	956.13	960.90	946.25
No. of required analyses	15,500	13,924	10,500	7,500

[9], respectively. The global sway at the top story is 25.52 cm (10.05 in.) which is less than the maximum sway. The maximum value for the stress ratio is 99.37 % and the maximum inter-story drift is equal to 1.215 cm (0.4784 in.). Figure 11.8 shows the values of the stress ratios for all elements of the optimum design obtained by the ICA algorithm.

11.5 Discussions

Many of metaheuristic algorithms are proposed based on the simulation of the natural processes. The genetic algorithms, particle swarm optimization, ant colony optimization, harmony search and charged system search are the most well-known metaheuristic algorithms. As an alternative to these metaheuristic approaches, this chapter investigates the performance of a new metaheuristic algorithm to optimize the design of skeletal structures. This method is called ICA which is a socio-politically motivated optimization algorithm.

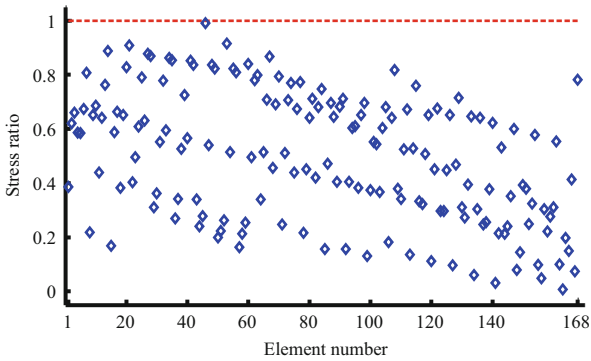

Fig. 11.8 The values of the stress ratios of elements for the ICA result [2]

In the ICA, an agent or a country can be treated as a colony or imperialist and the agents collectively form a number of empires. This algorithm starts with some random initial countries. Some of the best countries are selected to be the imperialist states and all the other countries form the colonies of these imperialists. Imperialistic competitions among the empires direct the search process towards the powerful imperialist and thus to the optimum spaces. During the competition, when weak empires collapse, the powerful ones take possession of their colonies. In addition, colonies of an empire move toward their related imperialist. In order to improve the ICA performance, here two movement steps are defined by using: (1) different random values for the components of the solution vector instead of only one value; (2) deviation by using orthogonal colony-imperialistic contacting line instead of using θ.

Four design examples consisting of two trusses and two frames are considered to illustrate the efficiency of the present algorithm. The comparisons of the numerical results of these structures utilizing the ICA and those obtained by other advanced optimization methods are performed to demonstrate the robustness of the present algorithm in finding good results in a less number of iterations. In order to highlight the positive characters of the ICA, a comparison of the ICA and the PSO algorithm is provided in the following:

- In the ICA algorithm, there is no need to save the pervious location of agents (velocity), while the PSO requires two positions saving memory (the current position and the pervious position).
- In the ICA algorithm, $\{V_1\}$ determines the movement direction of agents, while in the PSO, this is performed by the global and local best vectors. The vector $\{V_1\}$ is the best of the empire, i.e., it is the best agent among a predefined number of agents, while in the PSO the global best, denoted by $\{P_g\}$, is the position of the best agent of all agents. Therefore, $\{V_1\}$ will change for different agents during an iteration (depending on the empire which they belong to) and this helps the algorithm to increase the exploration ability, while $\{P_g\}$ is constant for all the agents in an iteration.

– In the ICA algorithm, saving the local best position of agents is not necessary, and instead the vector $\{V_2\}$ is utilized.

References

1. Kaveh A, Talatahari S (2010) Imperialist competitive algorithm for engineering design problems. Asian J Civil Eng 11(6):675–697
2. Kaveh A, Talatahari S (2010) Optimum design of skeletal structures using imperialist competitive algorithm. Comput Struct 88:1220–1229
3. Atashpaz-Gargari E, Lucas C (2007) Imperialist competitive algorithm: an algorithm for optimization inspired by imperialistic competition. In: IEEE congress on evolutionary computation, Singapore, pp 4661–4667
4. Atashpaz-Gargari E, Hashemzadeh F, Rajabioun R, Lucas C (2008) Colonial competitive algorithm: a novel approach for PID controller design in MIMO distillation column process. Int J Intell Comput Cybern 1(3):337–355
5. Kaveh A, Talatahari S (2009) Hybrid algorithm of harmony search, particle swarm and ant colony for structural design optimization. In: Geem ZW (ed) Harmony search algorithms for structural design. Springer, Berlin, Chapter 5
6. American Institute of Steel Construction (AISC) (1989) Manual of steel construction–allowable stress design, 9th edn. AISC, Chicago
7. American Institute of Steel Construction (AISC) (2001) Manual of steel construction–load resistance factor design, 3rd edn. AISC, Chicago
8. Kaveh A, Talatahari S (2009) A particle swarm ant colony optimization for truss structures with discrete variable. J Constr Steel Res 65(8–9):1558–1568
9. Kaveh A, Talatahari S (2010) A discrete big bang–big crunch algorithm for optimal design of skeletal structures. Asian J Civil Eng 11(1):103–122
10. Li LJ, Huang ZB, Liu F (2009) A heuristic particle swarm optimization method for truss structures with discrete variables. Comput Struct 87(7–8):435–443
11. Kaveh A, Talatahari S (2009) Particle swarm optimizer, ant colony strategy and harmony search scheme hybridized for optimization of truss structures. Comput Struct 87(5–6):267–283
12. Kaveh A, Talatahari S (2009) Size optimization of space trusses using Big Bang–Big Crunch algorithm. Comput Struct 87(17–18):1129–1140
13. Kaveh A, Talatahari S (2010) A novel heuristic optimization method: charged system search. Acta Mech 213(3–4):267–286
14. Wu SJ, Chow PT (1995) Steady-state genetic algorithms for discrete optimization of trusses. Comput Struct 56(6):979–991
15. Davison JH, Adams PF (1974) Stability of braced and unbraced frames. J Struct Div ASCE 100(2):319–334
16. Camp CV, Bichon J (2005) Design of steel frames using ant colony optimization. J Struct Eng ASCE 131:369–379
17. Degertekin SO (2008) Optimum design of steel frames using harmony search algorithm. Struct Multidiscip Optim 36:393–401

Chapter 12
Chaos Embedded Metaheuristic Algorithms

12.1 Introduction

In nature complex biological phenomena such as the collective behavior of birds, foraging activity of bees or cooperative behavior of ants may result from relatively simple rules which however present nonlinear behavior being sensitive to initial conditions. Such systems are generally known as "deterministic nonlinear systems" and the corresponding theory as "chaos theory". Thus real world systems that may seem to be stochastic or random, may present a nonlinear deterministic and chaotic behavior. Although chaos and random signals share the property of long term unpredictable irregular behavior and many of random generators in programming softwares as well as the chaotic maps are deterministic; however chaos can help order to arise from disorder. Similarly, many metaheuristics optimization algorithms are inspired from biological systems where order arises from disorder. In these cases disorder often indicates both non-organized patterns and irregular behavior, whereas order is the result of self-organization and evolution and often arises from a disorder condition or from the presence of dissymmetries. Self-organization and evolution are two key factors of many metaheuristic optimization techniques. Due to these common properties between chaos and optimization algorithms, simultaneous use of these concepts can improve the performance of the optimization algorithms [1]. Seemingly the benefits of such combination is a generic for other stochastic optimization and experimental studies confirmed this; although, this has not mathematically been proven yet [2].

Recently, chaos and metaheuristics have been combined in different studies for different purposes. Some of the works have intended to show the chaotic behaviors in the metaheuristic algorithms. In some of the works, chaos has been used to overcome the limitations of metaheuristics. Hence previous research can be classified into two types.

In the first type, chaos is inserted into the metaheuristics instead of a random number generator, *i.e.*, the chaotic signals are used to control the value of parameters in the metaheuristic's equations. The convergence properties of metaheuristics

A. Kaveh, *Advances in Metaheuristic Algorithms for Optimal Design of Structures*, 369
DOI 10.1007/978-3-319-05549-7_12, © Springer International Publishing Switzerland 2014

are closely connected to the random sequence applied on their operators during a run. In particular, when starting some optimizations with different random numbers, experience shows that the results may be very close but not equal, and require different numbers of generations to reach the same optimal value. The random numbers generation algorithms, on which most used metaheuristics tools rely, usually satisfy on their own some statistical tests like chi-square or normality. However, there are no analytical results that guarantee an improvement of the performance indexes of metaheuristics algorithms depending on the choice of a particular random number generator [3].

In the second type, chaotic search is incorporated into the procedures of the metaheuristics in order to enrich the searching behavior and to avoid being trapped in local optimums. A traditional chaos optimization algorithm (COA) which is a stochastic search technique was proposed based on the advantages of chaos variables. The simple philosophy of the COA contains two main stages: firstly mapping from the chaotic space to the solution space, and then searching optimal regions using chaotic dynamics instead of random search [4]. However, COA also has some disadvantages. For example, in the large-scale optimization problems the efficiency of the algorithm will be very low and the COA often needs a large number of iterations to reach the global optimum.

The main contribution of this chapter is to provide a state of the art review of the combination of chaos theory and metaheuristics, and describes the evolution of these algorithms along with some improvements, their combinations with various methods as well as their applications. Also a novel metaheuristic which is called chaotic swarming of particles (CSP) is introduced. The CSP uses chaos theory in order to improve the convergence characteristics of the particle swarm optimization (PSO) and to perform exploitation. This method is a kind of multi-phase optimization technique which employs chaos theory in two phases by the use of chaotic number generators each time a random number is needed by the classical PSO for parameters adaptation, and chaotic local search algorithm to avoid being trapped into local optimum.

12.2 An Overview of Chaotic Systems

In mathematic chaos is defined as “randomness” generated by simple deterministic systems. The randomness is a result of the sensitivity of chaotic systems to the initial conditions; it means that slight changes in the parameters or the starting values for the data lead to vastly different future behaviors, such as stable fixed points, periodic oscillations, bifurcations, and ergodicity. However, since the chaotic systems are deterministic, chaos implies order. A system can make the transformation from a regular periodic system to a complex chaotic system simply by changing one of the controlling parameters. Also a chaotic movement can go through every state in a certain area according to its own regularity, and every state is obtained only once [5]. An example of chaotic map is shown in Fig. 12.1.

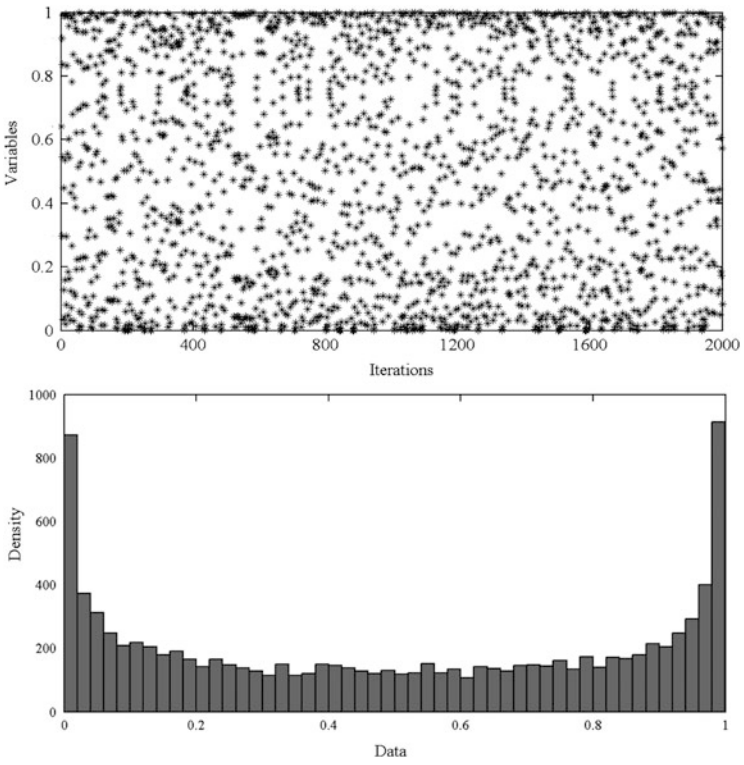

Fig. 12.1 An example of chaotic map (Logistic map)

Considering a discrete-time series, one can define chaos in the sense of Li-Yorke. A one-dimensional iterated map is based on a function of a real variable and takes the form

$$x_{t+1} = F(x_t) \tag{12.1}$$

where $x(t) \in \Re^n$, $t = 1, 2, 3, \ldots$ and F is a map from $\Re^n$ to itself.

Let $F^{(p)}$ denotes the composition of F with itself $p > 0$ times, then a point x is called a p-periodic point of F if $F^{(p)}(x) = x$ but $F^{(k)}(x) \neq x$ for all k such that $k \leq p$. In particular, a point x satisfying $F(x) = x$ is called a fixed point of F. The ε-neighborhood $N_\varepsilon(x)$ of a point x is defined by

$$N_\varepsilon(x) = \left\{ y \in \Re^n \middle| \|x - y\| \leq \varepsilon \right\} \tag{12.2}$$

where $\|.\|$ denotes the Euclidean norm in $\Re^n$. Then, we introduce the following definition of chaos in the sense of Li-Yorke [6]:

Definition 1 If a discrete-time series satisfies the following conditions, then it is called chaotic:

1. There exist a positive constant N such that for any $p \geq N$, F has a p-periodic point.
2. There exists an uncountable set $S \subset \Re^n$, which does not include any periodic point of F and satisfies the following conditions

   (a) $F(S) \subset S$

   (b) For any points $x, y \in S(x \neq y)$

$$\lim_{n\to\infty} \sup \left\| F^{(n)}(x) - F^{(n)}(y) \right\| > 0,$$

   and for any $x \in S$ and any periodic point y of F,

$$\lim_{n\to\infty} \sup \left\| F^{(n)}(x) - F^{(n)}(y) \right\| > 0.$$

   (c) There exists an uncountable subset $S_0 \subset S$ such that for any $x, y \in S_0$,

$$\lim_{n\to\infty} \inf \left\| F^{(n)}(x) - F^{(n)}(y) \right\| = 0$$

The set S in the above definition is called the scrambled set.

Then, it is well known that the existence of a fixed point called a snap-back repeller in a system implies that the system is chaotic in the sense of Li-Yorke [7]. Thus a system is chaotic if it contains infinitely many periodic orbits whose periods are arbitrarily large. This definition essentially is a result of Sarkovskii's theorem which was proved by the Russian mathematician Sarkovskii in 1964; however apparently presented in a well-known paper by Li and Yorke [6] in which the word chaos first appeared in its contemporary scientific meaning [8].

A chaotic map can be used as spread-spectrum sequence for random number sequence. Chaotic sequences have been proven to be easy and fast to generate and store, and therefore there is no need for storing long sequences. One needs only a few functions (chaotic maps) and few parameters (initial conditions) for very long sequences. Also an enormous number of different sequences can be generated simply by altering its initial condition. In addition, these sequences are deterministic and reproducible. The choice of chaotic sequences can be justified theoretically by their unpredictability, corresponding to their spread-spectrum characteristic and ergodic properties [9]. Therefore when a random number is needed, it can be generated by iterating one step of the chosen chaotic map (cm) being started from a random initial condition at the first iteration of the run. The literature is rich in chaotic time series sequences, some of these are listed in following subsections.

12.2.1 Logistic Map

This map, whose equation appears in nonlinear dynamics of biological population evidencing chaotic behavior (May [10])

$$x_{k+1} = ax_k(1 - x_k) \tag{12.3}$$

In this equation, x_k is the kth chaotic number, with k denoting the iteration number. Obviously, $x_k \in (0, 1)$ under the conditions that the initial $x_o \in (0, 1)$ and that $x_0 \notin \{0.0, 0.25, 0.5, 0.75, 1.0\}$. In the experiments $a = 4$ is utilized.

12.2.2 Tent Map

Tent map resembles the logistic map, Peitgen et al. [11]. It generates chaotic sequences in (0, 1) assuming the following form

$$x_{k+1} = \begin{cases} x_k/0.7 & x_k < 0.7 \\ 10/3x_k(1 - x_k) & otherwise \end{cases} \tag{12.4}$$

12.2.3 Sinusoidal Map

This iterator (May [10]) is represented by

$$x_{k+1} = ax_k^2 \sin(\pi x_k) \tag{12.5}$$

For $a = 2.3$ and $x_o = 0.7$ it has the following simplified form:

$$x_{k+1} = \sin(\pi x_k) \tag{12.6}$$

It generates chaotic sequence in (0, 1).

12.2.4 Gauss Map

The Gauss map is utilized for testing purpose in the literature (Peitgen et al. [11]) and is represented by

$$x_{k+1} = \begin{cases} 0 & x_k = 0 \\ 1/x_k \bmod(1) & otherwise \end{cases}$$

$$1/x_k \mathrm{mod}(1) = \frac{1}{x_k} - \left[\frac{1}{x_k}\right] \tag{12.7}$$

Here, $[x]$ denotes the largest integer less than x and acts as a shift on the continued fraction representation of numbers. This map also generates chaotic sequences in (0, 1).

12.2.5 Circle Map

The circle map (Zheng [12]) is represented by

$$x_{k+1} = x_k + b - (a/2\pi)\sin(2\pi x_k)\mathrm{mod}(1) \tag{12.8}$$

With $a = 0.5$ and $b = 0.2$, it generates chaotic sequence in (0, 1).

12.2.6 Sinus Map

Sinus map is defined as

$$x_{k+1} = 2.3(x_k)^{2\sin(\pi x_k)} \tag{12.9}$$

12.2.7 Henon Map

This map is a nonlinear two-dimensional map most frequently employed for testing purposes, and it is represented by

$$x_{k+1} = 1 - ax_k^2 + bx_{k-1} \tag{12.10}$$

The suggested parameter values are $a = 1.4$ and $b = 0.3$.

12.2.8 Ikeda Map

An Ikeda map is a discrete-time dynamical system defined by Dressler and Farmer [13]:

$$\begin{aligned} x_{n+1} &= 1 + 0.7(x_n \cos(\theta_n) - y_n \sin(\theta_n)), \\ y_{n+1} &= 0.7(x_n \sin(\theta_n) + y_n \cos(\theta_n)), \\ \theta_n &= 0.4 - \frac{6}{1 + x_n^2 + y_n^2} \end{aligned} \tag{12.11}$$

12.2.9 Zaslavskii Map

One of the interesting dynamic systems evidencing chaotic behavior is the Zaslavskii map (Zaslavskii [14]), the corresponding equation is given by

$$\begin{aligned} x_{k+1} &= x_k + v + \alpha y_{k+1} \pmod 1 \\ y_{k+1} &= \cos(2\pi x_k) + e^{-r} y_k \end{aligned} \tag{12.12}$$

where mod is the modulus after division and $v = 400$, $r = 3$, $\alpha = 12.6695$. In this case, $y_t \in [-1.0512, 1.0512]$.

12.3 Use of Chaotic Systems in Metaheuristics

In the artificial intelligence community, the term metaheuristic was created and is now well accepted for general algorithms that represent a family of approximate optimization methods which are not limited to a particular problem. There were many attempts to give a rigorous mathematical definition of metaheuristics. Here some of these are accompanied by explanations.

1. "They are solution methods that orchestrate an interaction between local improvement procedures and higher level strategies to create a process capable of escaping from local optima and performing a robust search of a solution space." (Glover and Kochenberger [15])
2. "These methods can be defined as upper level general methodologies that can be used as guiding strategies in designing underlying heuristics to solve specific optimization problems", Talbi [16]
3. "They are a set of concepts that can be used to define heuristic methods that can be applied to a wide set of different problems with relatively few modifications to make them adapted to a specific problem", Dorigo [17].

Design and implementation of such optimization methods had been at the origin of a multitude of contributions to the literature in the last 50 years as described in the previous chapters. Generally, a metaheuristic algorithm uses two basic strategies while searching for the global optima; exploration and exploitation. The exploration enables the algorithm to reach at the best local solutions within the search space, and the exploitation provides the ability to reach at the global optimum solution which may exist around the local solutions obtained. In exploitation, the promising regions are explored more comprehensively, while in

exploration the non-explored regions are visited to make sure that all the regions of the search space are fairly explored.

Due to common properties of chaos and metaheuristic optimization algorithms, simultaneous use of these concepts seems to improve the performance and to overcome the limitations of metaheuristics. The previous research can be categorized into two types. In the first type, chaotic system is inserted into the metaheuristics instead of a random number generator for updating the value of parameters; and in the second type, chaotic search is incorporated into the procedures of the metaheuristics in order to enrich the searching behavior and to avoid being trapped in local optimums using traditional COAs.

12.4 Chaotic Update of Internal Parameters for Metaheuristics

For simulating complex phenomena, sampling, numerical analysis, decision making and in particular in metaheuristic optimization, random sequences are needed with a long period and reasonable uniformity. On the other hand as mentioned before chaos is a deterministic, random-like process found in nonlinear dynamical system which is non-period, non-converging and bounded. The nature of chaos looks to be random and unpredictable, possessing an element of regularity. Mathematically, chaos is randomness of a simple deterministic dynamical system, and chaotic system may be considered as sources of randomness (Schuster [18]; Coelho and Mariani [19]; Alatas [20]).

However, metaheuristics are non-typical; hence, the critical issue in implementing metaheuristic methods is the determination of "proper" parameters which must be established before running these algorithms. The efficient determination of these parameters leads to a reasonable solution. That is why; these parameters may be selected chaotically by using chaotic maps. In this case, sequences generated from chaotic systems substitute random numbers for the parameters where it is necessary to make a random-based choice. By this way, it is intended to improve the global convergence and to prevent to stick on a local solution.

Alatas et al. [21] proposed different chaotic maps to update the parameters of PSO algorithm. This has been done by using of chaotic number generators each time a random number is needed by the classical PSO algorithm. Twelve chaos-embedded PSO methods have been proposed and eight chaotic maps have been analyzed in the unconstrained benchmark functions. The simulation results show that the application of deterministic chaotic signals may be a possible strategy to improve the performances of PSO algorithms. Also Alatas [20] presented another interesting application. He has integrated chaos search with HS for improved performance. Seven new chaotic HS algorithms have been developed using different chaotic maps. A similar utilizing of chaotic sequences for artificial bee colony

(ABC) (Alatas [22], BB-BC (Alatas [23]), ICA (Talatahari et al. [24]), and CSS (Talatahari et al. [25]) have been performed by researchers. Based on the results obtained from literature, it is not easy to say which chaotic map performs the best. However, we can say that chaotic maps have a considerable positive impact on the performance of metaheuristics.

In these studies generally unconstraint problems were considered. On the other hand, most of the real life problems including design optimization problems require several types of variables, objective and constraint functions simultaneously in their formulation. In engineering design as the first attempts to analyze the performance of metaheuristics in which chaotic maps are used for parameters updating process, Talatahari et al. [26] combined the benefits of chaotic maps and the ICA to determine optimum design of truss structures. These different chaotic maps were investigated by solving two benchmark truss examples involving 25- and 56-bar trusses to recognize the most suitable one for this algorithm. As an example, a 56-bar dome truss structure taken from [26] is shown in Fig. 12.2. Members of the dome are categorized into seven groups. Table 12.1 shows the statistical results and the optimum weight for the 56-bar dome truss using the ICA algorithms, where cm is a chaotic map based on the Sinusoidal map for CICA-1, Logistic map for CICA-2, Zaslavskii map for CICA-3 and Tent map for CICA-4. The results show that the use of Sinusoidal map (CICA-1) results in a better performance for the chaotic ICA than the others. Two other larger examples were also considered by Talatahari et al. [26] to obtain more clear details about the performance of the new algorithm. These were 200- and 244-bar trusses with 29 and 32 design variables, respectively. Almost for all examples, the performance of the new algorithm is far better than the standard ICA; especially when the standard deviations of the results are compared. The standard deviation of the new algorithm is much better than the standard ICA and this illustrates the high ability of the new algorithm.

As another attempt in optimization problems related to the engineering design, a new improved CSS using chaotic maps was presented for engineering optimization by Talatahari et al. [27]. They defined five different variants of the new methodology by adding the chaos to the enhanced CSS. Then, different chaotic systems were utilized instead of different parameters available in the algorithm. To evaluate the performance of the new algorithm two sets of examples were considered: In the first set four well-known benchmark examples including design of a piston lever, design of a welded beam, design of a four-storey, two-bay frame, and design of a car side impact were selected from literature to compare the variants of the new method. In the second set two mechanical examples consisting of a 4 step-cone pulley design and speed reducer design problems were utilized in order to compare the variants of the new method with other metaheuristics. As an example, in design of a 4 step-cone pulley the objective is to design a pulley with minimum weight using five design variables, as shown in Fig. 12.3. Four design variables are associated with the diameters of each step, and the fifth corresponds to the width of the pulley. In this example, it is assumed that the widths of the cone pulley and belt are identical. There are 11 constraints, out of which 3 are equality constraints and the remaining are inequality constraints. The constraints are imposed to assure the same belt

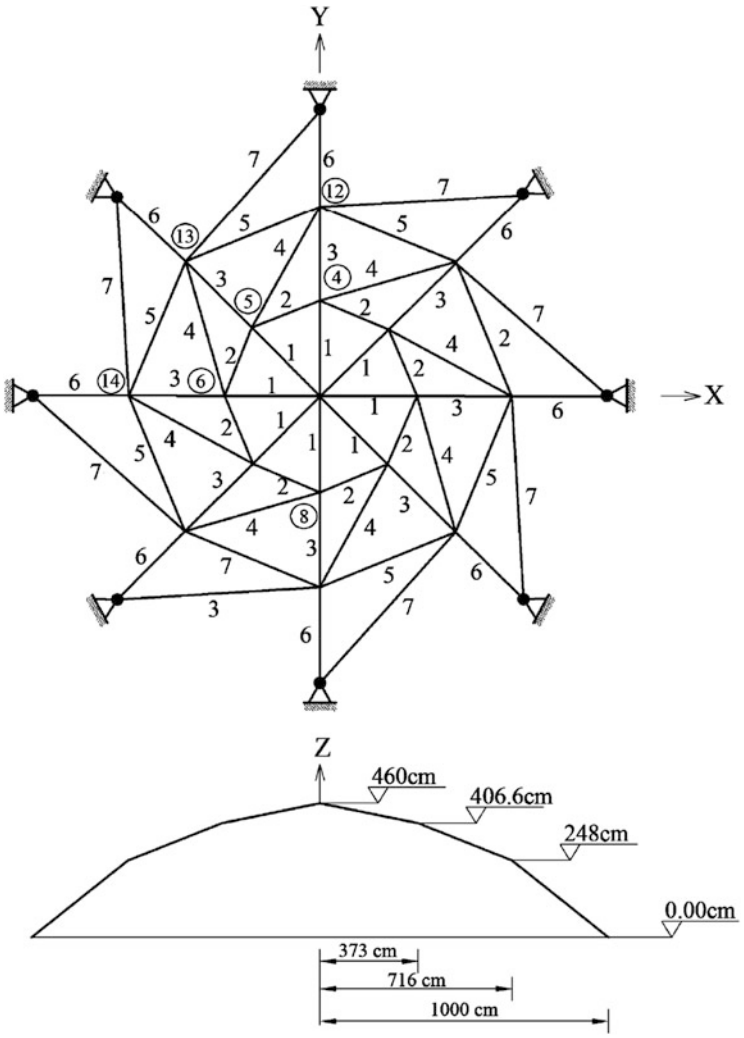

Fig. 12.2 Schematic of a 56-bar dome spatial truss structure [26]

Table 12.1 Optimal design comparison for the 56-bar dome truss

	ICA	CICA-1	CICA-2	CICA-3	CICA-4
Best weight (kg)	546.14	546.13	546.16	546.15	546.15
Average weight (kg)	547.91	546.21	546.31	546.24	546.34
Std dev (kg)	5.791	0.49	0.62	0.56	0.59

length for all the steps, tension ratios, and power transmitted by the belt. The 4 step pulley is designed to transmit at least 0.75 hp (0.75·745.6998 W), with an input speed of 350 rpm and output speeds of 750, 450, 250, and 150 rpm. This problem is

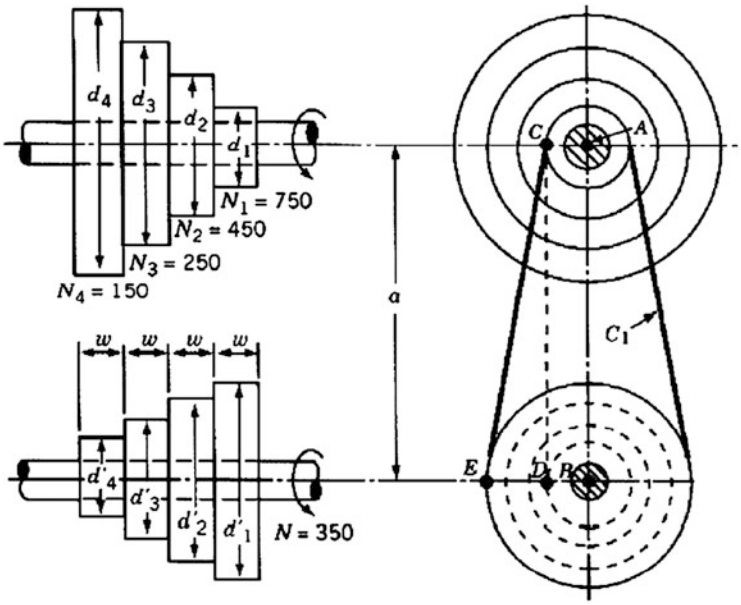

Fig. 12.3 A 4 step-cone pulley [28]

considered to compare the chaotic CSS (CCSS) method with other metaheuristic algorithms which was solved by using Teaching–learning-based optimization (TLBO) and ABC, previously, Rao et al. [28]. It is observed from Table 12.2 that CCSS gives better results than the other methods for the best, mean, and standard deviation, Talatahari et al. [27].

Due to the simplicity and potency of these methods, it seems that they can easily be utilized for many engineering optimization problems.

12.5 Chaotic Search Strategy in Metaheuristics

The basic idea of COA generally includes two major stages. Firstly, based on the selected chaotic map (cm) define a chaotic number generator for generating sequences of points then map them to a design space. Afterwards, evaluate the objective functions with respect to these points, and choose the point with the minimum objective function as the current optimum. Secondly, the current optimum is assumed to be close to the global optimum after certain iterations, and it is viewed as the center with a little chaotic perturbation, and the global optimum is obtained through fine search. Repeat the above two steps until some specified convergence criterion is satisfied, and then the global optimum is obtained, Yang et al. [29]. The pseudo-code of COA is summarized as follows

Table 12.2 Statistical results of the 4 step-cone pulley for different metaheuristics

Method	Best	Mean	Std dev
TLBO	16.63451	24.0113	0.34
ABC	16.63466	36.0995	0.06
CCSS	16.41235	29.1058	0.11

Step 1: *Initialization*. Initialize the number N of chaotic search, different initial value of n chaos variables cm_i^0, and the lower and upper bound of the decision variables (X_L and X_U). Set the iteration counter as $k = 1$. Determine the initial design variables as

$$x_i^0 = X_{L_i} + cm_i^0(X_{U_i} - X_{L_i}), \qquad i = 1, 2, \ldots, n \tag{12.13}$$

Evaluate the objective function and set $f* = f(x^0)$.
Step 2: *Variable mapping*. Map chaotic variables cm^k into the variance range of the optimization variables by the following equation

$$x_i^{k+1} = X_{L_i} + cm_i^{k+1}(X_{U_i} - X_{L_i}), \qquad i = 1, 2, \ldots, n \tag{12.14}$$

Step 3: *Searching for optimal solution*. Evaluate the objective function.

If $k \leq N$, then

If $f(x^{k+1}) \leq f*$, then $x^* = x^{k+1}, f* = f(x^{k+1})$.
Set $k = k + 1$, $cm^k = cm^{k+1}$, and go to step 2.

Else if $k > N$ is satisfied then stop.

Due to the pseudo-randomness of chaotic motion, the motion step of chaotic variables between two successive iterations is always big, which resulted in the big jump of the design variables in design space. Thus, even if the above COAs have reached the neighborhood of the optimum, it needs to spend much computational effort to approach the global optimum eventually by searching numerous points.

Hence, the hybrid methods attracted the attention of some researchers, in which chaos is incorporated into the metaheuristics where the parallel searching ability of metaheuristics and chaotic searching behavior are reasonably combined. Wu and Cheng [30] integrated GA with COA, which uses chaos sequence to generate original population and add chaotic fine search to genetic operation which can avoid premature convergence. Guo and Wang [31] presented a novel immune evolutionary algorithm (IEA) based on COA to improve the convergence performance of the IEA. Ji and Tang [32] and Liu et al. [4] suggested a hybrid method of SA and PSO combined with chaos search, and examined its efficiency with several nonlinear functions, respectively. Similar approaches were also presented for PSO by Wang and Liu [33], Gao and Liu [34], He et al. [35]. Baykasoglu [36] presented how can the performance of great deluge algorithm (GDA) be enhanced by integrating with COA for solving constrained non-linear engineering design

optimization problems. Such hybrid methods can save much CPU time and enhance the computational efficiency of algorithms.

12.6 A New Combination of Metaheuristics and Chaos Theory

CSP is a newly developed type of metaheuristic algorithms. This algorithm is proposed by Kaveh et al. [37]. The CSP is inspired from the chaotic and collective behavior of species such as bees, fishes, and birds in which chaos theory is used to control the value of the parameters of PSO and to increase the local search capability of the PSO in order to enhance search behavior and skip local optima. The CSP approach not only performs exploration by using the population-based evolutionary searching ability of PSO, but also performs exploitation by using the chaotic local searching behavior. The framework of the CSP is illustrated in Fig. 12.4. In the CLSPSO1 phase, the initial positions of the particles are determined chaotically in the search space. The values of the fitness function for the particles are also calculated. The best particle among the entire set of particles is treated as a global best (X_g). After reaching a pre-defined number of iterations (N_1), the CLSPSO1 is stopped and switched to PSO while CPVPSO applies for updating the value of parameters in the velocity updating equation. In the second phase, the CLSPSO2 (updating process) is activated if PSO stops moving. CLSPSO2 causes the particles to escape from local minima using the logistic map. After a better solution is found by the CLSPSO2 or after a fixed number of iterations (N_2), the PSO will continue. The algorithm is terminated when the termination criterion has been met: that is, if there is no significant improvement in the solution. The CSP algorithm can simply be described as follows:

12.6.1 The Standard PSO

PSO involves a number of particles, which are initialized randomly in the space of the design variables. These particles fly through the search space and their positions are updated based on the best positions of individual particles and the best position among all particles in the search space which in truss sizing problems corresponds to a particle with the smallest weight in PSO, a swarm consists of N particles moving around in a D-dimensional search space. The position of the jth particle at the kth iteration is used to evaluate the quality of the particle and represents candidate solution(s) for the search or optimization problems. The update moves a particle by adding a change velocity V_j^{k+1} to the current position $X_j{}^k$ as follows

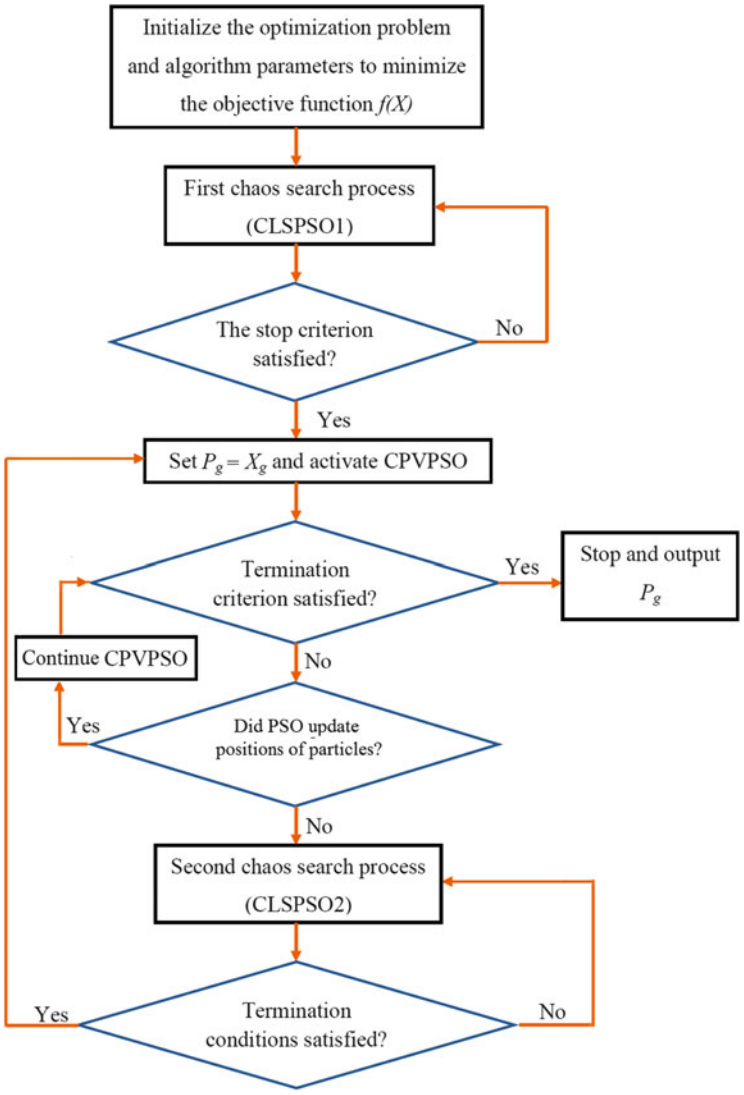

Fig. 12.4 Flowchart of the CSP algorithm [1]

$$\begin{aligned} V_j^{k+1} &= wV_j^k + c_1 \times r_{1j}^k \otimes \left(P_j^k - X_j^k\right) + c_2 \times r_{2j}^k \otimes \left(P_g^k - X_j^k\right) \\ X_j^{k+1} &= X_j^k + V_j^k \end{aligned} \tag{12.15}$$

where w is an inertia weight to control the influence of the previous velocity; $r^k{}_{1j}$ and $r^k{}_{2j}$ are random numbers uniformly distributed in the range of (0,1); c_1 and c_2 are two acceleration constants namely called cognitive and social parameter,

respectively; P_j^k is the best position of the jth particle up to iteration k; $P_g{}^k$ is the best position among all particles in the swarm up to iteration k. In order to increase PSO's exploration ability, the inertia weight is now modified during the optimization process with the following equation

$$w^{k+1} = w^k \times D_r \times rand \tag{12.16}$$

where D_r is the damping ratio which is a constant number in the interval (0,1); and *rand* is a uniformly distributed random number in the range of (0,1).

12.6.2 The CPVPSO Phase [37]

In this phase, when a random number is needed by PSO algorithm, it can be generated by iterating one step of the chosen chaotic map (cm) being started from a random initial condition of the first iteration of PSO. As we mentioned before one of the well-known chaotic maps is the Logistic map which is a polynomial map. This map is defined by (12.3).

In order to control values of PSO parameters by using chaotic maps, $r^k{}_{1j}$, $r^k{}_{2j}$, and *rand* are generated from the iterations of Logistic map instead of using classical random number generator as

$$\begin{aligned} V_j^{k+1} &= w^k \times V_j^k + c_1 \times cm^k \otimes \left(P_j^k - X_j^k\right) + c_2 \times cm^k \otimes \left(P_g^k - X_j^k\right) \\ w^{k+1} &= w^k \times D_r \times cm^k \end{aligned} \tag{12.17}$$

12.6.3 The CLSPSO Phase [37]

In this phase, COA is introduced in the PSO formulation. This is a kind of multi-phase optimization technique because chaotic optimization and PSO coexist and are switched to each other according to certain conditions. Here, chaotic search that uses Logistic map for the particle is incorporated to enhance search behavior and to skip local optima. The CLSPSO process is now described:

- *CLSPSO1* (*First chaotic search process*):

  Step 1: Set $t = 0$. Initialize the number of the first chaotic search N_1, initial value of chaos variables (cm^0), the lower and upper bound of the decision variables (X_{min} and X_{max}), and the number of particles. Determine the initial design variables for the jth particle as

$$X_j^0 = X_{\min} + cm_j^0 (X_{\max} - X_{\min}) \tag{12.18}$$

  Step 2: Evaluate the objective function and determine $X_g{}^0$ by finding $f^* = \min f(X_j{}^0)$.

Step 3: Map the chaotic variables cm^t into the variance range of the optimization variables by the following equation:

$$X_j^{t+1} = X_g^t + \left(2cm_j^t - 1\right)\left(X_g^t - X_j^t\right) \tag{12.19}$$

Step 4: Evaluate the new position ($X_j{}^{t+1}$).
Step 5: If the new solution is better than the initial solution $f(X_j{}^{t+1}) \leq f^*$, then $f^* = f(X_j{}^{t+1})$.
Step 6: Generate the next values of the chaotic variables by a chaotic map and set $t = t + 1$.
Step 7: If $t < N_1$ go to step 3, else stop the first chaotic search process and obtain the output X_g and f^* as the result of the CLSPSO1.
Step 8: Set X_g as the global best (P_g).

- *CLSPSO2 (Second chaotic search process):*

Step 1: Initialize the number of the second chaotic search N_2 and set i =1.
Step 2: Using the PSO algorithm, generate the global best $P^k{}_g$.
Step 3: Set $X^i{}_g = P^k{}_g$
Step 4: Update the global best position of the particles using the chaotic map by the following equation:

$$X_g^{i+1} = X_g^i + \left(2cm^i - 1\right)\frac{X_{\max} - X_{\min}}{k} \tag{12.20}$$

Step 5: If the new solution is better than the initial solution $f(X_g{}^{i+1}) \leq f(X_g{}^i)$, then $f^* = f(X_g{}^{i+1})$ *and* $P^k{}_g = X_g{}^{i+1}$.
Step 6: Generate the subsequent values of the chaotic variables by a chaotic map and set $i = i + 1$.
Step 7: If $i < N_2$ go to step 4, else stop the second chaotic search process and obtain the output $P^k{}_g$ and f^* as the result of the CLSPSO2.

12.6.4 Design Examples

In order to test the performance of the CSP method, two large-scale test problems are adapted from Kaveh et al. [37] which previously treated by other investigators are studied: the weight minimization 200-bar, and 942-bar truss. For all test cases, after a sensitivity analysis, the CSP internal parameters are set to: $w^0 = 0.9$, damping ratio (D_r) = 0.99, number of the first chaotic search (N_1) = 50 and number of the second chaotic search (N_2) = 10. Also the maximum number of iteration is set to 2,500, number of particles (N) = 100, and $c_1 = 1$, $c_2 = 3$.

The planar 200-bar truss structure shown in Fig. 12.5 is designed for minimum weight. Truss elements are divided into 29 groups (design variables) All members are made of steel: the material density and modulus of elasticity are 0.283 lb/in^3

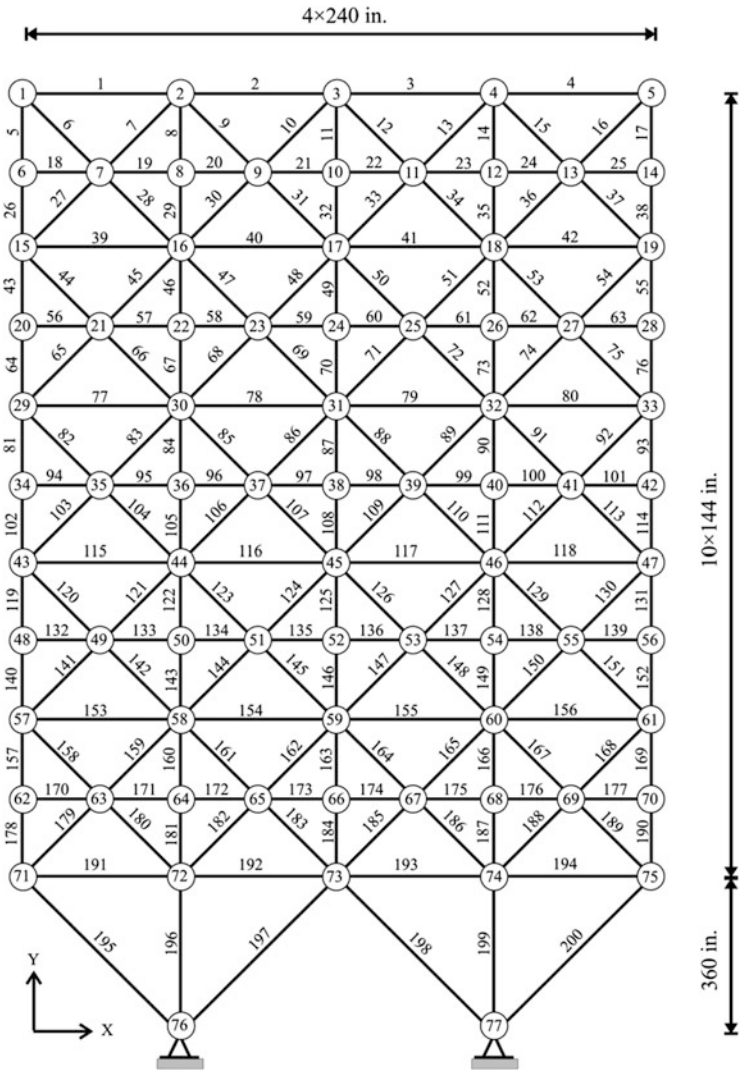

Fig. 12.5 A 200-bar truss structure [37]

(7,933.410 kg/m^3) and 30 Msi (206 GPa), respectively. Element stresses must not exceed $\pm$10 ksi (68.95 MPa). There are three independent loading conditions: (1) 1.0 kip (4.45 kN) acting in the positive x-direction at nodes 1, 6, 15, 20, 29, 34, 43, 48, 57, 62, and 71; (2) 10 kips (44.5 kN) acting in the negative y-direction at nodes 1, 2, 3, 4, 5, 6, 8, 10, 12, 14, 15, 16, 17, 18, 19, 20, 22, 24, . . . , 71, 72, 73, 74, and 75; and (3) conditions 1 and 2 acting together.

The minimum weight and the values of the cross sectional area obtained by CSP and some other previous studies reported in the literature such as a modified

Table 12.3 Comparison of optimized designs for the 200-bar truss

Element group	CMLPSA	GA	SAHS	CSP
$A_{1\sim4}$	0.1468	0.3469	0.154	0.1480
$A_{5,\ 8\ ,\ 11,\ 14,\ 17}$	0.9400	1.0810	0.941	0.9460
$A_{19\sim24}$	0.1000	0.1000	0.100	0.1010
$A_{18,\ 25,\ 56,\ 63,\ 94,\ 101,\ 132,139,\ 170,\ 177}$	0.1000	0.1000	0.100	0.1010
$A_{26,29,32,35,38}$	1.9400	2.1421	1.942	1.9461
$A_{6,\ 7,\ 9,\ 10,\ 12,\ 13,\ 15,\ 16,\ 27,28,\ 30,\ 31,\ 33,\ 34,\ 36,\ 37}$	0.2962	0.3470	0.301	0.2979
$A_{39\sim42}$	0.1000	0.1000	0.100	0.1010
$A_{43,\ 46,\ 49,\ 52,\ 55}$	3.1042	3.5650	3.108	3.1072
$A_{57\sim62}$	0.1000	0.3470	0.100	0.1010
$A_{64,\ 67,\ 70,\ 73,\ 76}$	4.1042	4.8050	4.106	4.1062
$A_{44,\ 45,\ 47,\ 48,\ 50,\ 51,\ 53,\ 54,65,\ 66,\ 68,\ 69,\ 71,\ 72,\ 74,\ 75}$	0.4034	0.4400	0.409	0.4049
$A_{77\sim80}$	0.1912	0.4400	0.191	0.1944
$A_{81,\ 84,\ 87,\ 90,\ 93}$	5.4284	5.9520	5.428	5.4299
$A_{95\sim100}$	0.1000	0.3470	0.100	0.1010
$A_{102,\ 105,\ 108,\ 111,\ 114}$	6.4284	6.5720	6.427	6.4299
$A_{82,\ 83,\ 85,\ 86,\ 88,\ 89,91,\ 92,\ 103,104,\ 106,\ 107,109,\ 110,\ 112,\ 113}$	0.5734	0.9540	0.581	0.5755
$A_{115\sim118}$	0.1327	0.3470	0.151	0.1349
$A_{119,\ 122,\ 125,\ 128,\ 131}$	7.9717	8.5250	7.973	7.9747
$A_{133\sim138}$	0.1000	0.1000	0.100	0.1010
$A_{140,\ 143,\ 146,\ 149,\ 152}$	8.9717	9.3000	8.974	8.9747
$A_{120,\ 121,\ 123,\ 124,\ 126,\ 127,129,130,\ 141,\ 142,\ 144,\ 145,\ 147,\ 148,150,\ 151}$	0.7049	0.9540	0.719	0.70648
$A_{135\sim156}$	0.4196	1.7639	0.422	0.4225
$A_{157,\ 160,\ 163,\ 166,\ 169}$	10.8636	13.3006	10.892	10.8685
$A_{171\sim176}$	0.1000	0.3470	0.100	0.1010
$A_{178,\ 181,\ 184,\ 187,\ 190}$	11.8606	13.3006	11.887	11.8684
$A_{158,\ 159,\ 161,\ 162,\ 164,\ 165,\ 167,168,\ 179,\ 180,\ 182,\ 183,\ 185,\ 186,188,\ 189}$	1.0339	2.1421	1.040	1.035999
$A_{191\sim194}$	6.6818	4.8050	6.646	6.6859
$A_{195,\ 197,\ 198,\ 200}$	10.8113	9.3000	10.804	10.8111
$A_{196,\ 199}$	13.8404	17.1740	13.870	13.84649
Best weight (lb)	25,445.6	28,533.1	25,491.9	25,467.9
Average weight (lb)	N/A	N/A	25,610.2	25,547.6
Std dev (lb)	N/A	N/A	141.85	135.09
No. of analyses	9,650	51,360	19,670	31,700

simulated annealing algorithm (CMLPSA) (Lamberti [38]), an improved GA (Togan and Daloglu [39]), and self adaptive HS (SAHS) (Degertekin [40]) are presented in Table 12.3. It can be seen that the CSP algorithm found an optimum weight of 25,467.95 lb after approximately 317 iterations and 31,700 analyses. The optimal design obtained using the CSP algorithm showed an excellent agreement with the previous designs reported in the literature [37].

As another example, the 26-story-tower space truss with 942 elements and 244 nodes is considered, as shown in Fig. 12.6. Fifty-nine design variables are used to represent the cross-sectional areas of 59 element groups in this structure,

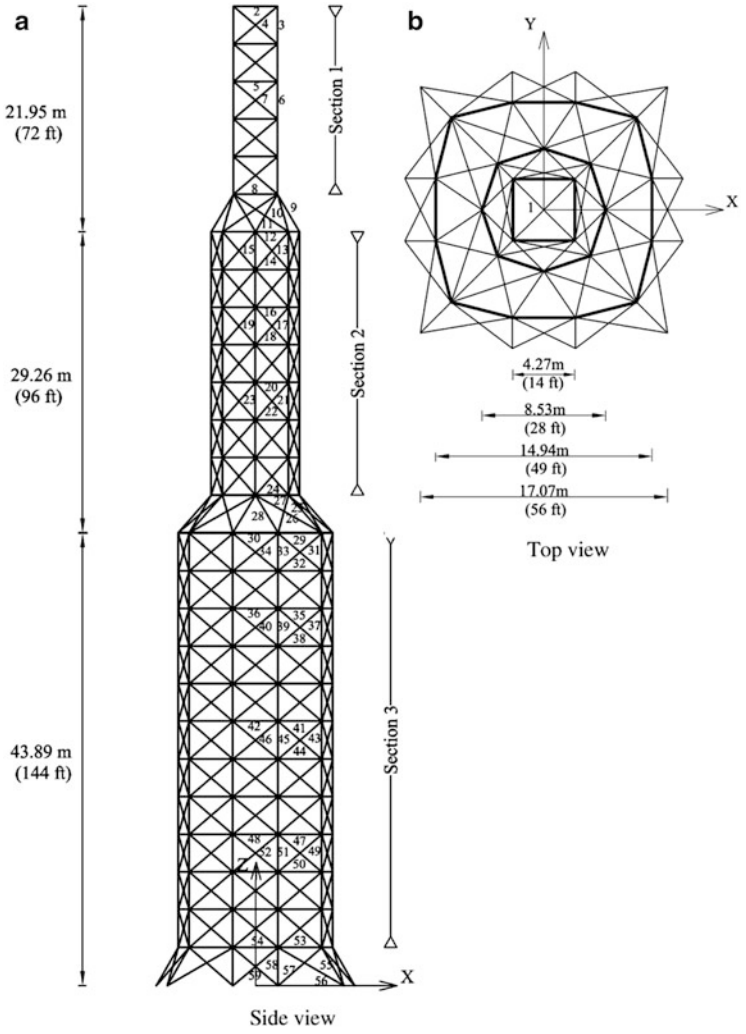

Fig. 12.6 Schematic of a 26-story space tower (**a**) Side view; (**b**) Top view

employing the symmetry of the structure. Figure 12.6 shows the geometry and the 59 element groups. The material density is 0.1 lb/in^3 (2,767.990 kg/m^3) and the modulus of elasticity is 10 Gsi (68.950 GPa). The members are subjected to the stress limits of $\pm$25 ksi (172.375 MPa) and the four nodes of the top level in the x, y and z directions are subjected to the displacement limits of $\pm$15.0 in (38.10 cm) (about 1/250 of the total height of the tower). The allowable cross-sectional areas in this example are selected from 0.1 to 200.0 in^2 (from 0.6452 to 1,290.32 cm^2). Loading conditions are presented in Table 12.4.

Table 12.4 Loading conditions for the spatial 26-story tower

Case number	Direction	Load
1	Vertical	3 kips (13.344 kN)
2	Vertical	6 kips (26.688 kN)
3	Vertical	9 kips (40.032 kN)
4	Horizontal (*X* direction)	1 kips (4.448 kN)
5	Horizontal (*X* direction)	1.5 kips (6.672 kN)
6	Horizontal (*Y* direction)	1 kips (4.448 kN)
7	Horizontal (*Y* direction)	1 kips (4.448 kN)

After 485 iterations and 48,500 analyses, CSP found an optimum weight corresponding to the design listed in Table 12.5. The best weights are 56,343 lb, 60,385 lb, 53,201 lb, and 52,401 lb for the GA, PSO, BB–BC and HBB–BC (Kaveh and Talatahari [41]), respectively. In addition, CSP is more efficient in terms of average weight and standard deviation of optimized weight. The average weight obtained by CSP is 53,147 lb which is 15.94 %, 29.36 %, 3.73 % and 0.72 % lighter than GA, PSO, BB-BC, and HBB-BC, respectively. Table 12.6 provides the statistic information for this example [37].

These results clearly demonstrat the performance of the proposed method with respect to classical and improved variants of metaheuristic algorithms. It has been proven that coupling emergent results in different areas, like those of PSO and complex dynamics, can improve the quality of results in some optimization problems. Furthermore, including chaotic search schemes may be an effective approach.

12.7 Discussion

As an important tool in optimization theory, metaheuristic algorithms explore the search space of the given data in both exploration and exploitation manner and provide a near-optimal solution within a reasonable time. Metaheuristics have many features that make them as suitable techniques not only as standalone algorithms but also to be combined with other optimization methods. Even the standard metaheuristics have been successfully implemented in various applications; however, many modification and improvements to these algorithms have also been reported in the literature. Each of them is tightly related to some aspects of these algorithms such as parameters setting or balancing of exploration and exploitation. In this chapter, we turned the attention to chaos embedded metaheuristic algorithms and survey most of the modifications proposed in the literature.

Chaos is a bounded unstable dynamic behavior that exhibits sensitive dependence on initial conditions and includes infinite unstable periodic motions in nonlinear systems. Recently, the idea of using the benefits of chaotic systems has been noticed in several fields. One of these fields is optimization theory. Experimental studies show the performance of combining chaos and metaheuristics. Here chaos embedded metaheuristics are classified into two general categories. First

Table 12.5 Optimized designs obtained for the 26-story tower

Members	Area	Members	Area	Members	Area
A_1	14.0925	A_{21}	4.3475	A_{41}	0.6235
A_2	8.6965	A_{22}	1.1995	A_{42}	2.9045
A_3	6.1505	A_{23}	6.2555	A_{43}	12.3365
A_4	0.9095	A_{24}	9.2665	A_{44}	1.2195
A_5	0.6245	A_{25}	8.9865	A_{45}	4.9785
A_6	4.6535	A_{26}	4.4975	A_{46}	1.0685
A_7	1.0435	A_{27}	2.9485	A_{47}	0.7465
A_8	13.0025	A_{28}	4.2215	A_{48}	1.4485
A_9	9.4465	A_{29}	5.9315	A_{49}	16.4445
A_{10}	6.7035	A_{30}	9.8325	A_{50}	1.8985
A_{11}	0.6035	A_{31}	13.8705	A_{51}	5.0325
A_{12}	1.2095	A_{32}	1.5125	A_{52}	1.0255
A_{13}	3.0725	A_{33}	3.0985	A_{53}	11.6285
A_{14}	1.0005	A_{34}	1.1185	A_{54}	15.4075
A_{15}	8.2485	A_{35}	0.5965	A_{55}	16.6135
A_{17}	0.7215	A_{37}	1.6875	A_{57}	3.1965
A_{18}	8.2665	A_{38}	8.0155	A_{58}	2.6845
A_{19}	1.0625	A_{39}	1.1215	A_{59}	4.3205
A_{20}	6.5035	A_{40}	4.7895		

Table 12.6 Comparison of optimization results for the 26-story tower

	GA	PSO	BB-BC	HBB-BC	CSP
Best weight (lb)	56,343	60,385	53,201	52,401	52,200
Average weight (lb)	63,223	75,242	55,206	53,532	53,147
Std dev (lb)	6,640.6	9,906.6	2,621.3	1,420.5	1,256.2
No. of analyses	50,000	50,000	50,000	30,000	48,500

category contains the algorithms in which chaos is used instead of random number generators. On the other hand in the second category chaotic search that uses chaotic map is incorporated into metaheuristics to enhance searching behavior of these algorithms and to skip local optima.

Finally a new combination of swarm intelligence and chaos theory is introduced in which the tendency to form swarms appearing in many different organisms and chaos theory has been the source of inspiration, and the algorithm is called CSP. This method is a kind of multi-phase optimization technique which employs chaos theory in two phases, in the first phase it controls the parameter values of the PSO and the second phase is utilized for local search using COA. Compared to those of the other metaheuristic algorithms the performance of the new method can be concluded.

References

1. Sheikholeslami R, Kaveh A (2013) A survey of chaos embedded metaheuristic algorithms. Int J Optim Civil Eng 3:617–633
2. Tavazoei MS, Haeri M (2007) An optimization algorithm based on chaotic behavior and fractal nature. J Comput Appl Math 206:1070–1081
3. Bucolo M, Caponetto R, Fortuna L, Frasca M, Rizzo A (2002) Does chaos work better than noise? IEEE Circ Syst Mag 2(3):4–19
4. Liu B, Wang L, Jin YH, Tang F, Huang DX (2005) Improved particle swarm optimization combined with chaos. Chaos Solitons Fractals 25(5):1261–1271
5. Liu S, Hou Z (2002) Weighted gradient direction based chaos optimization algorithm for nonlinear programming problem. Proceedings of the 4th world congress on intelligent control and automation, pp 1779–1783
6. Li TY, Yorke JA (1975) Period three implies chaos. Am Math Mon 82:985–992
7. Tatsumi K, Obita Y, Tanino T (2009) Chaos generator exploiting a gradient model with sinusoidal perturbations for global optimization. Chaos Solitons Fractals 42:1705–1723
8. Hilborn RC (2000) Chaos and nonlinear dynamics. Oxford University Press, New York
9. Heidari-Bateni G, McGillem CD (1994) A chaotic direct-sequence spread spectrum communication system. IEEE Trans Commun 42(2–4):1524–1527
10. May R (1976) Mathematical models with very complicated dynamics. Nature 261:459–467
11. Peitgen H, Jurgens H, Saupe D (1992) Chaos and fractals. Springer, Berlin
12. Zheng WM (1994) Kneading plane of the circle map. Chaos Solitons Fractals 4(7):1221–1233
13. Dressler U, Farmer JD (1992) Generalized Lyapunov exponents corresponding to higher derivatives. Physica D 59:365–377
14. Zaslavskii GM (1987) The simplest case of a strange attractor. Phys Lett A69(3):145–147
15. Glover F, Kochenberger GA (2003) Handbook of metaheuristic. Kluwer Academic, Boston
16. Talbi EG (2009) Metaheuristics: from design to implementation. Wiley, New Jersey
17. Dorigo M (2009) Metaheuristics network website (2000). http://www.metaheuristics.net/. Visited in January
18. Schuster HG (1988) Deterministic chaos: an introduction (2nd Revised edn). Physick-Verlag GmnH, Weinheim, Federal Republic of Germany
19. Coelho L, Mariani V (2008) Use of chaotic sequences in a biologically inspired algorithm for engineering design optimization. Expert Syst Appl 34:1905–1913
20. Alatas B (2010) Chaotic harmony search algorithm. Appl Math Comput 29(4):2687–2699
21. Alatas B, Akin E, Ozer AB (2009) Chaos embedded particle swarm optimization algorithms. Chaos Solitons Fractals 40:1715–1734
22. Alatas B (2010) Chaotic bee colony algorithms for global numerical optimization. Expert Syst Appl 37:5682–5687
23. Alatas B (2011) Uniform big bang-chaotic big crunch optimization. Commun Nonlinear Sci Numer Simul 16(9):3696–3703
24. Talatahari S, Farahmand Azar B, Sheikholeslami R, Gandomi AH (2012) Imperialist competitive algorithm combined with chaos for global optimization. Commun Nonlinear Sci Numer Simul 17:1312–1319
25. Talataharis S, Kaveh A, Sheikholeslami R (2011) An efficient charged system search using chaos for global optimization problems. Int J Optim Civil Eng 1(2):305–325
26. Talatahari S, Kaveh A, Sheikholeslami R (2012) Chaotic imperialist competitive algorithm for optimum design of truss structures. Struct Multidiscip Optim 46:355–367
27. Talatahari S, Kaveh A, Sheikholeslami R (2012) Engineering design optimization using chaotic enhanced charged system search algorithms. Acta Mech 223:2269–2285
28. Rao RV, Savsani VJ, Vakharia DP (2011) Teaching–learning-based optimization: a novel method for constrained mechanical design optimization problems. Comput Aided Des 43:303–315

29. Yang D, Li G, Cheng G (2007) On the efficiency of chaos optimization algorithms for global optimization. Chaos Solitons Fractals 34:1366–1375
30. Wu TB, Cheng Y, Zhou TY, Yue Z (2009) Optimization control of PID based on chaos genetic algorithm. Comput Simul 26:202–204
31. Guo ZL, Wang SA (2005) The comparative study of performance of three types of chaos immune optimization combination algorithms. J Syst Simul 17:307–309
32. Ji MJ, Tang HW (2004) Application of chaos in simulated annealing. Chaos Solitons Fractals 21:933–941
33. Wang Y, Liu JH (2010) Chaotic particle swarm optimization for assembly sequence planning. Robot Cim-Int Manuf 26:212–222
34. Gao L, Liu X (2009) A resilient particle swarm optimization algorithm based on chaos and applying it to optimize the fermentation process. Int J Inf Syst Sci 5:380–391
35. He Y, Zhou J, Li C, Yang J, Li Q (2008) A precise chaotic particle swarm optimization algorithm based on improved tent map. In Proceedings of the 4th international conference on natural computation, pp 569–573
36. Baykasoglu A (2012) Design optimization with chaos embedded great deluge algorithm. Appl Soft Comput 12(3):1055–1067
37. Kaveh A, Sheikholeslami R, Talatahari S, Keshvari-Ilkhichi M (2014) Chaotic swarming of particles: a new method for size optimization of truss structures. Adv Eng Softw 67:136–147
38. Lamberti L (2008) An efficient simulated annealing algorithm for design optimization of truss structures. Comput Struct 86:1936–1953
39. Togan V, Daloglu AT (2008) An improved genetic algorithm with initial population strategy and self-adaptive member grouping. Comput Struct 86:1204–1218
40. Degertekin SO (2012) Improved harmony search algorithms for sizing optimization of truss structures. Comput Struct 92–93:229–241
41. Kaveh A, Talatahari S (2009) Size optimization of space trusses using big bang-big crunch algorithm. Comput Struct 87:1129–1140

Chapter 13
A Multi-swarm Multi-objective Optimization Method for Structural Design

13.1 Introduction

In this chapter a multi-objective optimization algorithm is presented and applied to optimal design of large-scale skeletal structures [1]. Optimization is a process in which one seeks to minimize or maximize a function by systematically choosing the values of variables from/within a permissible set. In recent decades, a vast amount of research has been conducted in this field in order to design effective and efficient optimization algorithms. Besides, the application of the existing algorithms to engineering design problems has also been the focus of many studies (Gou et al. [2]; Lee and Geem [3]; Gero et al. [4]). In a vast majority of structural design applications, including previous studies (Kaveh and Talatahari [5]; Kaveh and Talatahari [6]; Kaveh and Talatahari [7]; Kaveh and Rahami [8]), the fitness function was based on a single evaluation criterion. For example, the total weight or total construction cost of a steel structural system has been frequently employed as the evaluation criterion in structural engineering applications. But in the practical optimization problems, usually more than one objective are required to be optimized, such as, minimum mass or cost, maximum stiffness, minimum displacement at specific structural points, maximum natural frequency of free vibration, maximum structural strain energy. This makes it necessary to formulate a multi-objective optimization problem, and look for the set of compromise solutions in the objective space. This set of solutions provides valuable information about all possible designs for the considered engineering problem and guides the designer to make the best decision. The application of multi-objective optimization algorithms to structural problems has attracted the interest of many researchers. For example, in (Mathakari et al. [9]) Genetic algorithm is employed for optimal design of truss structures, or in (Liu et al. [10]) Genetic algorithm is utilized for multi-objective optimization for performance-based seismic design of steel moment frame structures, and in (Paya et al. [11]) the problem of design of RC building frames is formulated as a multi-objective optimization problem and solved by simulated annealing. In all these studies, some well-known multi-objective algorithms have

A. Kaveh, *Advances in Metaheuristic Algorithms for Optimal Design of Structures*, 393
DOI 10.1007/978-3-319-05549-7_13, © Springer International Publishing Switzerland 2014

been applied to structural design problems. However, the approach which has attracted the attention of many researchers in recent years is to utilize a high-performance multi-objective optimization algorithm for structural design problems. For example, in (Su et al. [12]) an adaptive multi-island search strategy is incorporated with NSGA-II for solving the truss layout optimization problem, or in (Ohsaki et al. [13]) a hybrid algorithm of simulated annealing and tabu search is used for seismic design of steel frames with standard sections, and in (Omkar et al. [14]) the specific version of particle swarm optimization is utilized to solve design optimization problem of composite structures which is a highly multi-modal optimization problem. These are only three examples of such studies.

Similarly, the main aim of this study is to propose a new high-performance multi-objective optimization algorithm capable of solving large-scale structural design problems with continuous variables. The first step for proposing such an algorithm is to recognize and investigate the characteristics of this group of problems. By reviewing the corresponding literature, the properties of these problems can be summarized as follows:

(i) In these problems usually many design variables accompany the optimization process.
(ii) The fitness function in these problems is usually a multi-modal function, i.e., there are too many local optimal points. Therefore the utilized algorithm should be able to escape from all local optima.
(iii) In this group, the computational cost of fitness function evaluation is really considerable, because usually each fitness function evaluation means a complete analysis and design of a structure. Consequently, the utilized optimization algorithm should be able to find global optimum with lower number of fitness function evaluations.

The next step for proposing such an algorithm is to review the other existing multi-objective optimization algorithms and investigate their capabilities. Over the last decade, in the literature of evolutionary computation, a number of multi-objective evolutionary algorithms have been suggested which are based on different concepts. Some of which are: Multi-objective evolutionary algorithm based on decomposition (MOEA/D) (Zhang and Li [15]), Non-dominated Sorting Genetic Algorithm NSGA-II (Deb et al. [16]), Strength Pareto Evolutionary Algorithm SPEA2 (Zitzler et al. [17]), Multi-objective particle swarm optimization MOPSO (Coello et al. [18]), sMOPSO (Mostaghim and Teich [19]) and cMOPSO (Toscano and Coello [20]) which is a multi-swarm multi-objective optimization algorithm. Additionally, in (Toscano and Coello [20]; Fan and Chang [21]; Yen and Leong [22]; Leong and Yen [23]) the concept of multi-swarm population is used to improve the convergence rate of multi-objective particle swarm optimization. Furthermore, hybrid or memetic optimization algorithms have attracted the attention of many researchers in recent years. For example in (Goh et al. [24]; Sindhya et al. [25]) a gradient based method is incorporated with NSGA-II in order to improve the local search ability of this method and increase its convergence rate. It is clear that in the literature of evolutionary computation, the main approach has

been to design an optimization algorithm that is capable of solving different kinds of optimization problems, and reducing the number of fitness function evaluation was not the main aim. Consequently, some of these algorithms require more than 100,000 evaluations to show their great capability in covering the Pareto fronts.

In this study, the aim is to develop a new multi-objective optimization algorithm that is capable of solving structural design problems, with the above mentioned features, more efficiently than the other available algorithms in the literature. In other words, a multi-objective optimization algorithm which is capable to deal with multi-modal optimization problems having many design variables, and capable of covering the Pareto front with lower number of fitness function evaluation in comparison to other available multi-objective optimization methods. To achieve this goal, a new multi-swarm algorithm, which is composed of three main steps, is designed. In this algorithm the search process is performed by the use of the charge system search (CSS) (Kaveh and Talatahari [6]) procedure while a clustering algorithm is employed for grouping the particles in the search space. Additionally, a particle regeneration procedure is added to improve the algorithm's ability in escaping the local optima.

This chapter describes the proposed architecture of multi-objective optimization algorithm and its applications. Two different groups of problems, i.e. non-constrained with four mathematical problems and constrained with two problems of truss and frame sizing optimization problems, are used to evaluate the performance of the proposed algorithm. A comparison is drawn between our new algorithm and some other well-known multi-objective optimization methods, which are based on Genetic algorithm and particle swarm optimization. To carry out these computations, all optimization algorithms were developed by the use of MATLAB language. It is illustrated that the proposed algorithm, MO-MSCSS, is capable of covering all parts of Pareto front with higher rate of convergence in comparison to other methods [1].

13.2 Preliminaries

For a better understanding of the MOPs, the following concepts are important (Coello et al. [26]) which are summarized here:

Definition 1 (*General Multi-objective Optimization Problem*). In general, multi-objective optimization for minimization problem can be described as:

Find a vector $x = (x_1, x_2, \ldots, x_n)$ which satisfies k inequality constraints as

$$q_i(x) \leq 0 \ \ (i = 1, 2, \ldots, k)$$

and l equality constraints:

$$h_j(x) = 0 \ \ (j = 1, 2, \ldots, l)$$

and minimizes the vector function

$$\text{Min}_{x \in \Omega} \quad F(x) = \{f_1(x), f_2(x), \ldots, f_m(x)\} \tag{13.1}$$

where Ω is a set of decision vectors and m is the number of objectives. In a word, it aims to find vectors subjected to some constraints, which make all the objective values as small as possible.

Definition 2 (*Pareto Dominance*). A vector $\mathbf{u} = (u_1, u_2, \ldots, u_n)$ is said to be *dominate* to another vector $\mathbf{v} = (v_1, v_2, \ldots, v_n)$ (denoted by $\mathbf{u} \prec \mathbf{v}$) if and only if $\mathbf{u}$ is partially less than $\mathbf{v}$, i.e., $\forall\, i \in \{1, 2, \ldots, n\}$, $u_i \leq v_i \wedge \exists\, i \in \{1, 2, \ldots, n\} : u_i < v_i$.

Definition 3 (*Pareto Optimal*). A solution $x \in \Omega$ is said to be Pareto Optimal with respect to Ω if and only if there is no $x' \in \Omega$ for which $\mathbf{v} = (f_1(x'), f_2(x'), \ldots, f_n(x'))$ dominates $\mathbf{u} = (f_1(x), f_2(x), \ldots, f_n(x))$. The phrase *Pareto Optimal* is taken to mean with respect to the entire decision variable space unless otherwise specified.

Definition 4 (*Pareto Optimal Set*). For a given MOP, $F(x)$, the *Pareto Optimal Set* P is defined as

$$P = \left\{ x \in \Omega \middle| \neg \exists x' \in \Omega\, F(x') \prec F(x) \right\}$$

Definition 5 (*Pareto Optimal Front*). For a given MOP, $F(x)$, and Pareto Optimal Set P the *Pareto Front PF* is defined as $PF = \{\mathbf{u} = F(x) | x \in P\}$.

A solution is said to be Pareto Optimal if it is not dominated by any other solutions in the search space, also termed as non-dominated solution. In this chapter, we distinguish the true Pareto Optimal front, termed PF_{true}, and the final set of non-dominated solutions obtained by a multi-objective optimization algorithm, termed PF_{known} as defined by the aim of the multi-objective optimization algorithms is to find a well uniformly distributed PF_{known} that approximates PF_{known} as close as possible.

13.3 Background

The specific features of the charged system search algorithm have motivated us to utilize this method as the main engine of the search process in the MO-MSCSS. A brief detail of this method and an introduction to *k-means* clustering algorithm are presented in Sects. 13.3.1 and 13.3.2, respectively.

13.3.1 Charged System Search

The charged system search is based on electrostatic and Newtonian mechanics laws (Kaveh and Talatahari [6]). The Coulomb and Gauss laws provide the magnitude of the electric field at a point inside and outside a charged insulating solid sphere, respectively, as follows:

$$E_{ij} = \begin{cases} \dfrac{k_e q_i}{a^3} r_{ij} & if\ r_{ij} < a \\ \dfrac{k_e q_i}{{r_{ij}}^2} & if\ r_{ij} \geq a \end{cases} \tag{13.2}$$

where k_e is a constant known as the Coulomb constant; r_{ij} is the separation of the centre of sphere and the selected point; q_i is the magnitude of the charge; and a is the radius of the charged sphere. Using the principle of superposition, the resulting electric force due to N charged spheres is equal to:

$$\mathbf{F}_j = k_e q_j \sum_{i=1}^{N} \left(\pm \frac{q_i}{a^3} r_{ij}.i_1 \pm \frac{q_i}{{r_{ij}}^2}.i_2 \right) \frac{\mathbf{x}_i - \mathbf{x}_j}{\left\| \mathbf{x}_i - \mathbf{x}_j \right\|} \quad \left\langle \begin{matrix} i_1 = 1, i_2 = 0 & if\ r_{ij} < a \\ i_1 = 0, i_2 = 1 & if\ r_{ij} \geq a \end{matrix} \right. \tag{13.3}$$

where the magnitude of the charge q is defined considering the quality of its solution, as follows:

$$q_i = \frac{fit(i) - fitworst}{fitbest - fitworst} \tag{13.4}$$

where, *fitbest* and *fitworst* are the so far best and the worst fitness of all particles, and $fit(i)$ represents the objective function value or the fitness of the particle i; and N is the total number of particles. Each electrical force can be attractive or repulsive, i.e., each particle in the search space is attracted by better particles (with higher fitness value) and is repulsed by worse particles (with lower fitness value). According to this rule, in the first iteration where the particles are far from each other the magnitude of the resultant force acting on a particle is inversely proportional to the square of the separation between the particles. Thus, the exploration power in this condition is high because of performing more searches in the early iterations. It is necessary to increase the exploitation of the algorithm and to decrease the exploration gradually. After a number of searches where particles are collected in a small space, the resultant force becomes proportional to the separation distance of the particles. Therefore, the parameter a separates the global search phase and the local search phase. According to Newtonian mechanics, we have

$$\Delta \mathbf{x} = \mathbf{x}_{new} - \mathbf{x}_{old}, \mathbf{v} = \frac{\mathbf{x}_{new} - \mathbf{x}_{old}}{\Delta t}, \mathbf{a} = \frac{v_{new} - v_{old}}{\Delta t} \tag{13.5}$$

where $\mathbf{x}_{old}$ and $\mathbf{x}_{new}$ are the initial and final position of a particle, respectively; $\mathbf{v}$ is the velocity of the particle; and $\mathbf{a}$ is the acceleration of the particle. Combining the above equations and using Newton's second law, the displacement of any object as a function of time is obtained as

$$\mathbf{x}_{j,new} = k_a . \frac{1}{2} \frac{\mathbf{F}_j}{m_j} \Delta t^2 + k_v . \mathbf{v}_{j,old} . \Delta t + \mathbf{x}_{j,old} \tag{13.6}$$

where, m_j is the mass of the jth particle, which is considered equal to q_j as in the main algorithm (Kaveh and Talatahari [6]). Δt is the time step, and it is set to 1. k_a is the acceleration coefficient; k_v is the velocity coefficient to control the influence of the previous velocity. These coefficients can be considered fixed or adaptive during the search process (Kaveh and Talatahari [5, 7]). Also, we have:

$$\mathbf{v}_{j,new} = \frac{\mathbf{x}_{j,new} - \mathbf{x}_{j,old}}{\Delta t} \tag{13.7}$$

Inspired by the above electrostatic and Newtonian mechanics laws, the concept of the CSS optimization method is organized as follows:

1. Initialization: Initialize an array of particles with random positions. The initial velocities of these particles are taken as zero. Each particle has a charge of magnitude (q) defined considering the quality of its solution. The separation distance r_{ij} between two charged particles i and j is defined as Euclidean distance between them (in search space).
2. Search: The attracting or repulsing force vector for each particle is determined according to (13.3). Where in this equation $\mathbf{F}_j$ is the resultant force affecting the jth particle. After computing resultant forces acting on all particles, each particle is moved to its new position and its velocity is updated. This procedure continuous until the considered stopping criteria ends the search process.

13.3.2 Clustering

Clustering refers to the process of grouping samples so that the samples are similar within each group, these groups are called clusters. Different clustering methods are available in the literature and for this study one of the most prominent clustering methods is utilized, called *k-means* (Haritigan [27]). In this method, first k points are selected as initial centroids, where k is a user specified parameter, namely, the number of clusters desired. Each point is then assigned to the closest centroid, and each collection of points assigned to a centroid is a cluster. The centroid of each cluster is then updated based on the points assigned to the cluster. The process of

assignment and updating is repeated until no point changes clusters, or equivalently, until the centroids remain the same. In this chapter, clustering is done with respect to closeness in the search space.

13.4 MO-MSCSS

Approximation to the Pareto optimal set involves the following two distinct objectives: (1) to obtain a non-dominated front that is close to the true Pareto front and (2) to maintain the diversity of the solutions along the resulting Pareto front. For the problems with very many design variables, application of multi-swarm strategy has resulted in excellent results in both of these objectives, i.e. high convergence rate and maintaining diversity (Fan and Chang [21]; Yen and Leong [22]; Leong and Yen [23]). By employing several swarms, each swarm should cover just part of the Pareto front and this means that for its particles the range of changes for each design variable is limited to a smaller boundary. The great ability of this strategy is seen more when the considered optimization problem has a high number of design variables. On the other hand, by employing several swarms actually each swarm works as a local optimizer. This feature provides an opportunity to utilize a powerful local search algorithm as the search engine in each swarm.

As mentioned above, the application of gradient based algorithms as a local optimizer has been studied by many researchers. It is clear that gradient based algorithms have great potential to be used as local search algorithm in multi-objective optimization algorithms. These results encouraged us to use an evolutionary gradient based algorithm as the local search engine. The CSS algorithm, introduced in Sect. 13.3.1, uses (13.3) to guide the particles in the search space. It is seen that in CSS two terms contribute in guiding each particle in the search space. First, the entered force of the other particles and second particles velocity in the search space. It should be noted that the first term of (13.6) is an approximate estimation of the gradient of the search process. In fact in (Salomon [28]) it is proved that an acceptable estimation of the true gradient direction at point $\vec{x}_t$ can be obtained as follows:

$$\vec{g}_t = \lim_{\lambda \to \infty} \sum_{i=1}^{\lambda} \frac{1}{\lambda} \left(f\left(\vec{t}_i\right) - f\left(\vec{x}_t\right) \right) . \left(\vec{t}_i - \vec{x}_t \right) \tag{13.8}$$

This function is correct when the distance of all $\vec{t}_i$ to $\vec{x}_t$ is less than a specific value (σ_t). It means that all the considered points should be close to each other in the search space. Consequently, the first term of (13.6) guides each particle in the direction of space gradient at its location. The condition of (13.6), i.e. points should be close to each other, is fulfilled here because usually particles of each swarm are very close to each other. This is why CSS algorithm in a limited part of search

space, can be categorized as an evolutionary gradient based algorithm and this is the reason this algorithm is selected as the local search engine in the proposed algorithm.

Employing several swarms for the search process raises the issue of information exchange among swarms. Swarms traverse different parts of the search space and obtain information about the space. Exchanging the information among swarms improves search ability of all the swarms and it helps to maintain diversity of the solutions along the Pareto front (Yen and Daneshyari [29]). Additionally as mentioned above, the structural optimization problems objective function are multimodal functions, i.e. they have lots of local optima, and employed algorithm should be able to escape of these points. The considered solution for both of these problems, is particle regeneration (Fan and Chang [21]). It means the particles in each swarm are regenerated by the use of information provided by the archive members allocated to each swarm. In this way, first it is not required to exchange information among different swarms because in each iteration all swarms are regenerated, second this strategy helps the algorithm to escape from local optima. More details are presented in Sect. 13.4.1.

13.4.1 Algorithm Overview

The main algorithm of MO-MSCSS is quite similar to other multi-swarm algorithms. The generic steps of MO-MSCSS are as follows. First, based upon a preset number of swarms (k_{swarm}), every swarm of particles is initialized. Second, the members of local archive (L-archive) of each swarm are identified by the domination test and the internal iteration is reset to zero. In internal loop, the particles in each swarm will be guided by both the particles from their L-archive and other particles in their swarm (CSS search process). As soon as the force determination process is completed, the particles perform the move operation. Afterward, the L-archive of each swarm is updated. These steps are performed until they reach the maximum internal iteration (*iterIntMax*). By the end of internal loop, the following steps are performed: (1) all of the L-archives are merged into the global archive (G-archive) (2) A clustering algorithm is applied to the G-archive to group the non-dominated particles, where the number of groups is determined by the number of swarms chosen, and each of these groups is assigned to a swarm as its L-archive (3) the population-generation strategy is performed to regenerate the population of each swarm by the use of newly assigned L-archive to the swarm. Then other internal loop starts search process. These steps are performed until they reach the maximum external iteration (*iterExtMax*). Figure 13.1 shows the pseudo-code of the MO-MSCSS. For additional clarification, each part of the algorithm is described further in the following sections.

```
Begin
  Parameters initialization for CSS search algorithm, mutation operator, population generator.
  /* Population Initialization
  Set no. of swarms (k_swarm)
  Set Maximum internal iteration (iterIntMax)
  Set Maximum external iteration (iterExtMax)
  Set internal iteration iterInt=0. Set external iteration iterExt=0.
  For each swarm
      For each particle
          Fitness evaluation.
      EndFor
      Store all found non-dominated particles as members of L-archive.
  EndFor
  iterExt=1;
  While iterExt<iterExtMax
      For each swarm
          While interInt<iterIntMax
              Charge_magnitude_determination()
              For each particle
                  Determine the resultant force exerted to each particle using Eq.(10,11, 12) (CSS search
                  process). Move particle. Fitness evaluation.
                  Apply Mutation operator.
                  Maintain the particles within the search space.
                  Control the velocity of the particles ( v_max = x_max).
              EndFor
          EndWhile
          Store all newly found non-dominated particles in L-archive.
      EndFor
      Combine k_swarm L-archives and update G-archive.
      Apply clustering algorithm to group G-archive.
      Assign obtained groups to swarms as their L-archive.
      Population_generation_strategy()
  EndWhile
End
```

Fig. 13.1 Pseudo-code of the MO-MSCSS [1]

13.4.2 Search Process by CSS Algorithm

In the proposed algorithm, search process is accomplished by CSS algorithm. In this algorithm, as mentioned in Sect. 13.3.1, all particles which exist in a swarm, with respect to their fitness and distance, contribute in guiding a particle. Better particles attract and worse particles repulse the considered particle in the search space. This process is performed in four steps as follows:

Step 1 The Euclidean distance between all the particles in swarm(i) and also particles in archive(i) (in search space) are determined. Then these distances are normalized to R_{allow}. This parameter is considered to overcome the effect of the range of search variables (this parameter is set to 5 in this chapter).

Step 2 The resultant force exerted to each particle is computed using the following expression. This exerting force is formed from three parts:

1. A particle in the swarm is attracted by all archive members. In this case the resultant force is as follows:

$$\mathbf{F}_j = q_j \sum_{i=1}^{l} \left(\frac{Q_i}{a^3} r_{ij}.i_1 + \frac{Q_i}{{r_{ij}}^2}.i_2 \right) (\mathbf{x}(i) - \mathbf{x}(j)) \quad \left\langle \begin{array}{l} i_1 = 1, i_2 = 0 \ \ if \ \ r_{ij} < a \\ i_1 = 0, i_2 = 1 \ \ if \ \ r_{ij} \geq a \end{array} \right. \tag{13.9}$$

   where, $\mathbf{x}(i)$ and $\mathbf{x}(j)$ are the positions of the ith and jth particles, l is the number of archive members in archive (i), Q is the charge magnitude (fitness value) of archive members, q is the charge magnitude of an ordinary particle in swarm(i) and r_{ij} is the distance between the two particles i and j. In this chapter, a is set to 1.
2. A particle is attracted by other better particles in swarm(i), i.e. a particle j is attracted by particle i if and only if the charge magnitude of the particle i is higher than that of the particle j. In this case, the exerting force on each particle is equal to

$$\mathbf{F}_j = q_j \sum_{i=1}^{k} \left(\frac{q_i}{a^3} r_{ij}.i_1 + \frac{q_i}{{r_{ij}}^2}.i_2 \right) (\mathbf{x}(i) - \mathbf{x}(j)) \quad \left\langle \begin{array}{l} i_1 = 1, i_2 = 0 \ \ if \ \ r_{ij} < a \\ i_1 = 0, i_2 = 1 \ \ if \ \ r_{ij} \geq a \end{array} \right., \quad q_i > q_j \tag{13.10}$$

3. A particle is repulsed by other worse particles in swarm(i), i.e., a particle j is repulsed by particle i if and only if the charge magnitude of the particle i is lower than that of the particle j.

$$\mathbf{F}_j = q_j \sum_{i=1}^{k} \left(-\frac{q_i}{a^3} r_{ij}.i_1 - \frac{q_i}{{r_{ij}}^2}.i_2 \right) (\mathbf{x}(i) - \mathbf{x}(j) \quad \left\langle \begin{array}{l} i_1 = 1, i_2 = 0 \ \ if \ \ r_{ij} < a \\ i_1 = 0, i_2 = 1 \ \ if \ \ r_{ij} \geq a \end{array} \right., \quad q_i < q_j \tag{13.11}$$

These three cases are illustrated in Fig. 13.2a, b in which for example in Fig. 13.2a, particle P_1 is attracted by the archive members P'_1, P'_2, P'_3, P'_4 and P'_5 and also is repulsed by P_3, P_4 and P_6. In Fig. 13.2b the particle P_4 is attracted by the archive members P'_1, P'_2, P'_3, P'_4 and P'_5 and by other better swarm particles (P_1 and P_2) and is repulsed by P_3.

Step 3 The new position and velocity of each particle is obtained using the following expression:

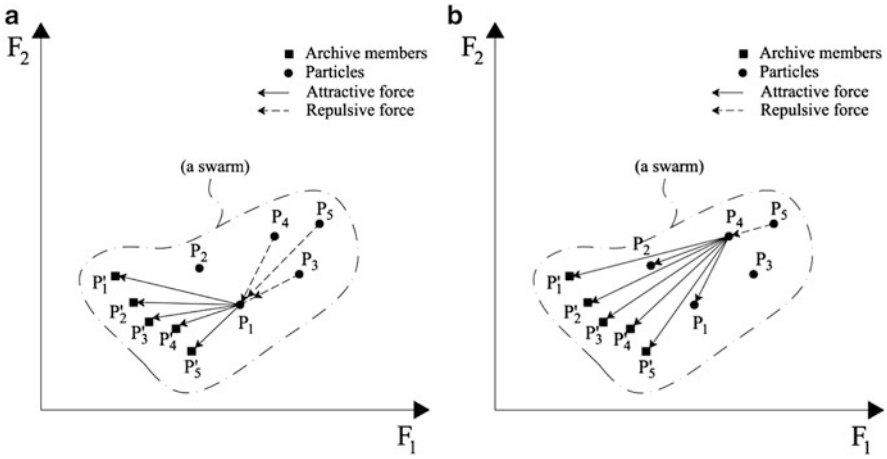

Fig. 13.2 Attraction and repulsion strategies in MO-MSCSS [1] (**a**) Case 1 (**b**) Case 2

$$\mathbf{x}_{j,new} = rand.\frac{\mathbf{F}_j}{m_j} + \omega.\mathbf{v}_{j,old} + \mathbf{x}_{j,old} \tag{13.12}$$

$$\mathbf{v}_{j,new} = \mathbf{x}_{j,new} - \mathbf{x}_{j,old} \tag{13.13}$$

where *rand* is a set of uniformly distributed random numbers in the range [0,1]. m_j is the mass of *j*th particle which is equated to q_j in this chapter.

Step 4 The particles are maintained within the search space when they go beyond their boundaries (Coello et al. [18]). When a design variable goes beyond its boundaries, then two things are done: (1) the decision variable takes the value of its corresponding boundary (either the lower or the upper boundary) and (2) its velocity is multiplied by (-1) so that it searches in the opposite direction.

13.4.3 Charge Magnitude of Particles

The charge magnitudes of the particles are related to their fitness values. The scheme which is employed in the proposed algorithm is similar to the fitness assignment algorithm which is introduced in SPEA2 (Zitzler et al. [17]), but with some modifications.

Here, each particle in the population is assigned a strength value, representing the number of solutions it dominates (P_t is the collection of members of population and $\overline{P}_t$ is the collection of members of the archive):

$$S(i) = \left| \left\{ j \,|\, j \in P_t + \overline{P_t} \wedge i \succ j \right\} \right| \tag{13.14}$$

where, $|.|$ denotes the cardinality of a set, + stands for multi-set union, and the symbol $\succ$ corresponds to the Pareto dominance relation. On the basis of the S values, the raw fitness $R(i)$ of an individual i is calculated as

$$R(i) = \sum_{j \in P_t + \overline{P_t}, j \succ i} S(j) \tag{13.15}$$

According to this methodology, a particle with lower raw fitness is a better solution than the other solutions with higher raw fitness values. In order to reverse this pattern, the charge magnitude of each particle is determined as follows:

$$q_i = \left(\frac{R_{\max} - R_i + \varepsilon}{R_{\max}} \right)^{1/2} \tag{13.16}$$

$$R_{\max} = \max(R(i)) \quad \text{for all } i = 1, 2, \ldots, N \tag{13.17}$$

where ε is a small positive number to avoid zero value for q in (13.16). According to this definition all members of the archive have zero charge and so this method is not verified for classifying these members. In order to provide a measure for qualifying diversity of the archive members, the following charge magnitude is introduced: First the *crowding distance* (Deb et al. [16]) for the archive members is calculated and then members are sorted in descending order according to their *crowding distance*, and the charge magnitude is determined as

$$Q_i = \left(1 + \frac{1}{\text{rank}(\text{archive}(i))} \right)^{1/2} \tag{13.18}$$

where rank(archive(i)) is the rank of archive(i) in the sorted list. Figure 13.3 shows the pseudo-code of the Charge-magnitude-determination strategy.

13.4.4 Population Regeneration

Different methods can be used for population regeneration step. In the proposed algorithm a simple equation is utilized to generate new population. This task is done by the use of archive members which are assigned to each swarm after clustering phase. The pseudo-code of population-generation strategy is presented in Fig. 13.4. In this algorithm one new particle j in a swarm is generated as follows:

```
Function Charge_magnitude_determination (Pop, Arch)
/* Pop= current population of swarm k
/* Arch= archive members of swarm k
Begin
  For each particle in Pop and Arch
      Compute the S value using Eq. 16.
  EndFor
  For each particle in Pop
      Compute the R value using Eq. 17.
      Compute the q value using Eq. 18, 19.
  EndFor
  For each particle in Arch
      Compute crowding distance
      Compute Q using Eq. 20
  EndFor
End
```

Fig. 13.3 Pseudo-code of the charge magnitude determination scheme [1]

$$x_j = h + w.(sw_{\max} - sw_{\min}).randn_j.\left(1 - \frac{iterExt}{iterExtMax}\right) \tag{13.19}$$

where, $randn_j$ is a random number from a standard normal distribution which changes for each particle, sw_{min} and sw_{max} are the minimum and maximum of all search variables in swarm archive(i) respectively, h is one randomly selected member of the swarm archive, and w is a parameter which increase the domain of new generated particles. It should be mentioned that this parameter is considered so that each swarm can cover much more space, and to enable the algorithm to escape from premature convergence. This parameter, in this study, is set to 3 in all examples.

13.4.5 Mutation Operator

CSS is known to have a very high convergence speed (Kaveh and Talatahari [6]). However, such convergence speed may be harmful in the context of multi-objective optimization, because a MO-MSCSS based algorithm may converge to a false Pareto front (i.e., the equivalent of a local optimum in global optimization). Especially in the proposed algorithm, the best solutions found are used to generate and guide particles of the swarms and if archive members get stuck in a local optimum all particles of the swarm are collected around them and whole algorithm will not be able to find the global optimum Pareto front. This drawback of the above optimization method, motivated the development of a mutation operator that tries to explore all of the search space. The choice of a good mutation operator is a difficult task that has a significant impact on performance. In the proposed algorithm,

```
Function Population generation (Arch)
/*Arch=L-archive members of swarm k
Begin
  For each swarm k
      Store the minimum and maximum of all search variables in swarm as swmin and swmax
      For each particle in swarm k
          Select randomly one of the particles in Arch(i)
          Generate one new particle in this swarm using Eq.21.
          Fitness evaluation
      EndFor
      Store newly generated population as particles of swarm(i)
  EndFor
End
```

Fig. 13.4 Pseudo-code of the population-generation scheme [1]

mutation (turbulence) operator is utilized, i.e. mutation operator is applied to each particle j with a predefined probability using the following formulation:

$$x_j^i = x_j^i + R_T x_j^i \tag{13.20}$$

where R_T is a random value in [−1,1]. For determining the probability of applying mutation operator, a nonlinear function, introduced in (Coello et al. [18]), is utilized as follows:

$$rand < (1 - iterInt/iterIntMax)^{5/MutationRate} \tag{13.21}$$

where, *rand* is a uniformly distributed random value in range [0,1]. If this equation is satisfied the mutation operator is applied to one of the search variables of the selected particle. It should be mentioned that this operator is applied in each swarm and *iterInt* and *iterIntMax* are internal iteration index and maximum number of internal iterations respectively.

13.4.6 Global Archive Updating Process

G-archive updating process consists of inserting all the obtained non-dominated solutions in all L-archive(i) ($i = 1, \ldots, k$) into the G-archive and eliminating all dominated solutions. Since the size of the external archive is limited, we apply a secondary mechanism for keeping this limit: We adopt the concept of *crowding distance* (Deb et al. [16]) in order to fix the size of the G-archive. First when non-dominated solutions are inserted into the G-archive, the size of this archive is considered free, After updating the G-archive we proceed to update the crowding values of the set of archive members and sort them in descending order and we eliminate as many members as necessary (from end of the list) in order to avoid exceeding the allowable size of the archive.

13.4.7 Constraint Handling

In order to handle the given constraints, a relatively simple scheme is implemented. Whenever two individuals are compared, first they are checked for constraint violation. If both are feasible, then the non-dominance is directly applied to decide the winner. If one is feasible and the other is infeasible, the feasible dominates. If both are infeasible, then the one with the lowest amount of constraint violation dominates the other. This is the approach that has been utilized in (Deb et al. [16]; Coello et al. [18]) to handle the constraints.

13.5 Structural Optimization

In this section, the AISC-ASD (1989) [30] code is utilized as the structural design code.

13.5.1 Statement of the Considered Optimization Design Problem

The considered structural multi-objective optimization problem can be expressed as follows:

$$\begin{aligned} &\text{minmize } \{ W(\mathbf{X}), U(\mathbf{X}) \} \\ &\text{Subject to } C_j(\mathbf{X}) \leq 0 \quad j = 1, \ldots, h \\ &\mathbf{X}_{\min} \leq \mathbf{X} \leq \mathbf{X}_{\max} \end{aligned} \tag{13.22}$$

where $\mathbf{X}$ is the design variables; $W(\mathbf{X})$ and $U(\mathbf{X})$ are, for example, the weight and displacement of the structure; $C_j(\mathbf{X})$ is the constraint; h is the number of constraints and $\mathbf{X}_{\min}$ and $\mathbf{X}_{\max}$ are the lower bound and upper bounds of design variables, respectively.

13.5.1.1 Design Constraints for Truss Structures

For truss structures, the constraints are as follows:

$$\begin{aligned} &\delta_{\min} \leq \delta_i \leq \delta_{\max} \quad i = 1, 2, \ldots, m \\ &\sigma_{\min} \leq \sigma \leq \sigma_{\max} \quad i = 1, 2, \ldots, n \\ &\sigma_i^b \leq \sigma_i \leq 0 \quad i = 1, 2, \ldots, nc \end{aligned} \tag{13.23}$$

where m is the number of nodes; nc denotes the number of compression elements; σ_i

and δ_i are the stress and nodal deflection, respectively; σ_i^b represents allowable buckling stress in member i when it is in compression.

13.5.1.2 Design Constraints for Frame Structures

For the frame structures, according to the AISC-ASD (1989) [30] code, the constraints are as follows:

The stress limitations:

$$\frac{f_a}{F_a}+\frac{f_{bx}}{F_{bx}}+\frac{f_{by}}{F_{by}}\le 1 \quad \text{For } \frac{f_a}{F_a}\le 0.15 \tag{13.24}$$

$$\frac{f_a}{F_a}+\frac{C_{mx}f_{bx}}{\left(1-\frac{f_a}{F'_{ex}}\right)F_{bx}}+\frac{C_{my}f_{by}}{\left(1-\frac{f_a}{F'_{ey}}\right)F_{by}}\le 1 \quad \text{For } \frac{f_a}{F_a}>0.15 \tag{13.25}$$

$$\frac{f_a}{0.6f_y}+\frac{f_{bx}}{F_{bx}}+\frac{f_{by}}{F_{by}}\le 1 \quad \text{For } \frac{f_a}{F_a}>0.15 \tag{13.26}$$

where f_a $(=P/A_i)$ represents the computed axial stress. The computed flexural stresses due to bending of the member about its major (x) and minor (y) principal axes are denoted by f_{bx} and f_{by}, respectively. F_{ex} and F_{ey} denote the Euler stresses about principal axes of the member that are divided by a factor of safety of 23/12. F_a stands for the allowable axial stress under axial compression force alone, and is calculated depending on elastic or inelastic bucking failure mode of the member according to the slenderness ratio:

$$F_a=\begin{cases}\left(\left(1-\dfrac{\lambda_i^2}{2C_C^2}\right)\Big/\left(\dfrac{5}{3}+\dfrac{3\lambda_i}{8C_C}-\dfrac{\lambda_i^3}{8C_C^3}\right)\right)F_y & \text{For } \lambda_i<C_C\\ \dfrac{12\pi^2E}{23\lambda_i^2} & \text{For } \lambda_i\ge C_C\end{cases} \tag{13.27}$$

where E = the modulus of elasticity; F_y = the yield stress of steel; C_c = the slenderness ratio dividing the elastic and inelastic buckling regions $\left(C_C=\sqrt{2\pi^2E/F_y}\right)$; λ_i = the slenderness ratio $(\lambda_i=kL_i/r_i)$ and k = the effective length factor in which for beam and bracing members, k is taken equal to unity. For column members, alignment charts are furnished in ASD-AISC (AISC 1989) [30] for calculation of k values for both braced and non-braced cases. In this study, however, the following approximate effective length formulas are used based on (Dumonteil [31]), which are as follows:

For non-braced members:

$$k = \sqrt{\frac{1.6G_AG_B + 4(G_A + G_B) + 7.5}{G_A + G_B + 7.5}} \tag{13.28}$$

For braced members:

$$k = \frac{3G_AG_B + 1.4(G_A + G_B) + 0.64}{3G_AG_B + 2(G_A + G_B) + 1.28} \tag{13.29}$$

Additionally, (13.30) represents the slenderness limitations imposed on all members such that maximum slenderness ratio is limited to 300 for members under tension, and to 200 for members under compression loads, i.e., we have:

$$\begin{cases} \lambda_i = \dfrac{k_i L_i}{r_i} \leq 300 & \text{For tension members} \\ \lambda_i = \dfrac{k_i L_i}{r_i} \leq 200 & \text{For compression members} \end{cases} \tag{13.30}$$

13.6 Numerical Examples

In this part, the performance of the proposed algorithm is compared with the performance of some other well-known methods. The examples are categorized into two groups:

1. Unconstrained problems,
2. Constrained problems.

13.6.1 Unconstrained Multi-objective Problems

For this group of problems, there is no structural unconstrained problem available in the literature. Thus the selected problems are mathematical. These problems have been designed in a way that examines the capability of a given multi-objective optimizer in dealing with problems having different characteristics (Coello et al. [26]).

In this study, the number of fitness function evaluation is restricted to 25,000 for this group of problems.

13.6.1.1 Performance Metrics

In order to provide a quantitative assessment for the performance of an multi-objective optimizer, three issues are often taken into consideration (Zitzlet et al. [32]):

- The distance of the resulting non-dominated set to the Pareto-optimal front should be minimized.
- A good (in most cases uniform) distribution of the solutions found is desirable. The assessment of this criterion might be based on a certain distance metric.
- The extent of the obtained non-dominated front should be maximized, i.e., for each objective, a wide range of values should be covered by the non-dominated solutions.

In order to evaluate the produced Pareto front by different methods, in this group three different qualitative measures are utilized.

Generational distance (GD) is a measure of the distance between the true (PF_{true}) and generated Pareto front (PF_{known}). This metric of individual distance representing the distance is given by

$$\mathrm{GD} = \frac{1}{n_{pf}} \left(\sum_{i=1}^{n_{pf}} d_i^2 \right)^{1/2} \tag{13.31}$$

where n_{pf} is the number of members in PF_{known} and d_i is the Euclidean distance between the member i in PF_{known} and its nearest member in PF_{true}. A smaller value of GD implies better convergence.

The metric of spacing (S) gives an indication of how evenly the solutions are distributed along the discovered Pareto-front:

$$\mathrm{S} = \left[\frac{1}{n_{pf} - 1} \sum_{i=1}^{n_{pf}} \left(d_i - \overline{d} \right)^2 \right]^{1/2} \quad \text{where} \quad \overline{d} = \frac{1}{n_{pf}} \sum_{i=1}^{n_{pf}} d_i \tag{13.32}$$

where n_{pf} is the number of members in PF_{known} and d_i is the Euclidean distance (in the objective space) between the member i in PF_{known} and its nearest member in PF_{known}. A smaller value of S implies a more uniform distribution of solutions in PF_{known}.

The metric of maximum spread (MS) measures how "well" the PF_{true} is covered by the PF_{known} through hyper-boxes formed by the extreme function values observed in the PF_{true} and PF_{known}. It is defined as

$$\mathrm{MS} = \left[\frac{1}{m}\sum_{i=1}^{m}\left[\frac{\min\left(f_i^{\max}, F_i^{\max}\right) - \max\left(f_i^{\min} - F_i^{\min}\right)}{F_i^{\max} - F_i^{\min}}\right]^2\right]^{1/2} \tag{13.33}$$

where m is the number of objectives, $f_i^{\max}$ and $f_i^{\min}$ are the maximum and minimum of the ith objective in PF_{known}, respectively, and $F_i^{\max}$ and $F_i^{\min}$ are the maximum and minimum of the ith objective in PF_{true}, respectively. A larger value of MS implies a better spread of solutions. The actual runtime required by the algorithm to complete a fixed number of iterations is named Computational time in this chapter.

13.6.1.2 Comparison of the Results

In this section, extensive empirical studies are conducted to analyze the performance of the proposed MO-MSCSS. It is compared to six of most well-known multi-objective optimization algorithms which are as follows:

Multi-objective Evolutionary Algorithm Based on Decomposition (MOEA/D-DE) (Li and Zhang [33]), Non-dominated Sorting Genetic Algorithm II (Deb et al. [16]), Strength Pareto Evolutionary algorithm SPEA2 (Zitzler et al. [17]), Multi-objective Particle Swarm Optimization MOPSO (Coello et al. [18]), sMOPSO (Mostaghim and Teich [19]) and the other method which is a Multi-swarm multi-objective particle swarm optimization cMOPSO (Toscanoa and Coello [20]).

For this part, four well-known benchmark problems ZDT1, ZDT3, ZDT4 and ZDT6 (Coello et al. [26]) are selected to examine the performance of the proposed algorithm (see Table 13.1), and in order to perform statistical analysis, each problem is solved for 30 times by each of the considered algorithms.

In this study, MOEA/D-DE, a newly introduced multi-objective optimizer, was run using a population size of 100, CR = 0.5, F = 0.5, T = 20 and mutation probability of $1/n$, where n is the population size. A real coded NSGA-II was run using a population size of 100, a crossover probability of 1 ($p_c = 1$), binary tournament selection, a mutation rate of $1/u$ (where $u =$ is the number of decision variables), and distribution indexes for crossover and mutation operators as $\eta_c = 20$ and $\eta_m = 20$, respectively (as recommended in (Deb et al. [16])). SPEA with real variable coding was run using a population size of 100, a crossover probability of 1 ($p_c = 1$), binary tournament, a mutation rate of $1/u$, and distribution indexes for crossover and mutation operators as $\eta_c = 20$ and $\eta_m = 20$, respectively. MOPSO used a population of 100 particles, an archive size of 100 particles, a mutation rate of 0.5, and 50 divisions for the adaptive grid. sMOPSO was run with a population of 100 particles with an archive size of 100 particles and a mutation probability of 0.05. cMOPSO used 40 particles, a maximum number of generations per swarm of 5, and a total of 5 swarms.

Table 13.1 The results of performance metrics for seven employed methods and four example problems

GD Metric						S Metric						MS Metric					
MO Algorithm	Problem	ZDT1	ZDT3	ZDT4	ZDT6	MO Algorithm	Problem	ZDT1	ZDT3	ZDT4	ZDT6	MO Algorithm	Problem	ZDT1	ZDT3	ZDT4	ZDT6
MOEA/D	Mean	0.00093	0.00048	9.62721	0.01566	MOEA/D	Mean	0.00935	0.01958	0.51877	0.00513	MOEA/D	Mean	0.994586	0.879545	66.66165	1.015911
	STD	0.00011	0.00008	2.45500	0.07086		STD	0.00024	0.00143	0.49734	0.00888		STD	0.00073	0.079716	17.28024	0.27294
	Max	0.00119	0.00064	14.94882	0.38713		Max	0.00971	0.02123	2.25934	0.04241		Max	0.995933	0.924799	104.4142	2.325679
	Min	0.00072	0.00038	5.05824	0.00040		Min	0.00883	0.01684	0.00000	0.00187		Min	0.992924	0.73514	34.57226	0.369686
NSGA-II	Mean	0.07314	0.09815	73.98550	0.56782	NSGA-II	Mean	0.00831	0.00903	0.58158	0.01313	NSGA-II	Mean	0.733601	0.620772	506.4044	4.014386
	STD	0.01163	0.01751	7.84662	0.02779		STD	0.00166	0.00192	0.83086	0.01367		STD	0.018095	0.018065	51.65567	0.202852
	Max	0.11046	0.12338	88.74648	0.62581		Max	0.01234	0.01480	4.70492	0.08176		Max	0.777987	0.675591	611.7619	4.380319
	Min	0.05155	0.05642	61.56290	0.51126		Min	0.00610	0.00611	0.05546	0.00484		Min	0.707515	0.598828	422.1283	3.557781
SPEA2	Mean	0.09921	0.12426	74.74094	0.55193	SPEA2	Mean	0.00953	0.01118	0.71556	0.01935	SPEA2	Mean	0.711402	0.611646	512.0635	3.900268
	STD	0.01278	0.01578	6.44831	0.02972		STD	0.00229	0.00410	0.84176	0.02327		STD	0.012859	0.011481	42.94838	0.263581
	Max	0.12846	0.15253	87.13895	0.60043		Max	0.01430	0.02234	4.50154	0.08931		Max	0.749994	0.649541	593.1068	4.361717
	Min	0.07293	0.09041	65.35925	0.45563		Min	0.00572	0.00495	0.11862	0.00527		Min	0.688648	0.598965	449.0035	3.174698
MOPSO	Mean	0.10813	0.10930	20.06437	0.44261	MOPSO	Mean	0.00908	0.01293	0.00069	0.02112	MOPSO	Mean	0.715349	0.615331	141.8311	3.320682
	STD	0.00442	0.00675	2.99151	0.02059		STD	0.00106	0.00180	0.00174	0.02789		STD	0.005046	0.005607	21.1691	0.143465
	Max	0.11646	0.12440	27.15087	0.49871		Max	0.01195	0.01710	0.00785	0.12157		Max	0.725496	0.6313	191.9712	3.610015
	Min	0.09844	0.09216	15.85873	0.40056		Min	0.00749	0.00978	0.00000	0.00000		Min	0.707541	0.605979	112.1028	3.019024
sMOPSO	Mean	0.12092	0.12506	66.39679	0.62909	sMOPSO	Mean	0.01698	0.02090	0.05370	0.00953	sMOPSO	Mean	0.73205	0.587145	457.2982	4.662698
	STD	0.01809	0.02386	6.28641	0.01403		STD	0.00272	0.00693	0.20437	0.01106		STD	0.030108	0.057694	40.11968	0.104652
	Max	0.16109	0.16698	81.36725	0.65730		Max	0.02154	0.03771	0.87887	0.06166		Max	0.805393	0.662961	523.6512	4.896523
	Min	0.08607	0.06924	52.30157	0.60005		Min	0.01181	0.00849	0.00000	0.00004		Min	0.657054	0.460019	368.3342	4.439298
cMOPSO	Mean	0.05751	0.02429	292.74755	0.76520	cMOPSO	Mean	0.00210	0.00263	0.00206	0.00670	cMOPSO	Mean	1.046583	0.656675	373.0403	5.057769
	STD	0.03009	0.03211	77.97355	1.16241		STD	0.00090	0.00144	0.01002	0.01031		STD	0.061859	0.036922	62.67976	0.148057
	Max	0.09878	0.07777	462.28783	6.01224		Max	0.00597	0.00669	0.05502	0.04467		Max	1.171651	0.726169	508.3717	5.275732
	Min	0.00025	0.00013	156.20527	0.23943		Min	0.00120	0.00090	0.00000	0.00048		Min	0.908414	0.558577	191.282	4.615477
MO-MSCSS	Mean	0.02676	0.00032	0.00015	0.00058	MO-MSCSS	Mean	0.00278	0.00433	0.00282	0.00297	MO-MSCSS	Mean	1	0.927747	0.999701	1
	STD	0.00018	0.00003	0.00004	0.00075		STD	0.00023	0.00031	0.00027	0.00075		STD	0	0.002623	0.00041	0
	Max	0.02708	0.00041	0.00023	0.00453		Max	0.00335	0.00490	0.00340	0.00533		Max	1	0.929222	1	1
	Min	0.02633	0.00026	0.00007	0.00040		Min	0.00238	0.00366	0.00246	0.00199		Min	1	0.916328	0.998489	1

In the proposed algorithm, four parameters should be specified by the user, which are as follows:

- Number of particles which contribute in search process (n): In this study 100 particles are utilized to solve optimization problems.
- Number of swarms (k): This parameter should be specified according to n. As mentioned in Sect. 13.4, in each swarm there should be enough number of particles that the employed equation can estimate the gradient of the space with an acceptable precision. We assign 10 particles to each swarm and consequently $n/10$ swarms should be considered.
- Archive size: 100 is considered in this study.
- Maximum number of internal iterations (*iterIntMax*): This parameter controls the power of the proposed algorithm in local search process and by increasing this parameter, more computational effort is consumed for this task. This parameter is considered as 5 in this chapter.

In this section the experimental results are presented in order to clarify the performance of the proposed algorithm.

1. ZDT1: This problem is defined as:

$$\text{ZDT1} : \min \begin{cases} f_1(\mathbf{x}) = x_1 \\ f_2(\mathbf{x}) = g(\mathbf{x}) \left[1 - \sqrt{x_1/g(\mathbf{x})}\right] \\ g(\mathbf{x}) = 1 + 9\left(\sum_{i=2}^{n} x_i\right)/(n-1) \end{cases} \tag{13.34}$$

where $x_i \in [0, 1]$, $i = 1, 2, \ldots, 100$. The Pareto-optimal region corresponds to $x_1^* \in [0, 1]$ and $x_i^* = 0$ for $i = 2, 3, \ldots, 100$. ZDT1 has convex Pareto front which challenge the algorithms' ability to find and produce a quality spread of the Pareto front. Note that the number of decision variables is set to 100 for this two objective test problem instead of the standard number, i.e., 30. This will allow us to exploit all MOs chosen when encountering a higher number of decision variables. The comparison of results between the true Pareto front of ZDT1 and the Pareto front produced by considered algorithms is shown in Fig. 13.5. While the mean value, standard deviation, maximum and minimum value of each of the considered performance metrics are presented in Table 13.1. From the plots of the evolved Pareto fronts in Fig. 13.5 and the results in Table 13.1, it can be observed that except MOEA/D-DE and MO-MSCSS, all other algorithms get stuck in local Pareto optimum and are unable of finding solutions near the global Pareto front (with this number of fitness function evaluation). All algorithms are capable of competitive results in the aspects of S metric. By regarding the MS and GD metric, it is seen that MOEA/D-DE and MO-MSCSS are the best algorithms.

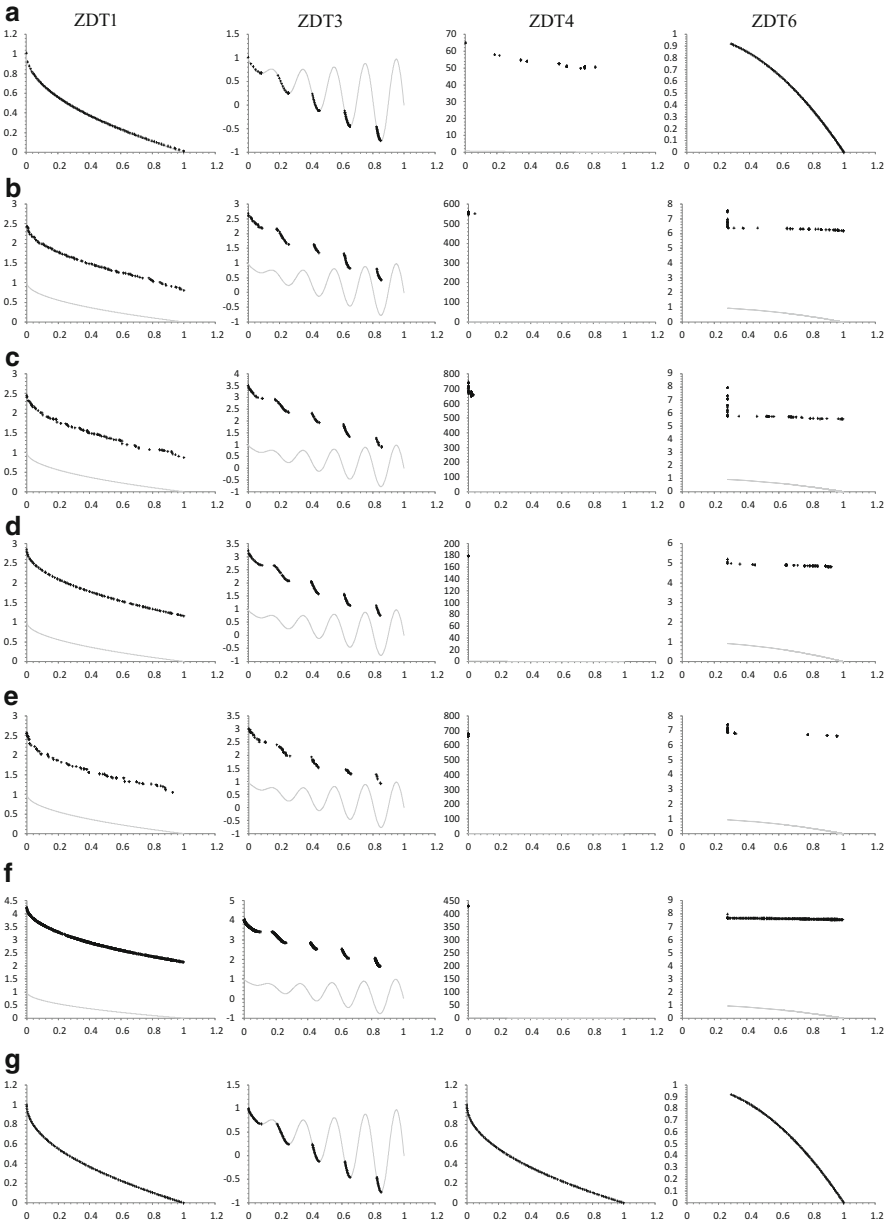

Fig. 13.5 The best obtained results with regard to the three performance metrics by different algorithms for 4 test problems [1] (**a**) MOEA/D-DE (**b**) NSGA-II (**c**) SPEA2 (**d**) MOPSO (**e**) sMOPSO (**f**) cMOPSO (**g**) MO-MSCSS

2. ZDT3: This problem is defined as:

$$\text{ZDT3}: \min \begin{cases} f_1(\mathbf{x}) = x_1 \\ f_2(\mathbf{x}) = g(\mathbf{x}) \left[1 - \sqrt{x_1/g(\mathbf{x})} - \dfrac{x_1}{g(\mathbf{x})} \sin(10\pi x_1)\right] \\ g(\mathbf{x}) = 1 + 9\left(\sum_{i=2}^{n} x_i\right)/(n-1) \end{cases} \tag{13.35}$$

where $x_i \in [0, 1]$, $i = 1, 2, \ldots, 100$. The Pareto-optimal region corresponds to $x_1^* \in [0, 1]$ and $x_i^* = 0$ for $i = 2, 3, \ldots, 100$. ZDT3 is a 100-variable problem which possesses a non-convex and disconnected Pareto front (the number of variables is set to 100 instead of the standard number, i.e., 30). It exploits the algorithms' ability to search for all of the disconnected regions and to maintain a uniform spread on those regions. Figure 13.5 illustrates the comparison of results between the true Pareto front of ZDT3 and the Pareto front produced by different considered algorithms. Also the results of different performance metrics are represented in Table 13.1. From the obtained results it can be seen that NSGA-II, SPEA2, MOPSO, sMOPSO, and cMOPSO have failed to find the true Pareto front for ZDT3 within the specified number of fitness function evaluation. By considering all the performance metrics, it can be seen that the performance of MO-MSCSS is the best among the six algorithms adopted.

3. ZDT4: This problem is defined as:

$$\text{ZDT4}: \min \begin{cases} f_1(\mathbf{x}) = x_1 \\ f_2(\mathbf{x}) = g(\mathbf{x}) \left[1 - \sqrt{x_1/g(\mathbf{x})}\right] \\ g(\mathbf{x}) = 1 + 10(n-1) + \sum_{i=2}^{n} \left[x_i^2 - 10\cos(4\pi x_i)\right] \end{cases} \tag{13.36}$$

where $x_1 \in [0, 1]$ and $x_i \in [-5, 5]$, $i = 2, 3, \ldots, 100$. This is a 100-variable problem which challenges the algorithm ability to deal with the problem of multi-modality (the number of variables is set to 100 instead of the standard number, i.e., 10). ZDT4 has 21^9 different local Pareto-optimal fronts in the search space, of which only one corresponds to the global Pareto-optimal front. The Euclidean distance in the decision space between solutions of two consecutive local Pareto-optimal sets is 0.25. The comparison of results between the true Pareto front of ZDT4 and the Pareto front produced by different considered algorithms is represented in Fig. 13.5. In Table 13.1, three considered performance metrics are represented numerically. It can be observed that all the algorithms, except MO-MSCSS are unable to find any solutions near the global Pareto front resulting in the relatively large GD for ZDT4 at the end of 25,000 evaluations. In this problem, which is similar to structural problems because of its multi-modality, it is clear that the proposed algorithm outperforms all the other considered algorithms.

4. ZDT6: The problem is defined as:

$$\text{ZDT6}: \min \begin{cases} f_1(\mathbf{x}) = 1 - \exp\left(-4x_1\right) \sin^6\left(6\pi x_1\right) \\ f_2(\mathbf{x}) = g(\mathbf{x}) \left[1 - \left(f_1(\mathbf{x})/g(\mathbf{x})\right)^2\right] \\ g(\mathbf{x}) = 1 + 9\left[\left(\sum_{i=2}^{n} x_i\right)/(n-1)\right]^{0.25} \end{cases} \tag{13.37}$$

where $x_i \in [0, 1]$, $i = 1, 2, \ldots, 100$. The Pareto-optimal region corresponds to $x_1^* \in [0, 1]$ and $x_i^* = 0$ for $i = 2, 3, \ldots, 100$. This is a 100-variable problem having a non-convex Pareto-optimal set (Number of variables is set to 100 instead of the standard number, i.e., 10). Moreover, the density of solutions across the Pareto-optimal region is non-uniform and the density towards the Pareto-optimal front is also thin. For this test problem, the adverse density of solutions across the Pareto-optimal front, coupled with the non-convex nature of the front, may cause difficulties for many multi-objective optimization algorithms to converge to the true Pareto-optimal front. The comparison of results between the true Pareto front of ZDT6 and the Pareto front produced all the considered algorithms are represented in Fig. 13.5. By considering the Table 13.1 it can be observed that except MO-MSCSS and MOEA/D-DE, all the other algorithms have problem in finding the global Pareto front. It is clear that considering all the performance metrics, the proposed algorithm outperforms all the other methods.

13.6.2 Constrained Multi-objective Problems

The considered problems in this section are structural optimization problems. In this section in order to evaluate the overall performance of the employed algorithms in solving more complex problems, each problem is solved five times by each algorithm. With regard to obtained results in the previous section it is clear that, except the proposed method, MOEA/D-DE, as the method based on genetic algorithm, and MOPSO, as the method based of particle swarm optimization, outperform all other considered methods. Thus, these two methods are utilized to perform comparative study. But unfortunately the version of MOED/D-DE (Jan and Zhang [34]) for the constrained problems is able to solve just scaled multi-objective problems and for solving un-scaled problems it requires some modifications. Consequently, in this section NSGA-II is selected as the method based on genetic algorithm, to perform the comparative study.

13.6.2.1 The Performance Metrics

In this group of problems, the true Pareto front is not known consequently the considered performance metrics considered in the previous section are not applicable here. In order to evaluate the performance of the algorithms other performance metric, C-metric, is utilized here for comparing the results obtained by different algorithms. Additionally the convergence process of each algorithm is presented graphically.

Set Coverage (C-metric): Let A and B be two approximations to the PF of a MOP, $C(A,B)$ is defined as the percentage of the solutions in B that are dominated by at least one solution in A, i.e.

$$C(A,B) = \frac{\left|\left\{u \in B \middle| \exists v \in A : v \text{ dominates } u\right\}\right|}{|B|} \tag{13.38}$$

$C(A,B)$ is not necessarily equal to $1 - C(B,A)$. $C(A,B) = 1$ means that all the solutions in B are dominated by some solutions in A, and $C(A,B) = 0$ means that no solution in B is dominated by a solution in A.

13.6.2.2 A 126-Bar Truss Structure

This example is a 126-bar spatial truss structure shown in Fig. 13.6. The problem is to find the cross-sectional areas of the members such that the total structural weight (first objective) and the resultant stress in truss members (second objective) are minimized concurrently. In other words, the problem second objective function is defined as follows:

$$Stress\,Index = \sum_{i=1}^{126} \frac{|\sigma_i|}{\sigma_{allowable}} \tag{13.39}$$

The material density is $\rho = 2767.99$ kg/m^3 (0.1 lb/in^3) and the modulus of elasticity is $E = 68,950$ MPa (1×10^4 ksi). The members are subjected to the stress limits of $\pm$ 172.375 MPa ($\pm$25 ksi). The upper and lower boundaries of each truss element are 0.6452 cm^2 (0.1 in^2) and 20.65 cm^2 (3.2 in^2), respectively. The 126 structural members of this spatial truss are sorted into 49 groups. In each story, we have: (1) A_1–A_4, (2) A_5–A_6, (3) A_7–A_8, (4) A_9–A_{10}, (5) A_{11}–A_{12}, (6) A_{13},A_{16}, (7) A_{17}–A_{18}. The applied loads at node 29 are $F_x = 5.0$ kips (22.25 kN), $F_y = 5.0$ kips (22.25 kN) and $F_z = -5.0$ kips (22.25 kN).

In this example, there are 49 design variables. The search process in all the algorithms is terminated after 30,000 fitness function evaluations. Each algorithm is run five times and the best one is selected to present graphically. Additionally, the results of the considered performance metric are presented in Table 13.2. The obtained Pareto fronts from different multi-objective optimization methods are

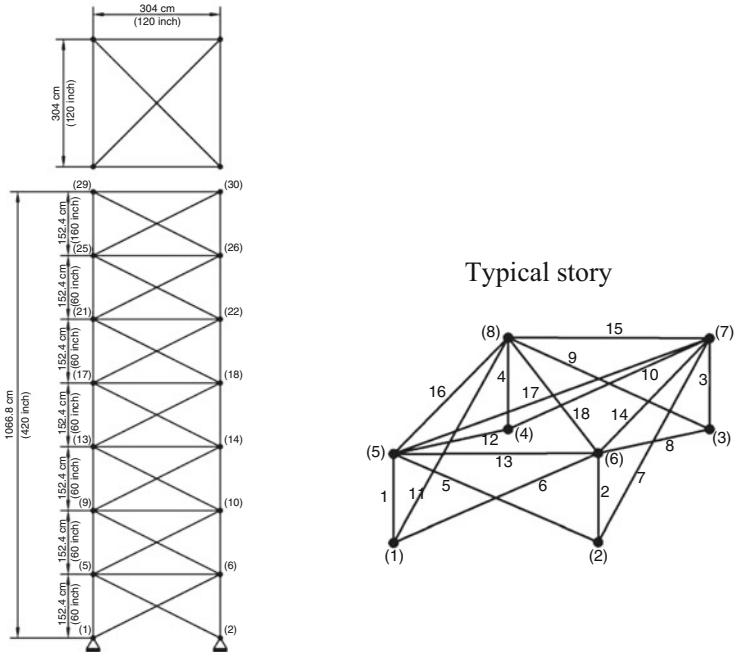

Fig. 13.6 Schematic of a 126-bar spatial truss

presented in Fig. 13.7. In this figure for each algorithm the obtained Pareto front in different iterations is presented which demonstrates the search process in each of the algorithms. Additionally the mean value and standard deviation of C-metric obtained in different runs are presented in Table 13.3. It is seen that in this example with 49 design variables, except the proposed algorithm, all other multi-objective optimizers have some problems in covering the Pareto front. Although MOPSO has acceptable convergence, it is not able to cover all parts of Pareto front and the obtained set of solutions is not distributed uniformly. It is indicated that NSGA-II has problems in converging to true Pareto front and also in covering all parts of it. The obtained results by MO-MSCSS illustrate this algorithm's ability to deal with complex multi-objective optimizations. The convergence to true Pareto front of the proposed algorithm and its ability in covering all parts of it is much better than the other employed algorithms. The obtained cross section areas by MO-MSCSS of two extreme points of Pareto front are presented in Table 13.2. The time required for MO-MSCSS is the best with respect to the other methods.

13.6.2.3 A 36-Story Frame Structure

The second example considered in this chapter is a 36-story un-braced plane steel frame consisting of 259 joints and 468 members, as shown in Fig. 13.8. The

Table 13.2 The cross section area of two extreme solutions in obtained Pareto front by MO-MSCSS (cm^2)

		Section no.	Section area	Section no.	Section area	Section no.	Section area	Section no.	Section area	Section no.	Section area	Section no.	Section area	Section no.	Section area
Extreme point 1	Minimum displacement	1	20.650	8	20.650	15	20.650	22	20.650	29	20.650	36	20.650	43	20.650
		2	20.650	9	20.650	16	20.650	23	20.650	30	20.650	37	20.650	44	20.650
		3	20.650	10	20.650	17	20.650	24	20.650	31	20.650	38	20.650	45	20.650
		4	20.650	11	20.650	18	20.650	25	20.650	32	20.650	39	20.650	46	20.650
		5	20.650	12	20.650	19	20.650	26	20.650	33	20.650	40	20.650	47	20.650
		6	20.650	13	20.650	20	20.650	27	20.650	34	20.650	41	20.650	48	20.650
		7	20.650	14	20.650	21	20.650	28	20.650	35	20.650	42	20.650	49	20.650
Extreme point 2	Minimum weight	1	3.816	8	3.1931	15	3.5200	22	2.1012	29	3.5724	36	0.8676	43	1.2586
		2	0.6500	9	0.6500	16	0.6500	23	0.6500	30	0.6500	37	0.6500	44	0.7900
		3	0.6500	10	0.7020	17	0.6500	24	0.6500	31	0.6500	38	0.7162	45	0.6500
		4	0.6500	11	0.6500	18	0.6500	25	0.6500	32	0.6500	39	0.6500	46	0.6500
		5	0.6500	12	0.6500	19	0.6500	26	0.9097	33	0.6500	40	0.6500	47	1.0100
		6	0.6500	13	0.6500	20	0.6500	27	0.6500	34	0.6500	41	0.6500	48	0.6500
		7	0.6500	14	0.6500	21	0.6500	28	0.6500	35	0.6500	42	0.6500	49	0.6500

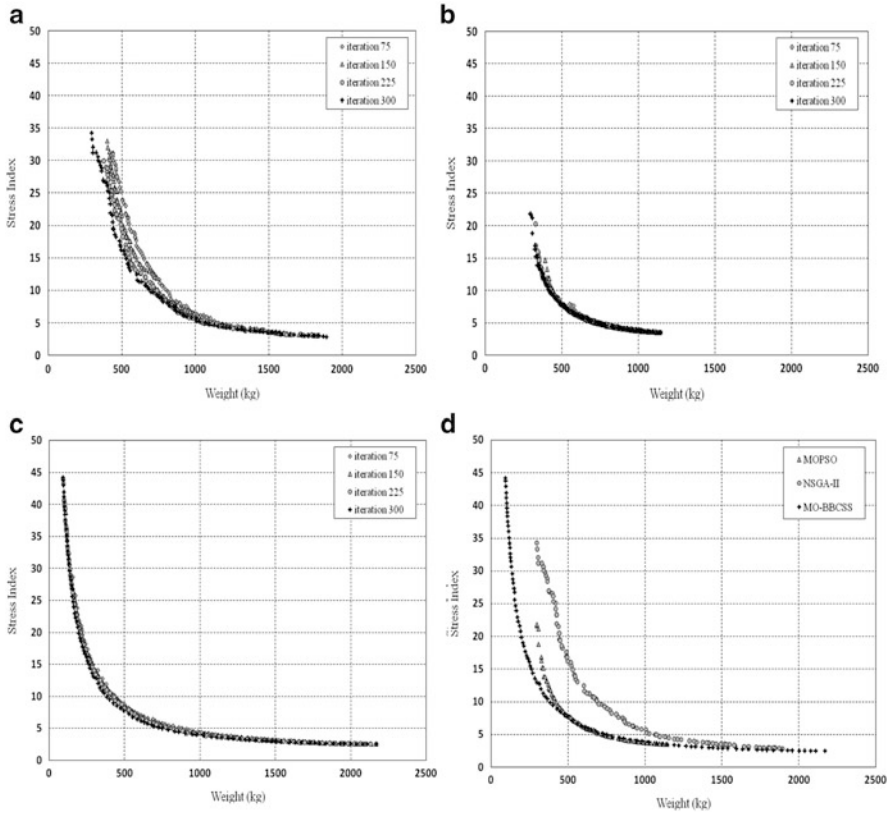

Fig. 13.7 Pareto front at different iteration of (**a**) NSGA-II (**b**) MOPSO (**c**) MO-MSCSS (**d**) All three considered methods of 126-bar truss example [1]

Table 13.3 Mean value and standard deviation of obtained C-metric

126-Bar truss structure				126-Bar truss structure			
Mean C (A,B)	0.978	Std C (A,B)	0.00447	Mean C (A,B)	0.836	Std C (A,B)	0.04147
A: MO-MSCSS B: NSGA-II				A: MO-MSCSS B: MOPSO			

material density is $\rho = 7,850$ kg/m^3 (0.284 lb/in^3), the modulus of elasticity is $E = 203,893.6$ MPa (2.96×10^4 ksi) and the yield stress $f_y = 253.1$ MPa (36.7 ksi).The members are subjected to the stress limits of $\pm$ 172.375 MPa ($\pm$25 ksi). The 468 frame members are collected in 60 different member groups, considering the practical fabrication requirements. That is, the columns in a story are collected in two member groups as inner columns and outer columns, similarly beams are divided into three groups, each two consecutive bays in a group.

The outer columns are grouped together as having the same section over three adjacent stories, as are inner columns, and all beams.

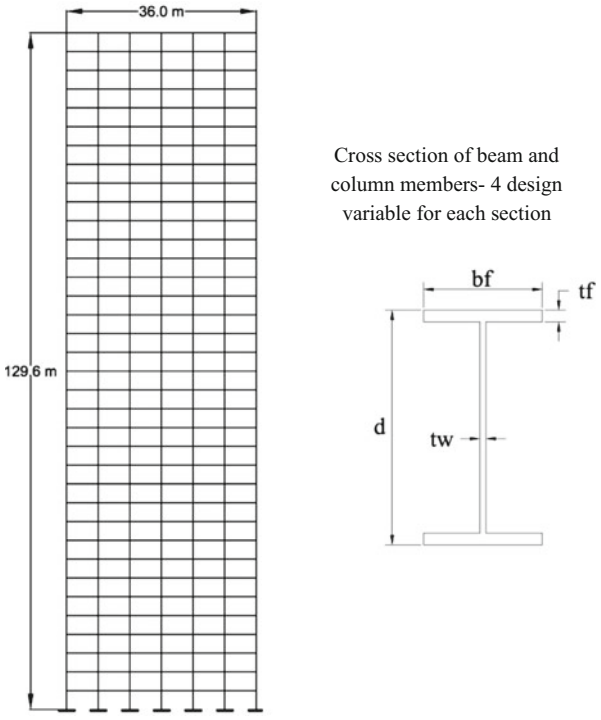

Fig. 13.8 Schematic of a 36-story 2D frame

It should be mentioned that in this example for computing the allowable flexural tensions it is assumed that all beams are laterally supported. The cross section of each member is assumed to be an I-shape and for each member four design variables are considered as shown in Fig. 13.8. In fact in this example we have to consider four design variables for each member, because for each member in addition to cross section, the moment of inertia should be calculated. Consequently, in this example we face with a multi-objective optimization problem with 240 design variables.

The upper and lower boundaries of design variables are 1–7 cm for *tf*, 0.6–2 cm for *tw*, 20–70 cm for *bf* and 10–120 cm for *d*, respectively. This frame is subjected to various gravity loads in addition to lateral wind forces. The gravity loads acting on beams cover dead (D), live (L) and snow (S) loads.

All the floors excluding the roof are subjected to a design dead load of 17.28 kN/m and a design live load of 14.16 kN/m. The roof is subjected to a design dead load of 17.28 kN/m plus snow load. The design snow load is computed using (7.1) in ASCE 7-05 (ASCE 7-05 2005), resulting in a design snow pressure of 4.5 kN/m. The design wind loads (W) are also computed according to ASCE 7-05 using the following equation:

Table 13.4 Applied wind load to beam-column joints

Story	kz	Wind load (kN)—windward face	Wind load (kN)—leeward face	Story	kz	Wind load (kN)—windward face	Wind load (kN)—leeward face	Story	kz	Wind load (kN)—windward face	Wind load (kN)—leeward face
36	1.84	17.87	11.17	24	1.72	16.65	10.41	12	1.52	14.76	9.23
35	1.83	17.78	11.11	23	1.70	16.53	10.33	11	1.50	14.54	9.09
34	1.82	17.69	11.06	22	1.69	16.40	10.25	10	1.48	14.30	8.94
33	1.82	17.60	11.00	21	1.68	16.27	10.17	9	1.45	14.04	8.78
32	1.81	17.51	10.94	20	1.66	16.13	10.08	8	1.42	13.76	8.60
31	1.80	17.41	10.88	19	1.65	15.99	9.99	7	1.39	13.44	8.40
30	1.79	17.31	10.82	18	1.63	15.84	9.90	6	1.35	13.09	8.18
29	1.78	17.21	10.76	17	1.62	15.68	9.80	5	1.31	12.68	7.92
28	1.76	17.11	10.69	16	1.60	15.52	9.70	4	1.26	12.20	7.62
27	1.75	17.00	10.62	15	1.58	15.35	9.59	3	1.20	11.60	7.25
26	1.74	16.89	10.55	14	1.56	15.16	9.48	2	1.11	10.81	6.76
25	1.73	16.77	10.48	13	1.54	14.97	9.36	1	0.99	9.58	5.99

$$p_w = \left(0.613 K_z K_{zt} K_d V^2 I\right)(G C_P) \tag{13.40}$$

where p_w is the design wind pressure in N/m^2; K_z is the velocity exposure coefficient; K_{zt} (=1.0) is the topographic factor, K_d (=0.85) is the wind directionality factor; I (=1.15) is the importance factor; and V (=46.94 m/s) is the basic wind; G (=0.85) is the gust factor, and C_p (=0.8) for windward face and −0.5 for leeward face) is the external pressure coefficient.

The calculated wind loads are applied as concentrated lateral loads on the external beam-column joints (nodes) located on windward and leeward facades at every floor level. The applied loads are summarized in Table 13.4. The load combination per AISC-ASD specification [35] is considered as

$$(\mathrm{D} + \mathrm{L} + \mathrm{S} + \mathrm{W}) \tag{13.41}$$

At the end it should be mentioned that here the aim is to simultaneously minimize two conflicting objective functions, structural weight and the lateral displacement of the roof story due to wind load. In this example, there are 240 design variables and the search process for all the algorithms is terminated after 50,000 fitness function evaluations. Each algorithm is run five times and the best one is selected to present graphically. The obtained Pareto fronts from considered multi-objective optimization methods are presented in Fig. 13.9.

Figure 13.9 presents the Pareto fronts obtained by different algorithms in four stages of search process. Additionally the mean value and standard deviation of C-metric obtained in different runs are presented in Table 13.5. It can be seen that this example is really challenging and, except the proposed algorithm, all other multi-objective optimizers have some deficiencies. As shown in Fig. 13.9, the proposed algorithm outperforms all other mentioned multi-objective optimization algorithms in all different criteria. It is seen that with the specified number of fitness function evaluation just MO-MSCSS is able to cover most parts of the true Pareto front. The time spent by four algorithms is compared in Table 13.6. It can be seen that in this example the time required for MO-MSCSS is approximately equal to the time spent by other mentioned methods.

13.7 Discussions

In this chapter, a new multi-objective optimization algorithm, named as MO-MSCSS, is proposed to deal with complex and large structural optimization problems. These problems have some specific features, and employing general optimization algorithms for solving such problems may cause numerical difficulties, such as finding local optimum solutions instead of the global optimum solution, or taking high amount of computational time. Thus proposing an efficient algorithm for this group of optimization problems can be valuable. In this study,

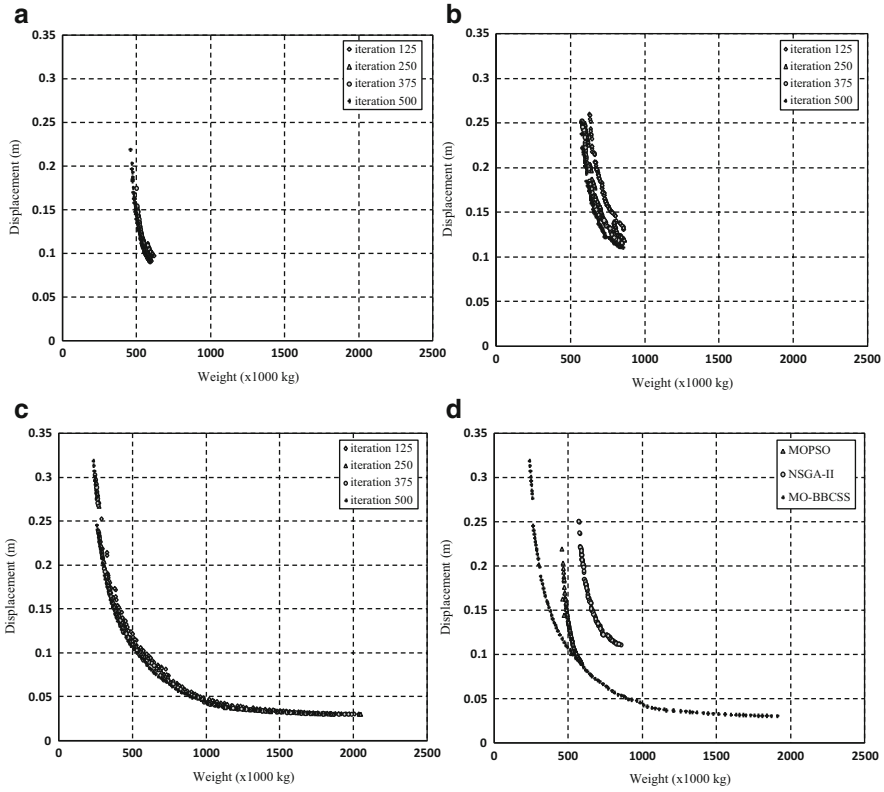

Fig. 13.9 Pareto front at different iteration of (**a**) NSGA-II (**b**) MOPSO (**c**) MO-MSCSS (**d**) Pareto fronts of 2D-frame example [1]

Table 13.5 Mean value and standard deviation of obtained C-metric

2D Frame structure				2D Frame structure			
Mean C (A,B)	1	Std C (A,B)	0	Mean C (A,B)	0.9604	Std C (A,B)	0.061202
A: MO-MSCSS B: NSGA-II				A: MO-MSCSS B: MOPSO			

Table 13.6 Time spent by MOPSO, NSGA-II and MO-MSCSS in three examples (This time is related to the search process in addition to time spent for structural analysis)

Example/algorithm	MOPSO	NSGA-II	MO-MSCSS
126 bar truss structure	200.86 (s)	274.46 (s)	200.03 (s)
36-story frame structure	2370.13 (s)	2118.11 (s)	2401.2 (s)

first we attempt to recognize and categorize the features of structural multi-objective optimization problems, mentioned in the literature by other researchers, and then find the best procedures for their solutions.

In structural optimization problems, the objective functions are multi-modal and a good algorithm will be the one which is capable of escaping the local optima. Additionally in this group of problems, the objective function is defined based on very many design variables and high computational cost is required for each fitness function evaluation. This is another problem that prevents the structural engineers to use the optimization techniques efficiently.

MO-MSCSS algorithm is a hybrid multi-swarm multi-objective optimization method which is based on a swarm-based local search process and the clustering concept. The particle regeneration procedure is another component of the proposed algorithm that helps to escape the local optima. In fact, all of the employed sub-procedures are selected based on their performance and effectiveness in covering the above mentioned problems. The results of the solved examples, both unconstrained and constrained, demonstrate that the proposed algorithm has outstanding abilities in solving large scale multi-objective optimization problems.

References

1. Kaveh A, Laknejadi K (2013) A new multi-swarm multi-objective optimization method for structural design. Adv Eng Softw 58:54–69
2. Gou X, Cheng G, Yamazaki K (2001) A new approach for the solution of singular optima in truss topology optimization with stress and local buckling constraints. Struct Multidiscip Optim 22:364–372
3. Lee KS, Geem ZW (2004) A new structural optimization method based on the harmony search algorithm. Comput Struct 82:781–798
4. Gero MBP, Garc AB, Diaz JJDC (2006) Design optimization of 3D steel structures: genetic algorithms vs. classical techniques. J Construct Steel Res 62:1303–1309
5. Kaveh A, Talatahari S (2009) Optimal design of skeletal structures via the charged system search algorithm. Struct Multidiscip Optim 37:893–911
6. Kaveh A, Talatahari S (2010) A novel heuristic optimization method: charged system search. Acta Mech 213:267–289
7. Kaveh A, Talatahari S (2010) Charged system search for optimum grillage systems design using the LRFD-AISC code. J Construct Steel Res 66:767–771
8. Kaveh A, Rahami H (2006) Nonlinear analysis and optimal design of structures via force method and genetic algorithm. Comput Struct 84:770–778
9. Mathakari S, Gardoni P, Agarwal P, Raich A (2007) Reliability-based optimal design of electrical transmission towers using multi-objective genetic algorithms. Comput Aided Civ Infrastruct Eng 22:282–292
10. Liu M, Burns SA, Wen YK (2005) Multi-objective optimization for performance-based seismic design of steel moment frame structures. Earthq Eng Struct Dynam 34:289–306
11. Paya I, Yepes V, Vidosa FG, Hospitaler A (2008) Multi-objective optimization of concrete frames by simulated annealing. Comput Aided Civ Infrastruct Eng 23:596–610
12. Su RY, Wang X, Gui L, Fan Z (2010) Multi-objective topology and sizing optimization of truss structures based on adaptive multi-island search strategy. Struct Multidiscip Optim 43:275–286
13. Ohsaki M, Kinoshita T, Pan P (2007) Multi-objective heuristic approaches to seismic design of steel frames with standard sections. Earthq Eng Struct Dynam 36:1481–1495

14. Omkar SN, Mudigere D, Naik GN, Gopalakrishnan S (2008) Vector evaluated particle swarm optimization (VEPSO) for multi-objective design optimization of composite structures. Comput Struct 86:1–14
15. Zhang Q, Li H (2007) MOEA/D: a multi-objective evolutionary algorithm based on decomposition. IEEE Trans Evol Comput 11(6):712–731
16. Deb K, Pratap A, Agarwal S, Meyarivan T (2002) A fast and elitist multi-objective genetic algorithm: NSGA-II. IEEE Trans Evol Comput 6:182–197
17. Zitzler E, Laumanns M, Thiele L (2001) SPEA2: improving the strength Pareto evolutionary algorithm. Swiss Federal Institute Technology, Zurich, Switzerland
18. Coello CAC, Pulido GT, Lechuga MS (2004) Handling multiple objectives with particle swarm optimization. IEEE Trans Evol Comput 8:256–279
19. Mostaghim S, Teich J (2004) Covering pareto-optimal fronts by subswarms in multi-objective particle swarm optimization. In Congress on Evolutionary Computation (CEC'2004) 2:1404–1411
20. Toscano PG, Coello CAC (2004) Using clustering techniques to improve the performance of a particle swarm optimizer. In Proceeding of genetic evolutionary computation conference, Seattle, WA, pp 225–237
21. Fan SKS, Chang JM (2010) Dynamic multi-swarm particle swarm optimizer using parallel PC cluster systems for global optimization of large-scale multimodal functions. Eng Optim 42:431–451
22. Yen GG, Leong WF (2009) Dynamic multiple swarms in multi-objective particle swarm optimization. IEEE Trans Syst Man Cybern 39(4):890–911
23. Leong WF, Yen GG (2008) PSO-based multi-objective optimization with dynamic population size and adaptive local archives. Trans Syst Man Cybern 38:1270–1293
24. Goh CK, Ong YS, Tan KC (2008) An investigation on evolutionary gradient search for multi-objective optimization, IEEE world congress on computational intelligence, pp 3741–3746
25. Sindhya K, Sinha A, Deb K, Miettinen K (2009) Local search based evolutionary multi-objective optimization algorithm for constrained and unconstrained problems, CEC'09 Proceeding of the eleventh congress on evolutionary computation
26. Coello CAC, Lechuga MS (2002) MOPSO: a proposal for multiple objective particle swarm optimization. In Proceeding Congress on Evolutionary Computation (CEC'2002) 1:1051–1056
27. Haritigan JA (1975) Clustering algorithms. Wiley, New York, USA
28. Salomon R (1998) Evolutionary algorithms and gradient search: similarities and differences. IEEE Trans Evol Comput 2:45–55
29. Yen GG, Daneshyari M (2006) Diversity-based information exchange among multiple swarms in particle swarm optimization, IEEE congress on evolutionary computation, Canada, pp 1686–1693
30. ASCE 7-05 (2005) Minimum design loads for building and other structures, USA
31. Dumonteil P (1992) Simple equations for effective length factors. Eng J AISC 29(3):111–115
32. Zitzler E, Deb K, Thiele L (2000) Comparison of multi-objective evolutionary algorithms: empirical results. Evol Comput 8(2):173–195
33. Li H, Zhang Q (2009) Multi-objective optimization problems with complicated pareto sets, MOEA/D and NSGA-II. IEEE Trans Evol Comput 13:284–302
34. Jan MA, Zhang Q (2001) MOEA/D for constrained multi-objective optimization: some preliminary experimental results, Comput Intell (UKCI):1–6
35. American Institute of Steel Construction (AISC) (1989) Manual of steel construction-allowable stress design, 9th edn. American Institute of Steel Construction, Chicago, IL