

A. Kaveh

# Advances in Metaheuristic Algorithms for Optimal Design of Structures

*Second Edition*



# Advances in Metaheuristic Algorithms for Optimal Design of Structures

A. Kaveh

# Advances in Metaheuristic Algorithms for Optimal Design of Structures

Second Edition



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 978-3-319-46172-4      ISBN 978-3-319-46173-1 (eBook)  
DOI 10.1007/978-3-319-46173-1

Library of Congress Control Number: 2016956716

© Springer International Publishing AG 2014, 2017

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, express or implied, with respect to the material contained herein or for any errors or omissions that may have been made.

Printed on acid-free paper

This Springer imprint is published by Springer Nature  
The registered company is Springer International Publishing AG  
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

# 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 meta-heuristic 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 meta-heuristics. 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 meta-heuristic algorithms to structural optimization in the last two decades. The present book contains part of this research suitable for various aspects of optimization for skeletal structures.

The 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 development of optimization and different meta-heuristic algorithms. Chapter 2 contains one of the most popular meta-heuristic known as the Particle Swarm Optimization (PSO). Chapter 3 provides an efficient meta-heuristic 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 meta-heuristic 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 Meta-heuristic (CEM) Algorithms. In Chap. 13 the Enhanced Colliding Bodies Optimization (ECBO) is presented. Chapter 14 contains Global Sensitivity Analysis Based (GSAB) optimization method. Chapter 15 presents another recently developed metaheuristic so-called Tug of War Optimization (TWO) method. In Chap. 16, the Water Evaporation Optimization (WEO) is presented that is another new addition to the optimization algorithms. Chapter 17 presents the Vibrating Particle System (VPS) Optimization. In Chap. 18 the Cyclical Parthenogenesis Optimization (CPO) algorithms is presented. Chapter 19 is devoted to optimal design of large scale frame structures. Finally, Chap. 20 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 truss 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. 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, in particular Mr. R.R. Pavan Kumar, the project manager, who prepared the careful page design of this book.

I would like to thank my former and present Ph.D. and M.Sc. students, Dr. A. Zolghadr, Dr. S. Talatahari, Dr. S.M. Massoudi, Dr. N. Farhoudi, Dr. T. Bakhshpoori, Mr. V.R. Mahdavi, Mr. M. Ilchi Ghazaan, Mr. A. Bolandgherami, Mr. R. Sheikholeslami, Mr. M. Khayatazad, Mr. M.A. Motie Share and Mr. S. Sabeti 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, John Wiley and Sons, and Elsevier.

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](mailto:alikaveh@iust.ac.ir).

Tehran  
September 2016

A. Kaveh

# Contents

<b>1</b>	<b>Introduction . . . . .</b>	1
1.1	Metaheuristic Algorithms for Optimization . . . . .	1
1.2	Optimal Design of Structures and Goals of the Present Book . . .	2
1.3	Organization of the Present Book . . . . .	5
	References . . . . .	10
<b>2</b>	<b>Particle Swarm Optimization . . . . .</b>	11
2.1	Introduction . . . . .	11
2.2	PSO Algorithm . . . . .	12
2.2.1	Development . . . . .	12
2.2.2	PSO Algorithm . . . . .	15
2.2.3	Parameters . . . . .	15
2.2.4	Premature Convergence . . . . .	17
2.2.5	Topology . . . . .	19
2.2.6	Biases . . . . .	20
2.3	Hybrid Algorithms . . . . .	21
2.4	Discrete PSO . . . . .	22
2.5	Democratic PSO for Structural Optimization . . . . .	23
2.5.1	Description of the Democratic PSO . . . . .	23
2.5.2	Truss Layout and Size Optimization with Frequency Constraints . . . . .	25
2.5.3	Numerical Examples . . . . .	26
	References . . . . .	40
<b>3</b>	<b>Charged System Search Algorithm . . . . .</b>	45
3.1	Introduction . . . . .	45
3.2	Charged System Search . . . . .	45
3.2.1	Background . . . . .	45
3.2.2	Presentation of Charged Search System . . . . .	49

3.3	Validation of CSS . . . . .	56
3.3.1	Description of the Examples . . . . .	57
3.3.2	Results . . . . .	57
3.4	Charged System Search for Structural Optimization . . . . .	64
3.4.1	Statement of the Optimization Design Problem . . . . .	64
3.4.2	CSS Algorithm-Based Structural Optimization Procedure . . . . .	70
3.5	Numerical Examples . . . . .	70
3.5.1	A Benchmark Truss . . . . .	72
3.5.2	A 120-Bar Dome Truss . . . . .	76
3.5.3	A 26-Story-Tower Space Truss . . . . .	79
3.5.4	An Unbraced Space Frame . . . . .	81
3.5.5	A Braced Space Frame . . . . .	84
3.6	Discussion . . . . .	86
3.6.1	Efficiency of the CSS Rules . . . . .	86
3.6.2	Comparison of the PSO and CSS . . . . .	88
3.6.3	Efficiency of the CSS . . . . .	88
	References . . . . .	88
<b>4</b>	<b>Magnetic Charged System Search . . . . .</b>	<b>91</b>
4.1	Introduction . . . . .	91
4.2	Magnetic Charged System Search Method . . . . .	91
4.2.1	Magnetic Laws . . . . .	92
4.2.2	A Brief Introduction to Charged System Search Algorithm . . . . .	94
4.2.3	Magnetic Charged System Search Algorithm . . . . .	96
4.2.4	Numerical Examples . . . . .	102
4.2.5	Engineering Examples . . . . .	112
4.3	Improved Magnetic Charged System Search . . . . .	118
4.3.1	A Discrete IMCSS . . . . .	119
4.3.2	An Improved Magnetic Charged System Search for Optimization of Truss Structures with Continuous and Discrete Variables . . . . .	120
	References . . . . .	137
<b>5</b>	<b>Field of Forces Optimization . . . . .</b>	<b>139</b>
5.1	Introduction . . . . .	139
5.2	Formulation of the Configuration Optimization Problems . . . . .	140
5.3	Fundamental Concepts of the Fields of Forces . . . . .	140
5.4	Necessary Definitions for a FOF-Based Model . . . . .	142
5.5	An FOF-Based General Method . . . . .	143
5.6	An Enhanced Charged System Search Algorithm for Configuration Optimization . . . . .	144
5.6.1	Review of the Charged System Search Algorithm . . . . .	144
5.6.2	An Enhanced Charged System Search Algorithm . . . . .	146

5.7	Design Examples . . . . .	147
5.7.1	18-Bar Planar Truss . . . . .	147
5.7.2	25-Bar Spatial Truss . . . . .	149
5.7.3	120-Bar Dome Truss . . . . .	152
5.8	Discussion . . . . .	154
	References . . . . .	159
<b>6</b>	<b>Dolphin Echolocation Optimization . . . . .</b>	<b>161</b>
6.1	Introduction . . . . .	161
6.2	Dolphin Echolocation in Nature . . . . .	161
6.3	Dolphin Echolocation Optimization . . . . .	162
6.3.1	Introduction to Dolphin Echolocation . . . . .	162
6.3.2	Dolphin Echolocation Algorithm . . . . .	163
6.4	Structural Optimization . . . . .	173
6.5	Numerical Examples . . . . .	173
6.5.1	Truss Structures . . . . .	173
	References . . . . .	197
<b>7</b>	<b>Colliding Bodies Optimization . . . . .</b>	<b>199</b>
7.1	Introduction . . . . .	199
7.2	Colliding Bodies Optimization . . . . .	199
7.2.1	The Collision Between Two Bodies . . . . .	200
7.2.2	The CBO Algorithm . . . . .	201
7.2.3	Test Problems and Optimization Results . . . . .	206
7.3	CBO for Optimum Design of Truss Structures with Continuous Variables . . . . .	217
7.3.1	Flowchart and CBO Algorithm . . . . .	218
7.3.2	Numerical Examples . . . . .	219
7.3.3	Discussion . . . . .	234
	References . . . . .	235
<b>8</b>	<b>Ray Optimization Algorithm . . . . .</b>	<b>237</b>
8.1	Introduction . . . . .	237
8.2	Ray Optimization for Continuous Variables . . . . .	238
8.2.1	Definitions and Concepts from Ray Theory . . . . .	238
8.2.2	Ray Optimization Method . . . . .	242
8.2.3	Validation of the Ray Optimization . . . . .	247
8.3	Ray Optimization for Size and Shape Optimization of Truss Structures . . . . .	255
8.3.1	Formulation . . . . .	255
8.3.2	Design Examples . . . . .	257
8.4	An Improved Ray Optimization Algorithm for Design of Truss Structures . . . . .	265
8.4.1	Introduction . . . . .	265
8.4.2	Improved Ray Optimization Algorithm . . . . .	267
8.4.3	Mathematical and Structural Design Examples . . . . .	269
	References . . . . .	279

<b>9 Modified Big Bang–Big Crunch Algorithm . . . . .</b>	281
9.1 Introduction . . . . .	281
9.2 MBB–BC Method . . . . .	281
9.2.1 Introduction to BB–BC Method . . . . .	281
9.2.2 A Modified BB–BC Algorithm . . . . .	284
9.3 Size Optimization of Space Trusses Using a MBB–BC Algorithm . . . . .	286
9.3.1 Formulation . . . . .	286
9.3.2 Design Examples . . . . .	288
9.4 Optimal Design of Schwedler and Ribbed Domes Using MBB–BC Algorithm . . . . .	302
9.4.1 Introduction . . . . .	302
9.4.2 Dome Structure Optimization Problems . . . . .	304
9.4.3 Pseudo Code of the Modified Big Bang–Big Crunch Algorithm . . . . .	307
9.4.4 Elastic Critical Load Analysis of Spatial Structures . . . . .	308
9.4.5 Configuration of Schwedler and Ribbed Domes . . . . .	309
9.4.6 Results and Discussion . . . . .	313
9.5 Concluding Remarks . . . . .	318
References . . . . .	319
<b>10 Cuckoo Search Optimization . . . . .</b>	321
10.1 Introduction . . . . .	321
10.2 Optimum Design of Truss Structures Using Cuckoo Search Algorithm with Lévy Flights . . . . .	322
10.2.1 Formulation . . . . .	322
10.2.2 Lévy Flights as Random Walks . . . . .	323
10.2.3 Cuckoo Search Algorithm . . . . .	324
10.2.4 Optimum Design of Truss Structures Using Cuckoo Search Algorithm . . . . .	326
10.2.5 Design Examples . . . . .	328
10.2.6 Discussions . . . . .	338
10.3 Optimum Design of Steel Frames . . . . .	339
10.3.1 Optimum Design of Planar Frames . . . . .	339
10.3.2 Optimum Design of Steel Frames Using Cuckoo Search Algorithm . . . . .	341
10.3.3 Design Examples . . . . .	342
10.3.4 Discussions . . . . .	350
References . . . . .	351
<b>11 Imperialist Competitive Algorithm . . . . .</b>	353
11.1 Introduction . . . . .	353
11.2 Optimum Design of Skeletal Structures . . . . .	354
11.2.1 Constraints for Truss Structures . . . . .	355
11.2.2 Constraints for Steel Frames . . . . .	355

11.3	Imperialist Competitive Algorithm . . . . .	357
11.4	Design Examples . . . . .	361
11.4.1	Design of a 120-Bar Dome-Shaped Truss . . . . .	361
11.4.2	Design of a 72-Bar Spatial Truss . . . . .	363
11.4.3	Design of a 3-Bay 15-Story Frame . . . . .	367
11.4.4	Design of a 3-Bay 24-Story Frame . . . . .	367
11.5	Discussions . . . . .	372
	References . . . . .	373
<b>12</b>	<b>Chaos Embedded Metaheuristic Algorithms . . . . .</b>	<b>375</b>
12.1	Introduction . . . . .	375
12.2	An Overview of Chaotic Systems . . . . .	376
12.2.1	Logistic Map . . . . .	379
12.2.2	Tent Map . . . . .	379
12.2.3	Sinusoidal Map . . . . .	379
12.2.4	Gauss Map . . . . .	380
12.2.5	Circle Map . . . . .	380
12.2.6	Sinus Map . . . . .	380
12.2.7	Henon Map . . . . .	380
12.2.8	Ikeda Map . . . . .	381
12.2.9	Zaslavskii Map . . . . .	381
12.3	Use of Chaotic Systems in Metaheuristics . . . . .	381
12.4	Chaotic Update of Internal Parameters for Metaheuristics . . . . .	382
12.5	Chaotic Search Strategy in Metaheuristics . . . . .	386
12.6	A New Combination of Metaheuristics and Chaos Theory . . . . .	387
12.6.1	The Original PSO . . . . .	388
12.6.2	The CPVPSO Phase . . . . .	389
12.6.3	The CLSPSO Phase . . . . .	390
12.6.4	Design Examples . . . . .	391
12.7	Concluding Remarks . . . . .	396
	References . . . . .	397
<b>13</b>	<b>Enhanced Colliding Bodies Optimization . . . . .</b>	<b>399</b>
13.1	Introduction . . . . .	399
13.2	Structural Optimization . . . . .	399
13.3	An Enhanced Colliding Bodies Optimization (ECBO) . . . . .	401
13.3.1	A Brief Explanation of the CBO Algorithm . . . . .	401
13.3.2	The ECBO Algorithm . . . . .	403
13.4	Mathematical Optimization Problems . . . . .	405
13.5	Design Examples . . . . .	405
13.5.1	A 25-Bar Space Truss . . . . .	409
13.5.2	A 72-Bar Space Truss . . . . .	411
13.5.3	A 582-Bar Tower Truss . . . . .	413
13.5.4	A 3-Bay 15-Story Frame . . . . .	415

13.5.5	A 3-Bay 24-Story Frame . . . . .	417
13.6	Concluding Remarks . . . . .	423
	References . . . . .	424
<b>14</b>	<b>Global Sensitivity Analysis-Based Optimization Algorithm . . . . .</b>	<b>427</b>
14.1	Introduction . . . . .	427
14.2	Background Study . . . . .	428
14.2.1	Variance-Based Sensitivity Indices . . . . .	428
14.2.2	The Variance-Based Sensitivity Analysis Using Space-Partition Method . . . . .	428
14.3	A Global Sensitivity Analysis-Based Algorithm . . . . .	429
14.3.1	Methodology . . . . .	430
14.4	Numerical Examples . . . . .	433
14.4.1	Design of a Tension/Compression Spring . . . . .	433
14.4.2	A Constrained Function . . . . .	438
14.4.3	A Planar 17-Bar Truss Problem . . . . .	439
14.4.4	A 72-Bar Spatial Truss Structure . . . . .	440
14.4.5	A 120-Bar Truss Dome . . . . .	444
14.5	Concluding Remarks . . . . .	446
	References . . . . .	447
<b>15</b>	<b>Tug of War Optimization . . . . .</b>	<b>451</b>
15.1	Introduction . . . . .	451
15.2	Tug of War Optimization Method . . . . .	451
15.2.1	Idealized Tug of War Framework . . . . .	451
15.2.2	Tug of War Optimization Algorithm . . . . .	453
15.3	Mathematical and Engineering Design Problems . . . . .	456
15.3.1	Mathematical Optimization Problems . . . . .	457
15.3.2	Engineering Design Problems . . . . .	457
15.4	Structural Optimization Problems . . . . .	463
15.4.1	Truss Weight Optimization with Static Constraints . . . . .	464
15.4.2	Truss Weight Optimization with Dynamic Constraints . . . . .	473
15.5	Concluding Remarks . . . . .	482
	References . . . . .	486
<b>16</b>	<b>Water Evaporation Optimization Algorithm . . . . .</b>	<b>489</b>
16.1	Introduction . . . . .	489
16.2	Basic Water Evaporation Optimization Algorithm . . . . .	490
16.3	Water Evaporation Optimization with Mixed Phases . . . . .	494
16.4	Test Problems and Optimization Results . . . . .	498
16.4.1	A Spatial 25-Bar Tower Truss . . . . .	499
16.4.2	A Spatial 72-Bar Truss . . . . .	501

16.4.3 A 3-Bay 15-Story Frame . . . . .	503
16.4.4 A 3-Bay 24-Story Frame . . . . .	506
16.5 Concluding Remarks . . . . .	506
References . . . . .	509
<b>17 Vibrating Particles System Algorithm . . . . .</b>	<b>511</b>
17.1 Introduction . . . . .	511
17.2 Formulation of the Structural Optimization Problems . . . . .	512
17.3 The Damped Free Vibration . . . . .	512
17.4 A New Metaheuristic Algorithm Based on the Vibrating Particles System . . . . .	515
17.5 Search Behavior of the Vibrating Particles System Algorithm . . . . .	517
17.6 Test Problems and Optimization Results . . . . .	520
17.6.1 A Spatial 120-Bar Dome-Shaped Truss . . . . .	520
17.6.2 A 200-Bar Planar Truss . . . . .	524
17.6.3 A 3-Bay 15-Story Frame Structure . . . . .	528
17.6.4 A 3-Bay 24-Story Frame Structure . . . . .	531
17.7 Concluding Remarks . . . . .	537
References . . . . .	537
<b>18 Cyclical Parthenogenesis Optimization Algorithm . . . . .</b>	<b>541</b>
18.1 Introduction . . . . .	541
18.2 Cyclical Parthenogenesis Algorithm . . . . .	542
18.2.1 Aphids and Cyclical Parthenogenesis . . . . .	542
18.2.2 Description of Cyclical Parthenogenesis Algorithm . . . . .	543
18.3 Sensitivity Analysis of CPA . . . . .	545
18.4 Test Problems and Optimization Results . . . . .	552
18.4.1 Mathematical Optimization Problems . . . . .	553
18.4.2 Truss Design Problems . . . . .	556
18.5 Concluding Remarks . . . . .	571
References . . . . .	571
<b>19 Optimal Design of Large-Scale Frame Structures . . . . .</b>	<b>573</b>
19.1 Introduction . . . . .	573
19.2 Code-Based Design Optimization of Steel Frames . . . . .	575
19.3 Cascade Sizing Optimization Utilizing a Series of Design Variable Configurations . . . . .	578
19.3.1 Cascade Optimization Strategy . . . . .	578
19.3.2 Multi-DVC Cascade Optimization . . . . .	579
19.4 Colliding Bodies Optimization and Its Enhanced Version . . . . .	581
19.4.1 A Brief Explanation of the CBO Algorithm . . . . .	581
19.4.2 The ECBO Algorithm . . . . .	583
19.5 Numerical Examples . . . . .	586
19.5.1 A 1860-Member Steel Space Frame . . . . .	587
19.5.2 A 3590-Member Steel Space Frame . . . . .	590

19.5.3	A 3328-Member Steel Space Frame . . . . .	594
19.6	Concluding Remarks . . . . .	601
	References . . . . .	601
<b>20</b>	<b>Multi-Objective Optimization of Truss Structures . . . . .</b>	<b>603</b>
20.1	Introduction . . . . .	603
20.2	Multi-Objective Optimization Concepts . . . . .	604
20.3	Charged System Search Algorithm . . . . .	605
20.4	Multi-Objective Charged System Search Optimization Algorithm . . . . .	607
20.4.1	Algorithm . . . . .	607
20.5	Multi-Criteria Decision Making . . . . .	611
20.6	Numerical Examples . . . . .	612
20.6.1	Design of a 2-Bar Truss . . . . .	613
20.6.2	Design of an I-Beam . . . . .	614
20.6.3	Design of a Welded Beam . . . . .	616
20.6.4	Design of a 25-Bar Truss . . . . .	618
20.6.5	Design of a 56-Bar . . . . .	620
20.6.6	Design of a 272-Bar Transmission Tower . . . . .	622
20.7	Concluding Remarks . . . . .	625
	References . . . . .	630

# 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 issues.

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 also 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 minima. 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]. 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 Glover in the paper [5].

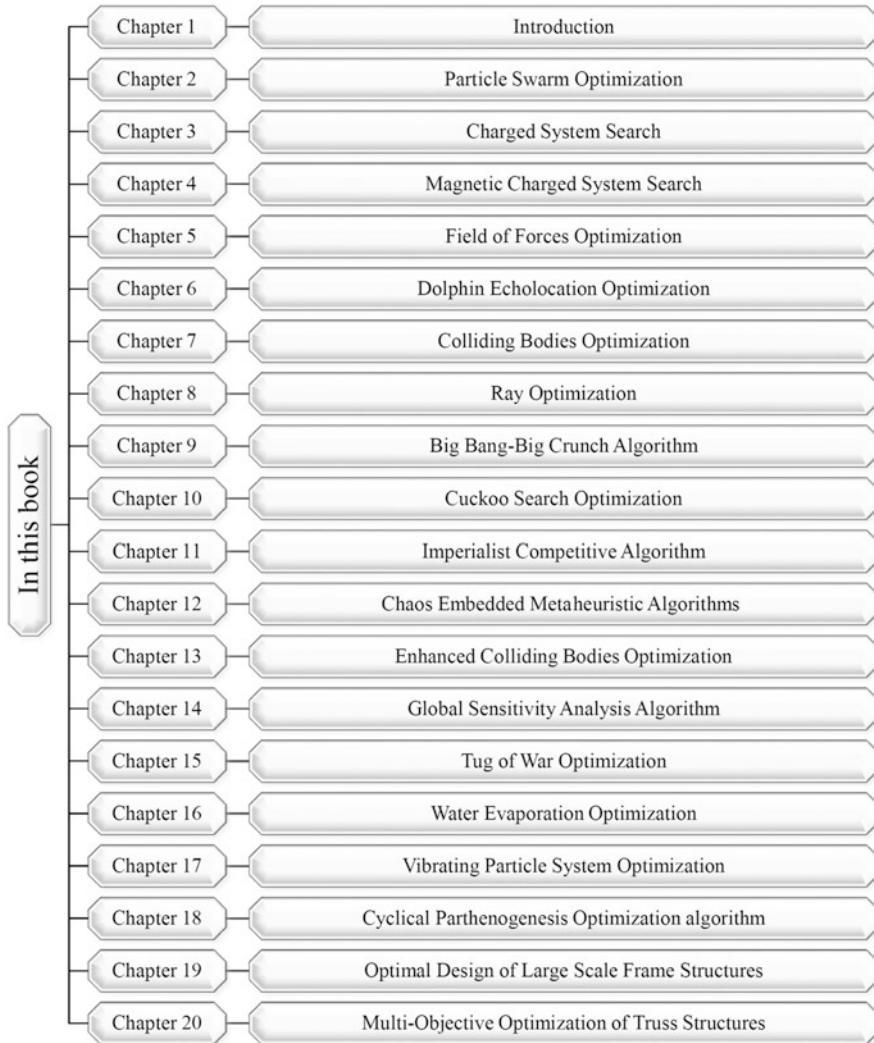
A heuristic method can be considered as a procedure that is likely to discover a very good feasible solution, but not necessarily an optimal solution, for a considered specific problem. No guarantee can be provided about the quality of the solution obtained, but a well-designed heuristic method usually 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 algorithms have 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

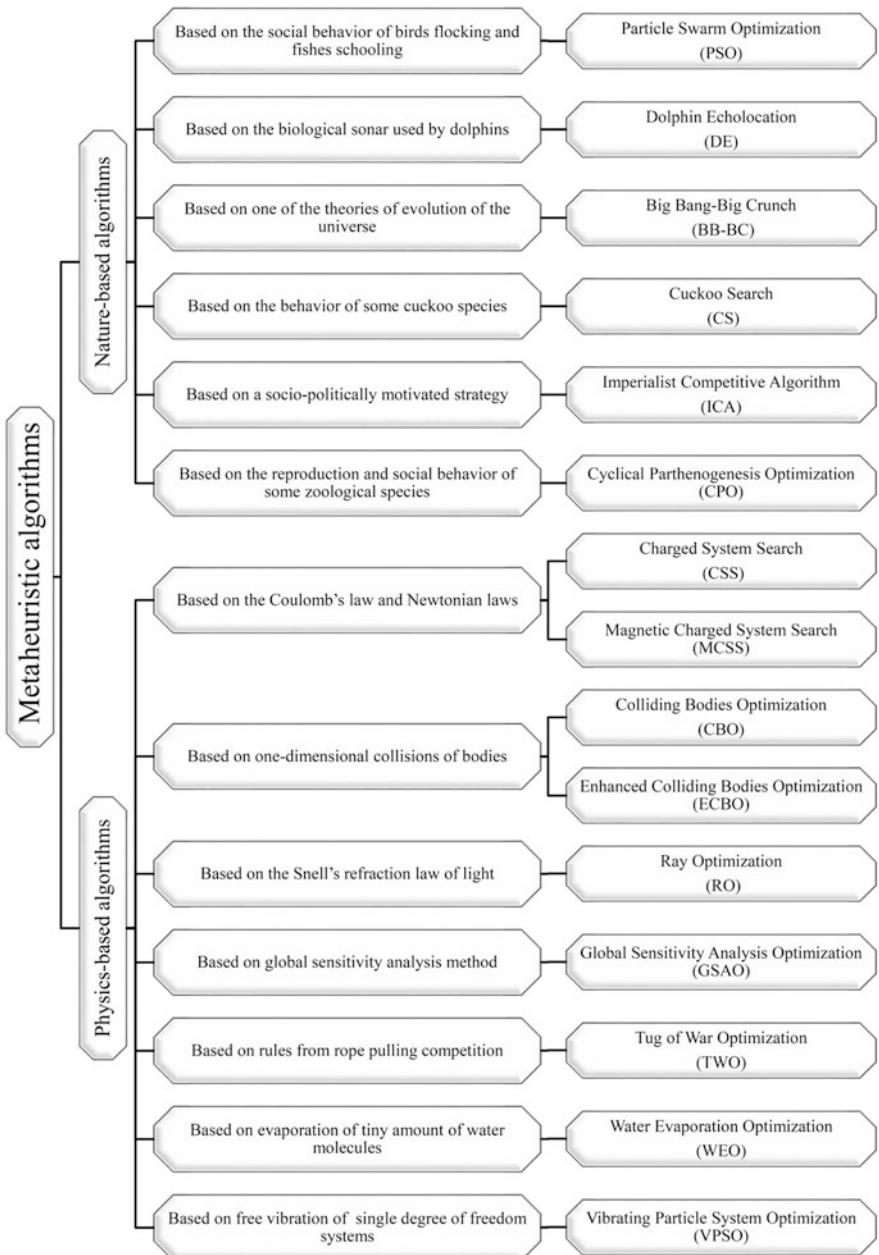
In the area of structural engineering that is the main concern of this book, one tries to achieve certain objectives in order to optimize weight, construction cost, geometry, layout, topology, and time satisfying certain constraints. Since resources, fund, and time are always limited, one has to find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



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

Most of these methods are either nature-based or physics-based algorithm, (Fig. 1.2). Though many design examples are included, however, the results may or may not have small constraint violations and do not constitute the main objective of the book.



**Fig. 1.2** Classification of the metaheuristics presented in this book

## 1.3 Organization of the Present Book

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 and Kennedy. The algorithms mimic the social behavior of birds flocking and fishes schooling. Starting from 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 interparticle communications. These mathematical expressions, in their simplest and most basic form, suggest the movement of each particle toward 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 (CSS) algorithm, 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 governing laws of motion from the Newtonian 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 the governing laws of motion from the Newtonian mechanics. 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 the 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 force 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 progress on the optimization process, the power of rules increases, and the power of

randomization decreases. These rules can be modeled by a familiar concept of physics as well known as the *field of force* (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 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 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.

**Chapter 7** contains the most recently developed algorithm so-called colliding bodies optimization (CBO) proposed by Kaveh and Mahdavi. This novel 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 independent of internal parameters.

**Chapter 8** presents the ray optimization (RO) algorithm originally developed by Kaveh and Khayatazad. 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, 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 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 start 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 discovery 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, forms 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 toward optimum points.

**Chapter 12** is an introduction to chaos embedded metaheuristic algorithms. 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, chaos can help order to arise from disorder. Similarly, many metaheuristic 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. Seemingly the benefits of such combination are a generic for other stochastic

optimization, and experimental studies confirmed this, although this has not mathematically been proven yet.

**Chapter 13** extends the CBO to enhanced colliding bodies optimization (ECBO) which uses a memory to save some best solutions developed by Kaveh and Ilchi Ghazaan. In addition, a mechanism is utilized to escape from local optima. The performance of the proposed algorithm is compared to those of the CBO and some optimization techniques on some benchmark mathematical functions and two standard discrete and continuous structural problems.

**Chapter 14** presents the global sensitivity analysis (GSA) widely used to investigate the sensitivity of the model output with respect to its input parameters. In this chapter, a new single-solution metaheuristic optimization algorithm is presented based on GSA and applied to some benchmark constraint optimization problems and optimal design of truss structures. This algorithm is originally developed by Kaveh and Mahdavi. In this method, the single solution moves toward the specified region, using the sensitivity indicator of variables. Unlike the common metaheuristic algorithms where all the variables are simultaneously changed in the optimization process, in this approach first the high sensitive variables of solution and then the less sensitive ones are iteratively changed in the search space.

**Chapter 15** presents a population-based metaheuristic algorithm inspired by the game of tug of war and originally developed by Kaveh and Zolghadr. Utilizing a sport metaphor, the algorithm, denoted as tug of war optimization (TWO), considers each candidate solution as a team participating in a series of rope pulling competitions. The teams exert pulling forces on each other based on the quality of the solutions they represent. The competing teams move to their new positions according to the governing laws of motion from the Newtonian mechanics. Unlike many other metaheuristic methods, the algorithm is formulated in such a way that considers the qualities of both of the interacting solutions.

**Chapter 16** presents the water evaporation optimization (WEO) as a physics-based metaheuristic algorithm that mimics the well-known rules governing the evaporation process of water molecules from a solid surface with different wettability. This algorithm is originally developed by Kaveh and Bakhshpoori. In the WEO algorithm, molecules are updated globally and locally respectively in two independent sequential phases: monolayer and droplet evaporation phases. In this study, the computational cost of the WEO is improved through the simultaneous utilizing of both phases.

**Chapter 17** presents a metaheuristic algorithm based on free vibration of single degree of freedom systems with viscous damping, and it is called vibrating particles system (VPS). This algorithm is originally developed by Kaveh and Ilchi Ghazaan. The solution candidates are considered as particles that gradually approach to their equilibrium positions. Equilibrium positions are achieved from current population and historically best position in order to have a proper balance between diversification and intensification. To evaluate the performance of the proposed method, it is applied to sizing optimization of four skeletal structures including trusses and frames.

**Chapter 18** presents a nature-inspired population-based metaheuristic algorithm. The algorithm, called cyclical parthenogenesis algorithm (CPA), is inspired by reproduction and social behavior of some zoological species like aphids, which alternate between sexual and asexual reproduction. This algorithm is originally developed by Kaveh and Zolghadr. The algorithm considers each candidate solution as a living organism and iteratively improves the quality of solutions utilizing reproduction and displacement mechanisms. Mathematical and engineering design problems are employed in order to investigate the viability of the proposed algorithm. The results indicate that the performance of the newly proposed algorithm is comparable to other state-of-the-art metaheuristic algorithms.

**Chapter 19** employs the idea of cascade optimization method which allows a single optimization problem to be tackled in a number of successive autonomous optimization stages. In each stage of cascade procedure, a design variable configuration is defined for the problem in a manner that at early stages, the optimizer deals with small number of design variables and at subsequent stages gradually faces with the main problem consisting of a large number of design variables. This algorithm is originally developed by Kaveh and Bolandgerami. In order to investigate the efficiency of this method, in all stages of cascade procedure, the utilized optimization algorithm is the enhanced colliding bodies optimization which is a powerful metaheuristic. Three large-scale space steel frames with 1860, 3590, and 3328 members are investigated for testing the algorithm.

**Chapter 20** consists of a multi-objective optimization method to solve large-scale truss structures in continuous search space that is developed by Kaveh and Massoudi. This method is based on the charged system search, which has been used for single objective optimization in Chap. 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 multimodal 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, and easy to implement; has a good balance between exploration and exploitation, higher capability to avoid being trapped in local optima, higher accuracy, and the applicability 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].

## References

1. Majid KI (1974) Optimum design of structures. Newness-Butterworth, UK
2. Kirsch U (1993) Structural optimization: fundamentals and applications. Springer, Berlin, Heidelberg
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, Norwell, MA
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

# 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]. These algorithms mimic the social behavior of birds flocking and fishes schooling. Starting from 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 performed through moving the particles around the search space by means of a set of simple mathematical expressions which model some interparticle communications. These mathematical expressions, in their simplest and most basic form, suggest the movement of each particle toward 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 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 is the proximity principle: the population should be able to carry out simple space and time computations. Second is the quality principle: the population should be able to respond to quality factors in the environment. Third is the principle of diverse response: the population should not commit its activities along excessively narrow channels. Fourth is the principle of stability: the population should not change its mode of behavior every time the environment changes. Fifth is the principle of adaptability: the population must be able to change behavior mode when it is worth the computational price. They also mention that they compromiseingly call their massless 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 algorithm's 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 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 is 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 nondeterministic 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 were introduced as:

$$v_{i,j}^{k+1} = v_{i,j}^k + c_1 r_1 (x_{best_{i,j}}^k - x_{i,j}^k) + c_2 r_2 (x_{gbest_j}^k - x_{i,j}^k) \quad (2.1)$$

$$x_{i,j}^{k+1} = x_{i,j}^k + v_{i,j}^{k+1} \quad (2.2)$$

where  $x_{i,j}^k$  and  $v_{i,j}^k$  are the jth component of the  $i$ th particle's position and velocity vector, respectively, in the  $k$ th iteration;  $r_1$  and  $r_2$  are two random numbers uniformly distributed in the range (1,0);  $x_{best_i}$  and  $x_{gbest}$  indicate the best positions experienced so far by the  $i$ th particle and the whole swarm, respectively; and  $c_1$  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 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 toward 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 Eq. (2.1) and concluded that the first term, previous velocity of the particle, has an important effect on global and local search balance. By eliminating this term, the particles cannot 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 introduced into Eq. (2.1) in order to maintain balance between these two effects:

$$v_{i,j}^{k+1} = w v_{i,j}^k + c_1 r_1 (x_{lbest_{i,j}}^k - x_{i,j}^k) + c_2 r_2 (x_{gbest_j}^k - x_{i,j}^k) \quad (2.3)$$

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 have 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:

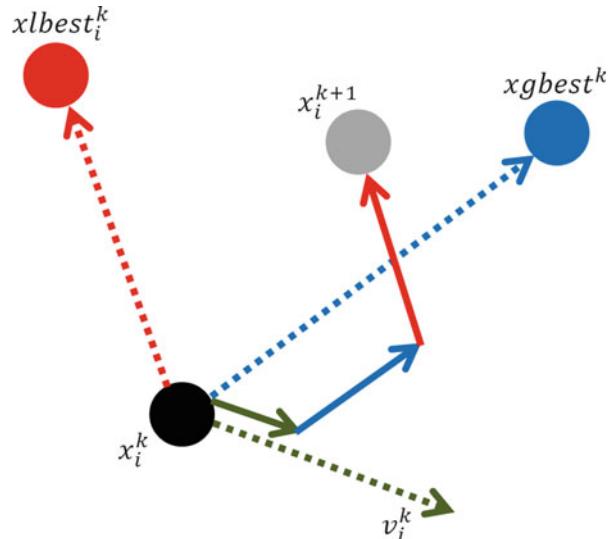
$$v_{i,j}^{k+1} = \chi \left[ v_{i,j}^k + c_1 r_1 (x_{lb\text{est}}_{i,j}^k - x_{i,j}^k) + c_2 r_2 (x_{gb\text{est}}_j^k - x_{i,j}^k) \right] \quad (2.4)$$

$$\chi = \frac{2}{|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}|} \quad \text{where } \varphi = c_1 + c_2, \quad \varphi > 4 \quad (2.5)$$

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).

**Fig. 2.1** Schematic movement of a particle based on Eq. (2.4)



### 2.2.2 PSO Algorithm

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

```

procedure Particle Swarm Optimization
begin
    Initialize  $x_i$ ,  $v_i$  and  $x_{best_i}$  for each particle  $i$ ;
    while (not termination condition) do
        begin
            for each particle  $i$ 
                Evaluate objective function;
                Update  $x_{best_i}$ 
            end
            for each  $i$ 
                Set  $g$  equal to index of neighbor with best  $x_{best_i}$ ;
                Use  $g$  to calculate  $v_i$ ;
                Update  $x_i = x_i + v_i$ ;
                Evaluate objective function;
                Update  $x_{best_i}$ 
            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 in 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 hypersurface. The suitable values for a metaheuristic's parameters depend on relative ruggedness and smoothness of this hyperspace. For example, it is imaginable that in a smoother hyperspace, 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 thorough exploration of the search space will be advisable.

Generally speaking, there are two different strategies for parameter value selection, namely, off-line 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 upon 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 parameters. 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 Vmax to Xmax 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 Vmax to Xmax 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 evaluations that is intended to be performed. Performance evaluation of PSO is performed using some mathematical functions. As mentioned before, the results of the abovementioned off-line parameter tuning studies are all problem dependent and could 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, and 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 abovementioned 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  $x_{\text{gbest}}$  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 suboptimal solution bounce away. A random direction changer, a realistic bounce, and a random velocity changer were 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.

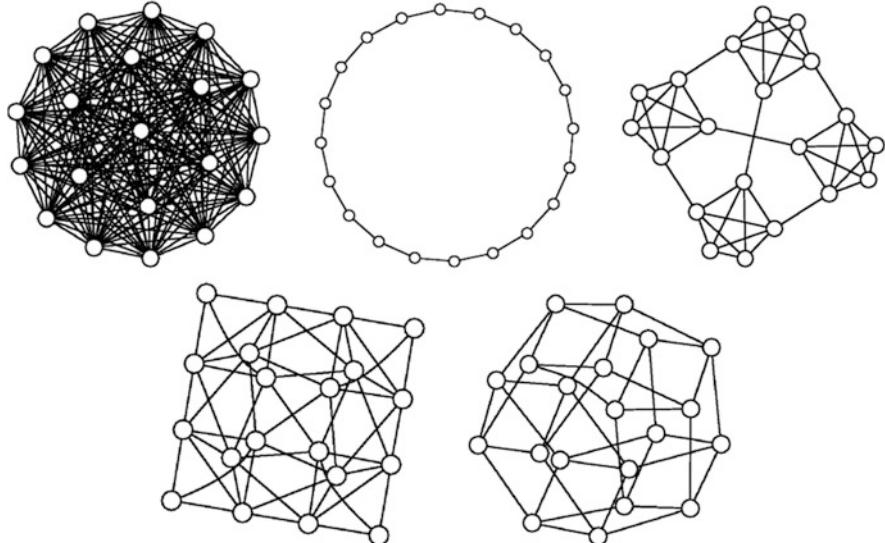
Implementing diversity measures is another way to control swarm stagnation. Riget and Vesterstrøm [25] utilized such a measure along with alternative attraction and repulsion of the particles to and from the swarm best position. Repulsion could be induced by inverting the velocity update rule. The approach 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 algorithm's 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. Twenty-six test functions with different characteristics such as unimodality, multimodality, 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 derives the updating information of a particle from 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 in the problems under consideration.



**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 wrap around on the edges as a torus

### 2.2.5 Topology

While  $x_{\text{best}}$  of Eq. (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 interparticle communications in PSO [29, 30]. In fact in updating a particle's position,  $x_{\text{best}}$  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.

### 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 optimum. 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 an 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 (TS) algorithm [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, Løvbjerg 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 offsprings. Moreover, to keep away from suboptimal solutions, subpopulations were introduced.

Another approach to be mentioned is to use different metaheuristics simultaneously. Krink and Løvbjerg [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 abovementioned 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 updating 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 tuning 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 by 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:

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

in which  $d_{i,j}^k$  is the  $j$ th variable of the vector  $D$  for the  $i$ th particle. The vector  $D$  represents the democratic effect of the other particles of the swarm on the movement of the  $i$ th 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:

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

where  $Q_{ik}$  is the weight of the  $k$ th particle in the democratic movement vector of the  $i$ th 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; and  $E$  is the eligibility parameter and is analogous to parameter  $P$  in CSS [53]. In a minimization problem,  $E$  can be defined as:

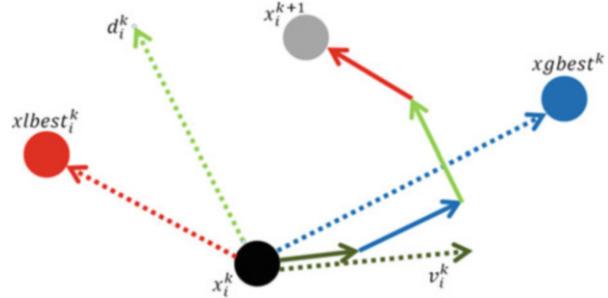
$$E_{ik} = \begin{cases} 1 & \frac{obj(k) - obj(i)}{obj_{worst} - obj_{best}} > rand \vee obj(k) < obj(i) \\ 0 & else \end{cases} \quad (2.9)$$

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.

Since a term is added to the velocity vector of PSO, the parameter  $\chi$  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 Eq. (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

**Fig. 2.3** Schematic movement of a particle based on Eq. (2.6)



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:

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

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; and  $f_{\text{penalty}}(X)$  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;  $\omega_j$  is the  $j$ th natural frequency of the structure and  $\omega_j^*$  is its upper bound.  $\omega_k$  is the  $k$ th natural frequency of the structure and  $\omega_k^*$  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:

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

where  $\rho_i$  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:

$$f_{penalty}(X) = (1 + \epsilon_1 \cdot v)^{\epsilon_2}, \quad v = \sum_{i=1}^q v_i \quad (2.12)$$

where  $q$  is the number of frequency constraints.

$$v_i = \begin{cases} 0 & \text{if the } i\text{th constraint is satisfied} \\ \left| 1 - \frac{\omega_i}{\omega_i^*} \right| & \text{else} \end{cases} \quad (2.13)$$

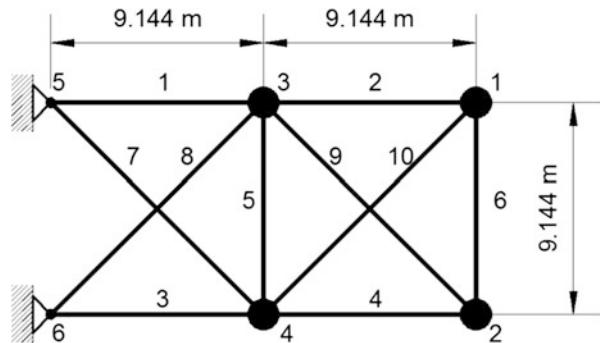
The parameters  $\epsilon_1$  and  $\epsilon_2$  are selected considering the exploration and the exploitation balance of the problem. In this study,  $\epsilon_1$  is taken as unity, and  $\epsilon_2$  starts from 1.5 and linearly increases to 6 in all test examples. These values penalize the unfeasible 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 proposed algorithm 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  $\chi$  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. A total number of 200 iterations are considered for all of the examples. The side constraints are handled using an

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



HS-based constraint handling technique, as introduced by Kaveh and Talatahari [47]. 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 areas is assumed to be an independent variable. A nonstructural 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] using 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–BBC 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 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 2.1** Material properties, variable bounds, and frequency constraints for the 10-bar truss structure

Property/unit	Value
E(modulus of elasticity)/N/m <sup>2</sup>	$6.89 \times 10^{10}$
$\rho$ (material density)/kg/m <sup>3</sup>	2770.0
Added mass/kg	454.0
Design variable lower bound/m <sup>2</sup>	$0.645 \times 10^{-4}$
Design variable upper bound/m <sup>2</sup>	$50 \times 10^{-4}$
L(main bar's dimension)/m	9.144
Constraints on the first three frequencies/Hz	$\omega_1 \geq 7, \omega_2 \geq 15, \omega_3 \geq 20$

**Table 2.2** Optimized designs (cm<sup>2</sup>) 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]	Proposed algorithm
						Standard CSS	
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

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

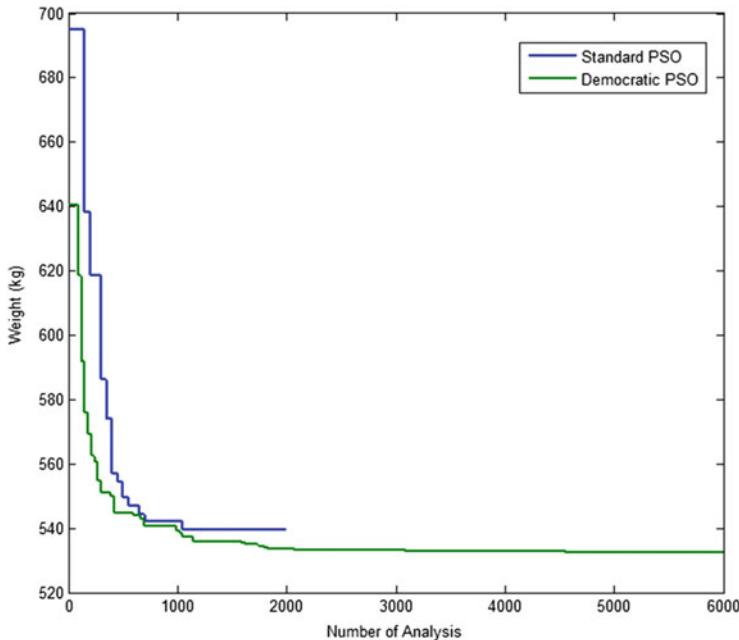
Figure 2.5 compares the convergence curves for the 10-bar planar truss obtained by the democratic PSO and the standard PSO.

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 2000th 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

**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	Proposed algorithm
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	20.598	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



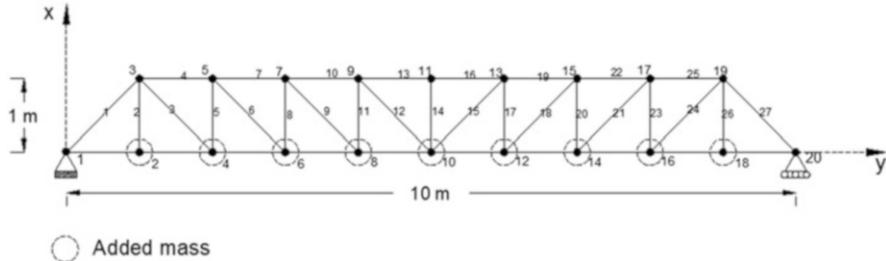
**Fig. 2.5** Comparison of convergence curves of democratic and standard PSO algorithms recorded in the 10-bar problem

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 1000 analyses and after that the convergence curve becomes straight. On the other hand, democratic PSO reaches an initial convergence after about 1500th analyses, and it still keeps exploring the search space until it reaches the final result at 3000th 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 2000th 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.

### 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  $4 \times 10^{-3} \text{ m}^2$ . 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



**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 <sup>2</sup>	$2.1 \times 10^{11}$
$\rho$ (material density)/kg/m <sup>3</sup>	7800
Design variable lower bound/m <sup>2</sup>	$1 \times 10^{-4}$
Design variable upper bound/m <sup>2</sup>	$10 \times 10^{-4}$
Added mass/kg	10
Constraints on first three frequencies/Hz	$\omega_1 \geq 20, \omega_2 \geq 40, \omega_3 \geq 60$

lower chord, a nonstructural 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 19 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 constraints, and added masses are listed in Table 2.4.

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).

**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	Proposed algorithm
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 ( $\text{cm}^2$ )	3.2508	2.8932	2.6797	2.9082	2.6208
A2, A26 ( $\text{cm}^2$ )	1.2364	1.1201	1.1568	1.0212	1.0397
A3, A24 ( $\text{cm}^2$ )	1.0000	1.0000	2.3476	1.0363	1.0464
A4, A25 ( $\text{cm}^2$ )	2.5386	1.8655	1.7182	3.9147	2.7163
A5, A23 ( $\text{cm}^2$ )	1.3714	1.5962	1.2751	1.0025	1.0252
A6, A21 ( $\text{cm}^2$ )	1.3681	1.2642	1.4819	1.2167	1.5081
A7, A22 ( $\text{cm}^2$ )	2.4290	1.8254	4.6850	2.7146	2.3750
A8, A20 ( $\text{cm}^2$ )	1.6522	2.0009	1.1246	1.2663	1.4498
A9, A18 ( $\text{cm}^2$ )	1.8257	1.9526	2.1214	1.8006	1.4499
A10, A19 ( $\text{cm}^2$ )	2.3022	1.9705	3.8600	4.0274	2.5327
A11, A17 ( $\text{cm}^2$ )	1.3103	1.8294	2.9817	1.3364	1.2358
A12, A15 ( $\text{cm}^2$ )	1.4067	1.2358	1.2021	1.0548	1.3528
A13, A16 ( $\text{cm}^2$ )	2.1896	1.4049	1.2563	2.8116	2.9144
A14 ( $\text{cm}^2$ )	1.0000	1.0000	3.3276	1.1702	1.0085
Weight (kg)	366.50	368.84	377.20	362.84	360.40

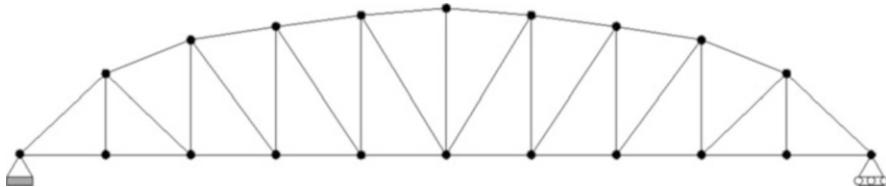
**Table 2.6** Natural frequencies (Hz) evaluated at the optimized designs for the planar 37-bar truss

Frequency number	Wang et al. [63]	Lingyun et al. [64]	Gomes [60]	Kaveh and Zolghadr [65] Standard CSS	Proposed algorithm
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

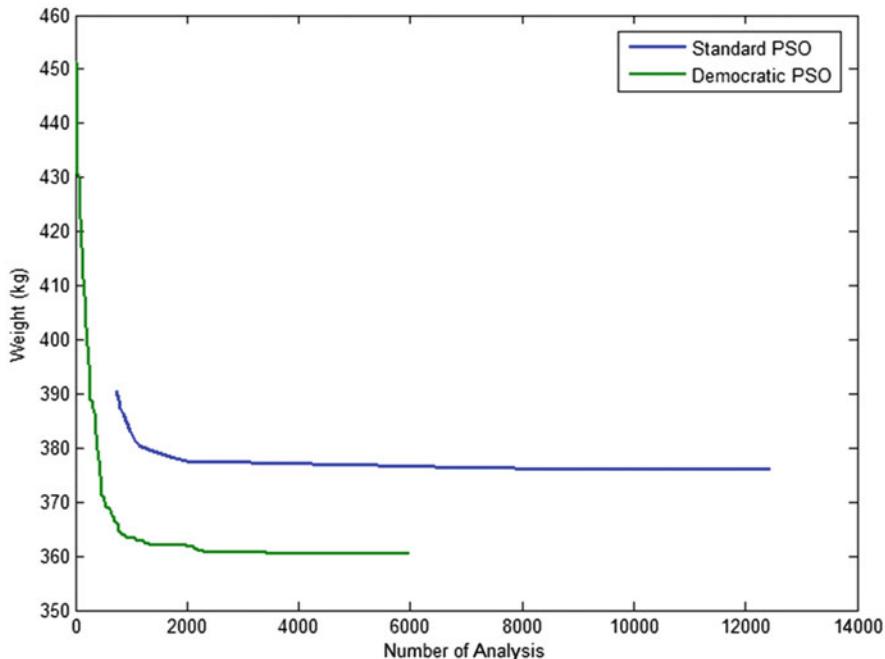
### 2.5.3.3 A 52-Bar Dome-Like Truss

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. Nonstructural 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.



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

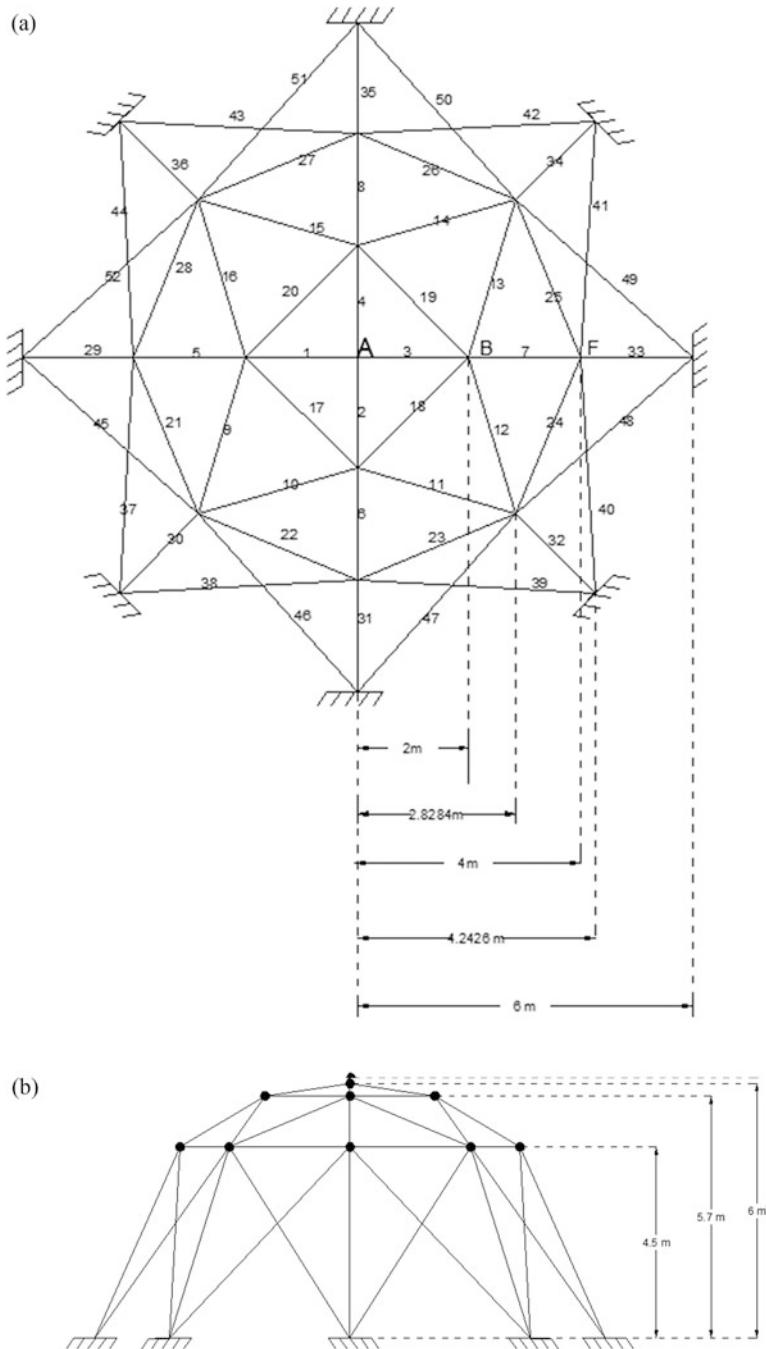


**Fig. 2.8** Comparison of convergence curves of democratic and standard PSO algorithms recorded in the 37-bar Pratt type planar truss

All of the elements of the structure are categorized in 8 groups according to Table 2.8. All free nodes are permitted to move  $\pm 2$  m from their initial position in a symmetrical manner. This is a configuration optimization problem with 13 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. The authors have studied the problem using the standard and an enhanced CSS [65] and a hybridized CSS–BBC 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.



**Fig. 2.9** Schematic of the initial layout of the spatial 52-bar truss. (a) Top view, (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 <sup>2</sup>	$2.1 \times 10^{11}$
$\rho$ (material density)/kg/m <sup>3</sup>	7800
Added mass/kg	50
Allowable range for cross sections/m <sup>2</sup>	$0.0001 \leq A \leq 0.001$
Constraints on the 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

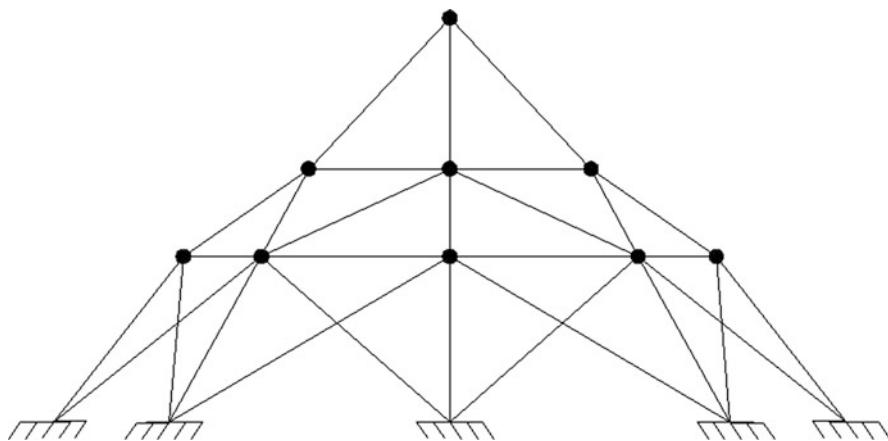
**Table 2.9** Optimized designs obtained for the spatial 52-bar truss problem

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

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 (6000 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

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

Frequency number	Lin et al. [67]	Lingyun et al. [64]	Gomes [60]	Kaveh and Zolghadr [65]	Present work
				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

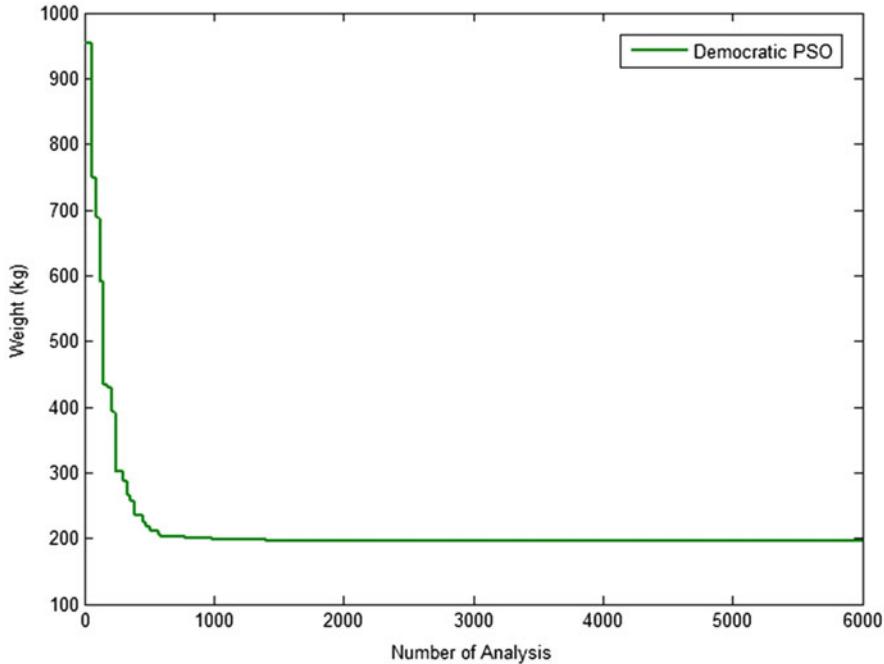
standard deviation of the results gained by DPSO are 198.71 kg and 13.85 kg, respectively, while PSO has obtained a mean weight of 234.3 kg and a standard deviation of 5.22 kg. DPSO performs considerably better in terms of best and mean weight.

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

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 Fig. 2.11. The convergence curve for the standard PSO is not available in reference [60].

#### 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.

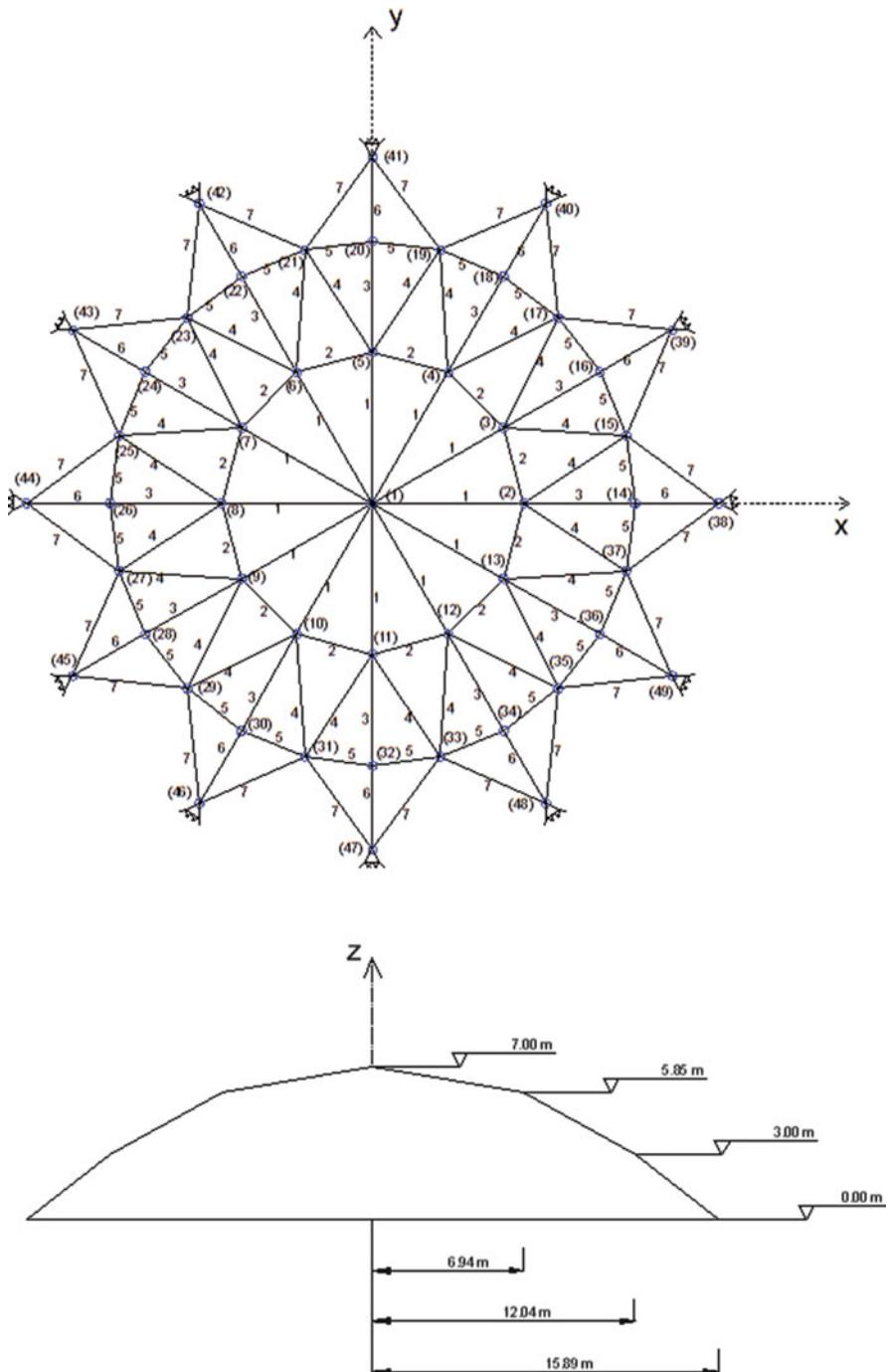


**Fig. 2.11** Convergence curve of the democratic PSO for the spatial 52-bar truss

The authors used the problem as a size optimization problem with frequency constraints in [65]. Nonstructural masses are attached to all free nodes as follows: 3000 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. Thirty particles and 200 iterations are used for both methods. Table 2.12 represents a comparison between the final results obtained by 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 8895.99 kg and 4.26 kg, respectively, while PSO has obtained a mean weight of 9251.84 kg and a standard deviation of 89.38 kg. This shows that the democratic PSO outperforms the standard version in all of the abovementioned aspects. Figure 2.13 shows the convergence curves for both methods.



**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

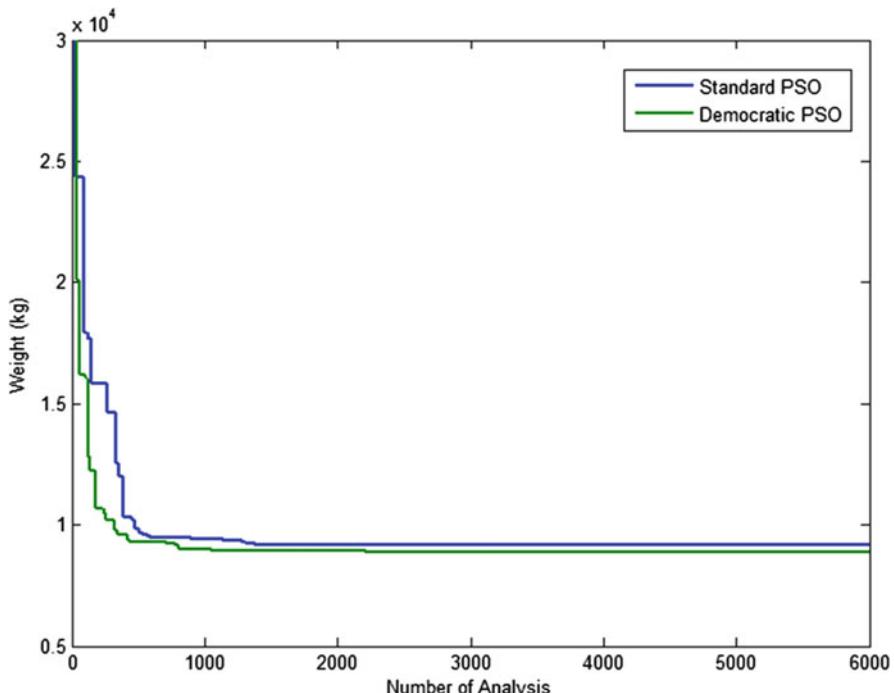
Property/unit	Value
E(Modulus of elasticity)/N/m <sup>2</sup>	$2.1 \times 10^{11}$
$\rho$ (Material density)/kg/m <sup>3</sup>	7971.810
Added mass/kg	$m_1 = 3000, m_1 = 500, m_2 = 100$
Allowable range for cross sections/m <sup>2</sup>	$0.0001 \leq A \leq 0.01293$
Constraints on first three frequencies/Hz	$\omega_1 \geq 9 \quad \omega_2 \geq 11$

**Table 2.12** Optimized designs (cm<sup>2</sup>) 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)	9171.93	8890.48

**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



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

## References

1. Kennedy J, Eberhart R (1995) Particle swarm optimization. Proc IEEE Int Conf Neural Netw 4:1942–1948
2. Shi Y, Eberhart R (1998) A modified particle swarm optimizer. In: Proceedings of IEEE World Congress on computational intelligence. 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. Reynolds 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. Santa Fe Institute, Addison-Wesley, USA
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. AP Professional, San Diego, CA, pp 212–226, Chapter 6
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 of 1999 ICEC, Washington, DC, pp 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, New York, pp 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: The 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. Inf Process 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: Third world congress on nature and biologically inspired computing (NaBIC 2011), IEEE, 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, Vesterstrom JS, Riget J (2002) Particle swarm optimization with spatial particle extension. In: Proceedings of the IEEE congress on evolutionary computation (CEC 2002), Honolulu, Hawaii
25. Riget J, Vesterstrom 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 13th Irish international conference on artificial intelligence and cognitive science 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)—Volume 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. In: Advances in self-organizing maps, Lecture notes in computer science 5629. pp 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) Confinements and biases in particle swarm optimisation. Open access archive HAL
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 Probab* 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 Biol Chem* 32:53–60
45. Løvbjerg M, Rasmussen TK, Krink T (2001) Hybrid particle swarm optimizer with breeding and subpopulations In: Proceedings of the genetic and evolutionary computation conference (GECCO-2001)
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) No 2439, pp 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, Indiana, 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 fifth 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 Probl Eng*, Article ID 607418, p 11
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 is presented, which is known as the charged system search (CSS) [1]. In this algorithm the governing Coulomb law from electrostatics and the governing laws of motion from the Newtonian 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 magnitude of the resultant force is determined by using the electrostatics laws, and the quality of the movement is determined using the governing laws of motion from the Newtonian mechanics. 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].

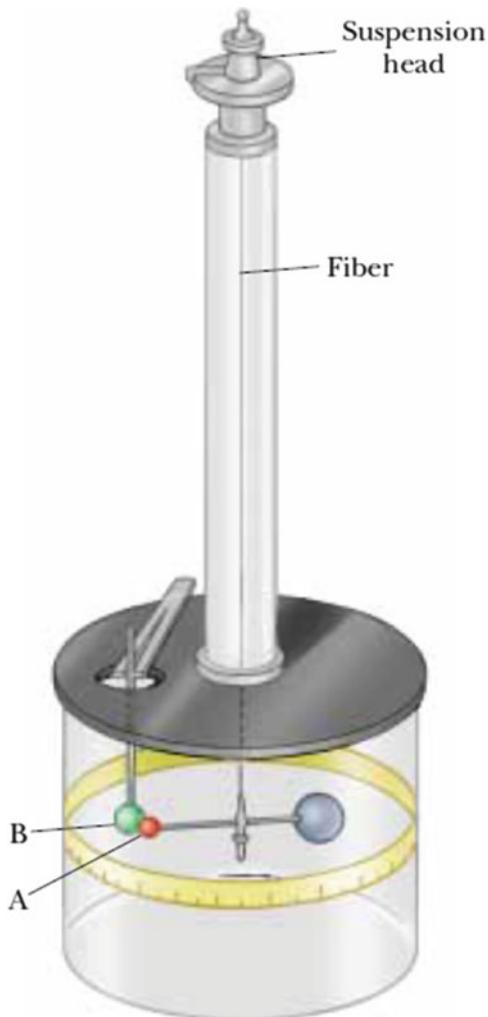
### 3.2 Charged System Search

#### 3.2.1 Background

##### 3.2.1.1 Electrical Laws

In physics, an electric charge creates an electric field in its surrounding space, 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

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



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 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:

$$F_{ij} = k_e \frac{q_i q_j}{r_{ij}^2} \quad (3.1)$$

where  $k_e$  is a constant called the Coulomb constant and  $r_{ij}$  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:

$$E_{ij} = k_e \frac{q_i}{r_{ij}^2} \quad (3.2)$$

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

$$E_{ij} = k_e \frac{q_i}{a^3} r_{ij} \quad (3.3)$$

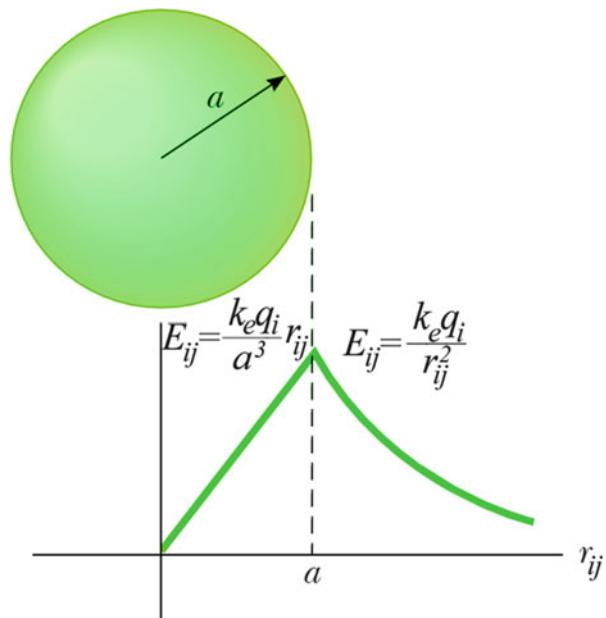
Note that this result shows that  $E_{ij} \rightarrow 0$  as  $r_{ij} \rightarrow 0$ . Therefore, the result eliminates the problem that would exist at  $r_{ij} = 0$  if  $E_{ij}$  is varied as  $1/r_{ij}^2$  inside the sphere as it does outside the sphere. That is, if  $E_{ij} \propto 1/r_{ij}^2$ , 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 points inside and outside the sphere coincide when  $r_{ij} = a$ . A plot of  $E_{ij}$  versus  $r_{ij}$  is shown in Fig. 3.2, Ref. [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:

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

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

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



$$E_{ij} = \begin{cases} \frac{k_e q_i}{a^3} r_{ij} & \text{if } r_{ij} < a \\ \frac{k_e q_i}{r_{ij}^2} & \text{if } r_{ij} \geq a \end{cases} \quad (3.5)$$

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:

$$\mathbf{F}_{ij} = E_{ij} q_j \frac{\mathbf{r}_i - \mathbf{r}_j}{\|\mathbf{r}_i - \mathbf{r}_j\|} \quad (3.6)$$

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

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

### 3.2.1.2 The Governing Laws of Motion from the Newtonian Mechanics

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 the change in its position. As a particle moves from an initial position  $\mathbf{r}_{old}$  to a final position  $\mathbf{r}_{new}$ , its displacement is given by:

$$\Delta\mathbf{r} = \mathbf{r}_{new} - \mathbf{r}_{old} \quad (3.8)$$

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

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

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:

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

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

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

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:

$$\mathbf{F} = m \cdot \mathbf{a} \quad (3.12)$$

where  $m$  is the mass of the object.

Substituting Eq. (3.12) in Eq. (3.11), we have

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

### 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 electric fields of the other agents. The magnitude 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 governing laws of motion from the Newtonian mechanics. 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,  $fit(i)$ . In order to introduce CSS, the following rules are developed:

**Rule 1** Many of the natural evolution algorithms maintain a population of solutions which evolve through random alterations and selection [4, 5]. Similarly, CSS considers a number of charged particles (CPs). Each CP has a magnitude of charge ( $q_i$ ) and as a result creates an electric field in its surrounding space. The magnitude of the charge is defined considering the quality of its solution as follows:

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

where  $fitbest$  and  $fitworst$  are so far the 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:

$$r_{ij} = \frac{\|\mathbf{X}_i - \mathbf{X}_j\|}{\|( \mathbf{X}_i + \mathbf{X}_j ) / 2 - \mathbf{X}_{best}\| + \epsilon} \quad (3.15)$$

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

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

$$x_{i,j}^{(0)} = x_{i,\min} + rand \cdot (x_{i,\max} - x_{i,\min}), \quad i = 1, 2, \dots, n \quad (3.16)$$

where  $x_{i,j}^{(0)}$  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:

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

**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 the other. In other words, a good CP attracts a bad CP:

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

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

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

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 solutions can be considered. Therefore, it seems that the third of the above conditions is the best rule because of providing strong exploration ability and an efficient exploitation.

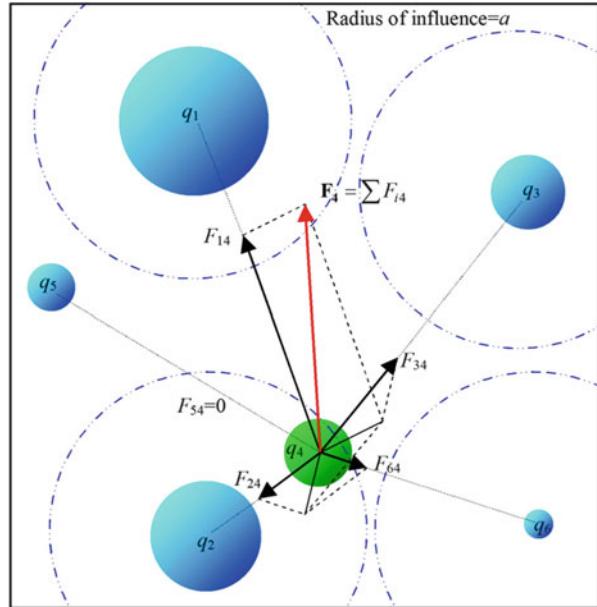
**Rule 4** The value of the resultant electrical force acting on a CP is determined using Eq. (3.7) as:

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

where  $\mathbf{F}_j$  is the resultant force acting on the  $j$ th 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. Here, 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:

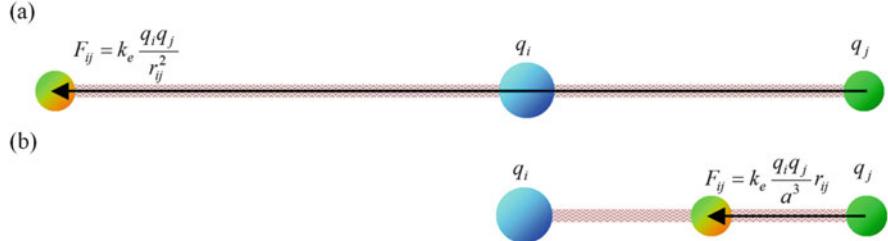
**Fig. 3.3** Determining the resultant electrical force acting on a CP [1]



$$a = 0.10 \times \max(\{x_{i,\max} - x_{i,\min} | i = 1, 2, \dots, n\}) \quad (3.21)$$

According to this rule, in the first iterations 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 distance 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 distance 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 ( $F_{ij} \propto r_{ij}$ ) guarantees 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



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

fitness (great  $q_i$ ) can create a bigger attracting 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 Eqs. (3.9) and (3.13) as follows:

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

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

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  $q_j$ .  $\Delta t$  is the time step and is set to unity.

The effect of the previous 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 decreased 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 previous 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:

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

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 increases. In this way, the balance between the

exploration and exploitation is saved. Considering the values of these parameters, Eqs. (3.22) and (3.23) can be rewritten as:

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

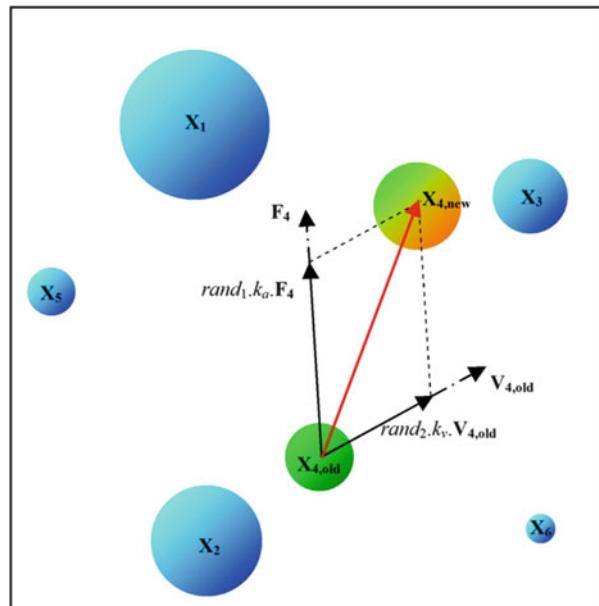
$$\mathbf{V}_{j,new} = \mathbf{X}_{j,new} - \mathbf{X}_{j,old} \quad (3.26)$$

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's 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 chapter, 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 Eq. (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

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



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:

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

where “w.p.” is the abbreviation for “with the probability”;  $x_{i,j}$  is the  $i$ th component of the CP  $j$ ; the charged memory considering rate (CMCR) 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  $(1 - \text{CMCR})$  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  $(1 - \text{PAR})$  sets the rate of doing nothing. For more details, the reader may refer to Refs. [4, 6].

**Rule 8** The terminating criterion is one of the following:

- Maximum number of iterations: The optimization process is terminated after a fixed number of iterations, for example, 1000 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 prefixed 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 prefixed 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 force 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 unimodal and multimodal functions. These numerical examples are presented in Sect. 3.1. The performance of the CSS to optimize these functions is investigated in Sect. 3.2. In Sect. 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.

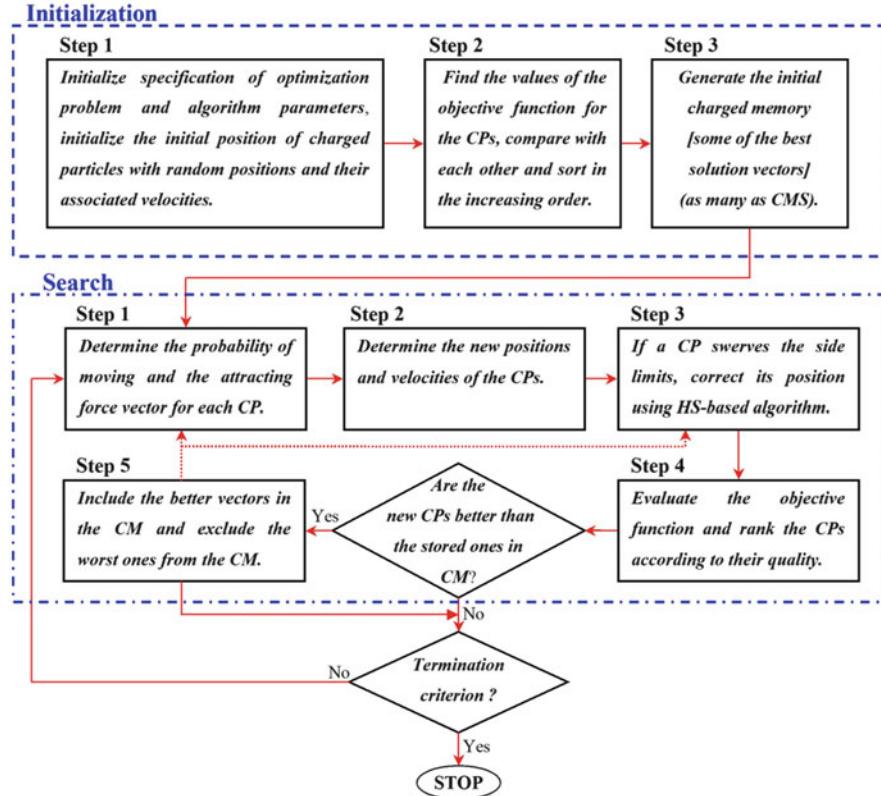


Fig. 3.6 The flowchart of the CSS [1]

### 3.3.1 Description of the Examples

In this section, a number of benchmark functions chosen from Ref. [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.

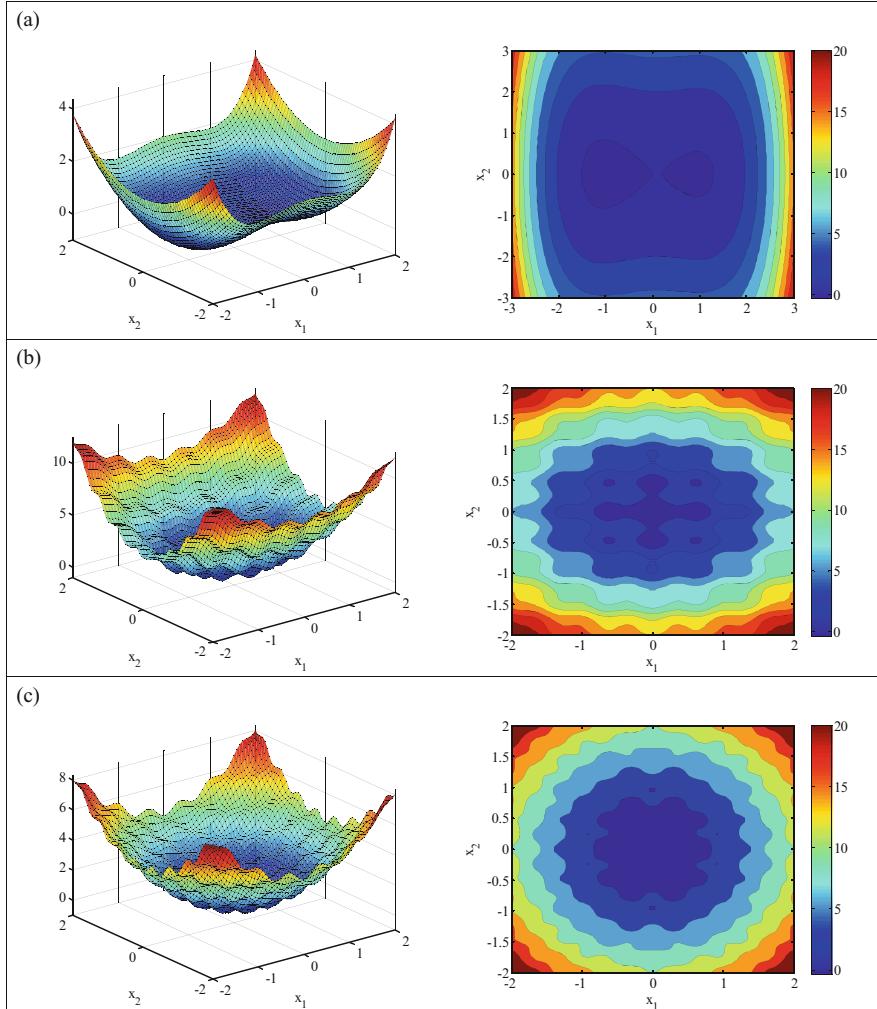
### 3.3.2 Results

Similar to the other metaheuristics, 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

**Table 3.1** Specifications of the benchmark problems

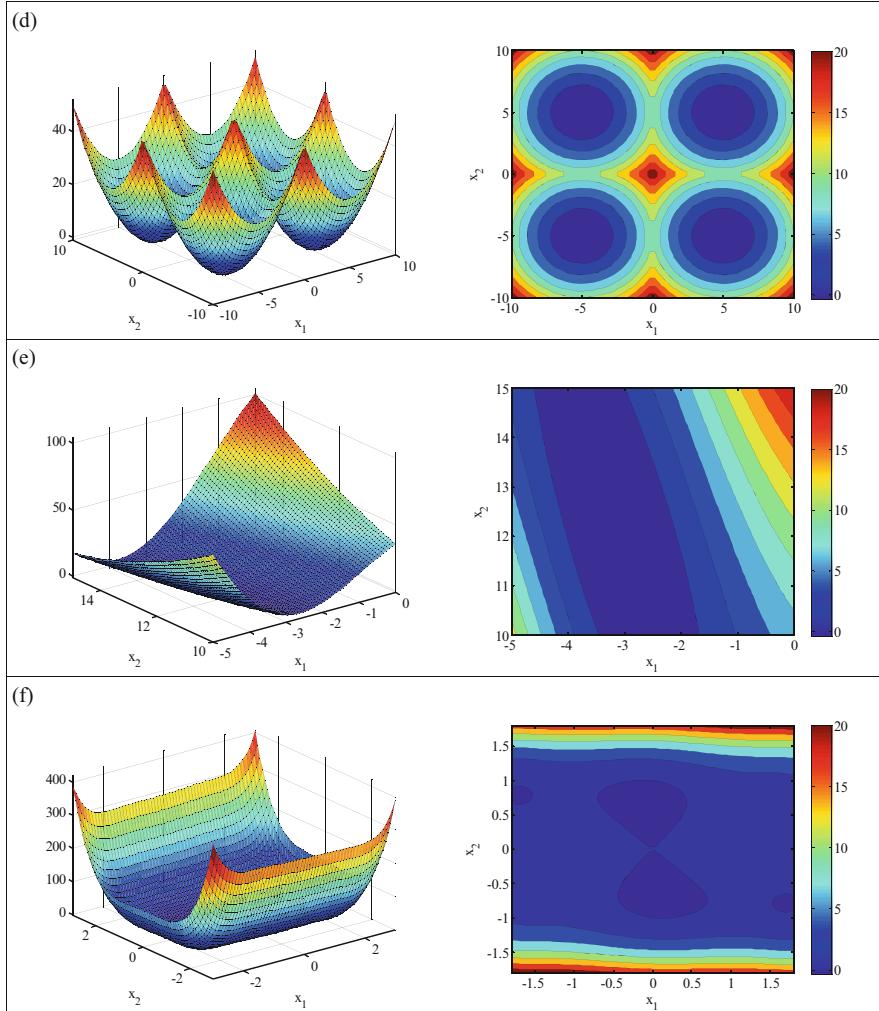
Function name	Interval	Function	Global minimum
Aluffi-Pentini	$\mathbf{X} \in [-10, 10]^2$	$f(\mathbf{X}) = \frac{1}{4}x_1^4 - \frac{1}{2}x_1^2 + \frac{1}{10}x_1 + \frac{1}{2}x_2^2$	-0.352386
Bohachevsky 1	$\mathbf{X} \in [-100, 100]^2$	$f(\mathbf{X}) = x_1^2 + 2x_2^2 - \frac{3}{10} \cos(3\pi x_1) - \frac{4}{10} \cos(4\pi x_2) + \frac{7}{10}$	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 Lagoo	$\mathbf{X} \in [-10, 10]^2$	$f(\mathbf{X}) = ( x_1  - 5)^2 + ( x_2  - 5)^2$	0.0
Brainin	$0 \leq x_2 \leq 15 - 5 \leq x_1 \leq 10$	$f(\mathbf{X}) = (x_2 - \frac{5}{4}x_1^2 + \frac{5}{4}x_1)^2 + 10(1 - \frac{1}{8\pi}) \cos(x_1) + 10$	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
DeJong	$\mathbf{X} \in [-5.12, 5.12]^3$	$f(\mathbf{X}) = x_1^2 + x_2^2 + x_3^2$	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	$\mathbf{X} \in [-2, 2]^2$	$f(\mathbf{X}) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] \\ \times [30 + (2x_1 - 3x_2)^2((18 - 32x_1 - 12x_1^2 + 48x_2^2 - 36x_1x_2 + 27x_2^2)^2]$	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	$\mathbf{X} \in [0, 1]^3$	$f(\mathbf{X}) = -\sum_{i=1}^4 c_i \exp \left( -\sum_{j=1}^3 a_{ij} (x_j - p_{ij})^2 \right)$ $a = \begin{bmatrix} 3 & 10 & 30 \\ 0.1 & 10 & 35 \\ 3 & 10 & 30 \\ 0.1 & 10 & 35 \end{bmatrix}, c = \begin{bmatrix} 1 \\ 1.2 \\ 3 \\ 3.2 \end{bmatrix} \text{ and}$ $p = \begin{bmatrix} 0.3689 & 0.117 & 0.2673 \\ 0.4699 & 0.4387 & 0.747 \\ 0.1091 & 0.8732 & 0.5547 \\ 0.03815 & 0.5743 & 0.8828 \end{bmatrix}.$	-3.862782
Hartman 6	$\mathbf{X} \in [0, 1]^6$	$f(\mathbf{X}) = -\sum_{i=1}^4 c_i \exp \left( -\sum_{j=1}^6 a_{ij} (x_j - p_{ij})^2 \right)$ $a = \begin{bmatrix} 10 & 3 & 17 & 3.5 & 1.7 & 8 \\ 0.05 & 10 & 17 & 0.1 & 8 & 14 \\ 3 & 3.5 & 17 & 10 & 17 & 8 \\ 17 & 8 & 0.05 & 10 & 0.1 & 14 \end{bmatrix}, c = \begin{bmatrix} 1 \\ 1.2 \\ 3 \\ 3.2 \end{bmatrix} \text{ and}$ $p = \begin{bmatrix} 0.1312 & 0.1696 & 0.5569 & 0.0124 & 0.8283 & 0.5886 \\ 0.2329 & 0.4135 & 0.8307 & 0.3736 & 0.1004 & 0.9991 \\ 0.2348 & 0.1451 & 0.3522 & 0.2883 & 0.3047 & 0.6650 \\ 0.4047 & 0.8828 & 0.8732 & 0.5743 & 0.1091 & 0.0381 \end{bmatrix}$	-3.322368
Rastrigin	$\mathbf{X} \in [-1, 1]^2$	$f(\mathbf{X}) = \sum_{i=1}^2 (x_i^2 - \cos(18x_i))$	-2.0
Rosenbrock	$\mathbf{X} \in [-30, 30]^n, n = 2$	$f(\mathbf{X}) = \sum_{i=1}^{n-1} 100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2$	0.0

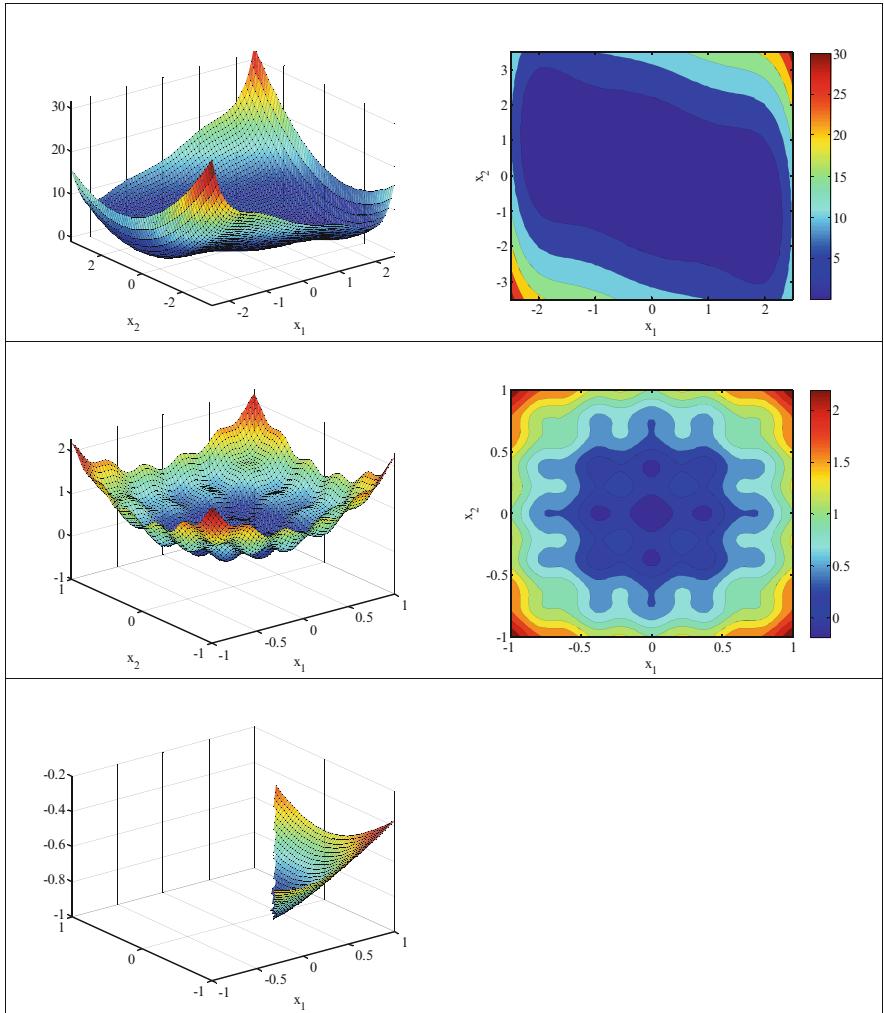


**Fig. 3.7** A perspective view and the related contour lines for some of function when  $n = 2$ , [1]. (a) Aluffi-Pentini, (b) Bohachevsky 1, (c) Bohachevsky 2, (d) Becker and Lago, (e) Branin, (f) Camel, (g) Cb3, (h) Cosine mixture, (i) Exponential, (j) Griewank, (k) Rastrigin, (l) Rosenbrock

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

**Fig. 3.7 (continued)**

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 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, and 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 favorable region 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 metaheuristic 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

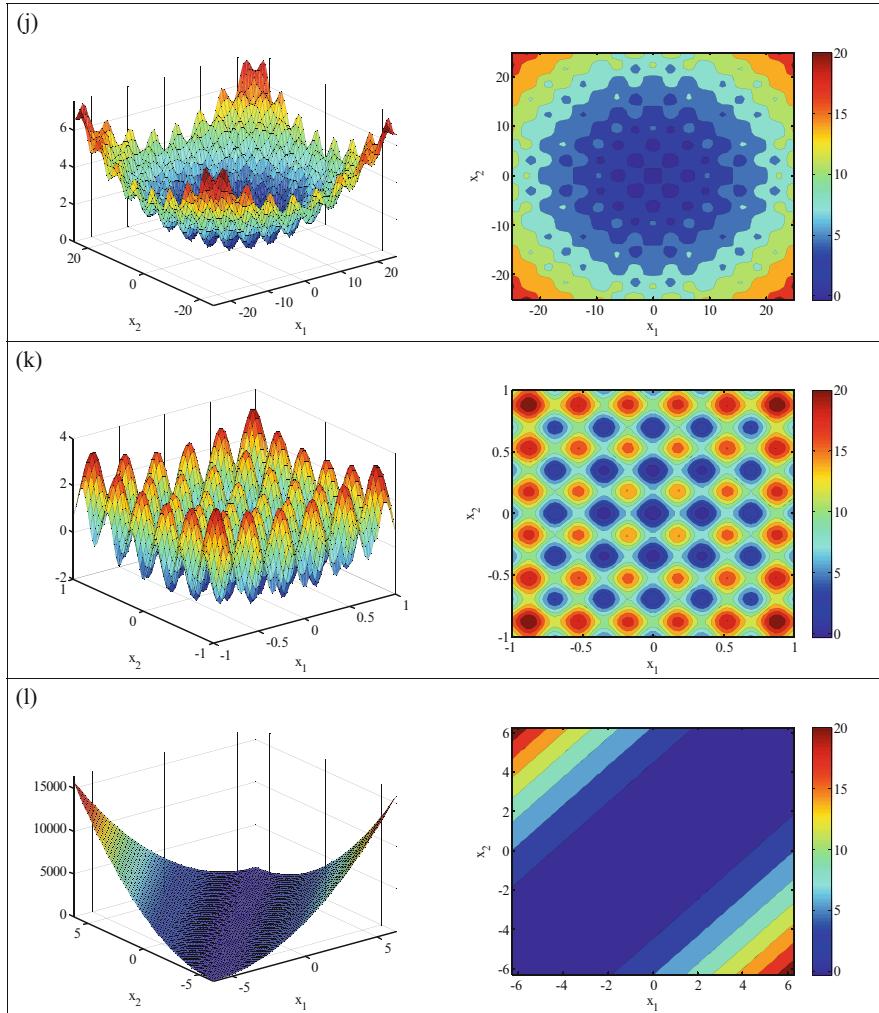


Fig. 3.7 (continued)

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 exploitation, 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.

**Table 3.2** Performance comparison for the benchmark problems

Function	GEN	GEN-S	GEN-S-M	GEN-S-M-LS	CSS
AP	1360 (0.99)	1360	1277	1253	804
Bf1	3992	3356	1640	1615	1187
Bf2	20,234	3373	1676	1636	742
BL	19,596	2412	2439	1436	423
Branin	1442	1418	1404	1257	852
Camel	1358	1358	1336	1300	575
Cb3	9771	2045	1163	1118	436
CM	2105	2105	1743	1539	1563
Dejoung	9900	3040	1462	1281	630
Exp2	938	936	817	807	132
Exp4	3237	3237	2054	1496	867
Exp8	3237	3237	2054	1496	1426
Goldstein and Price	1478	1478	1408	1325	682
Griewank	18,838 (0.91)	3111 (0.91)	1764	1652 (0.99)	1551
Hartman3	1350	1350	1332	1274	860
Hartman6	2562 (0.54)	2562 (0.54)	2530 (0.67)	1865 (0.68)	1783
Rastrigin	1533 (0.97)	1523 (0.97)	1392	1381	1402
Rosenbrock	9380	3739	1675	1462	1452
Total	112,311 (96.72)	41,640 (96.77)	29,166 (98.16)	25,193 (98.16)	17,367

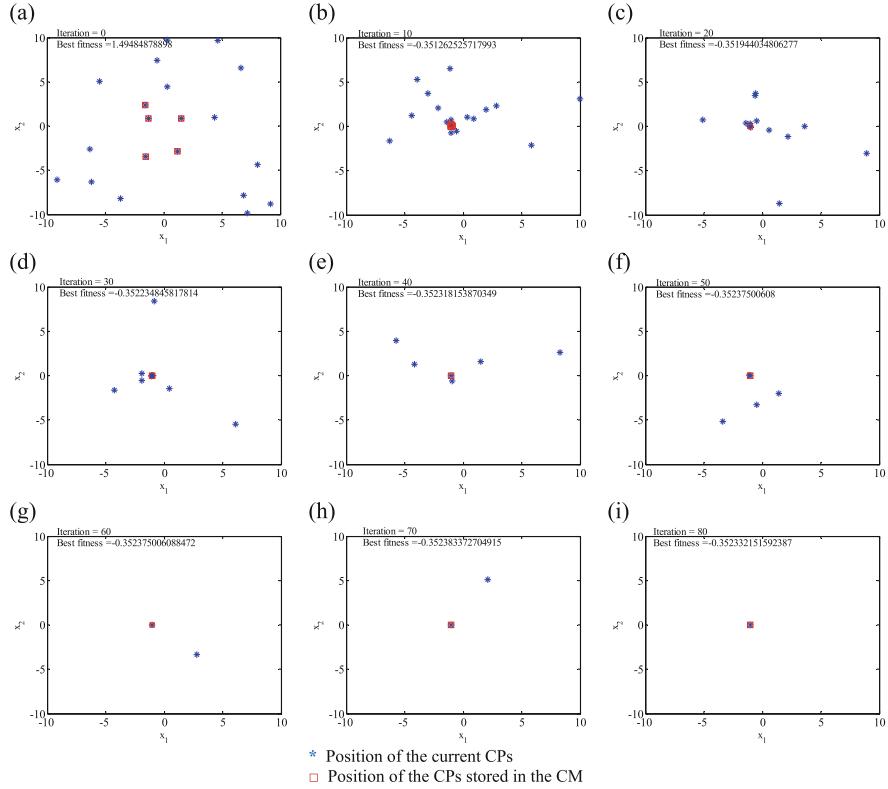
## 3.4 Charged System Search for Structural Optimization

### 3.4.1 Statement of the Optimization Design Problem

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

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

where  $W(\mathbf{X})$  is the weight of the structure;  $n$  is the number of members making up the structure;  $\rho_i$  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]).



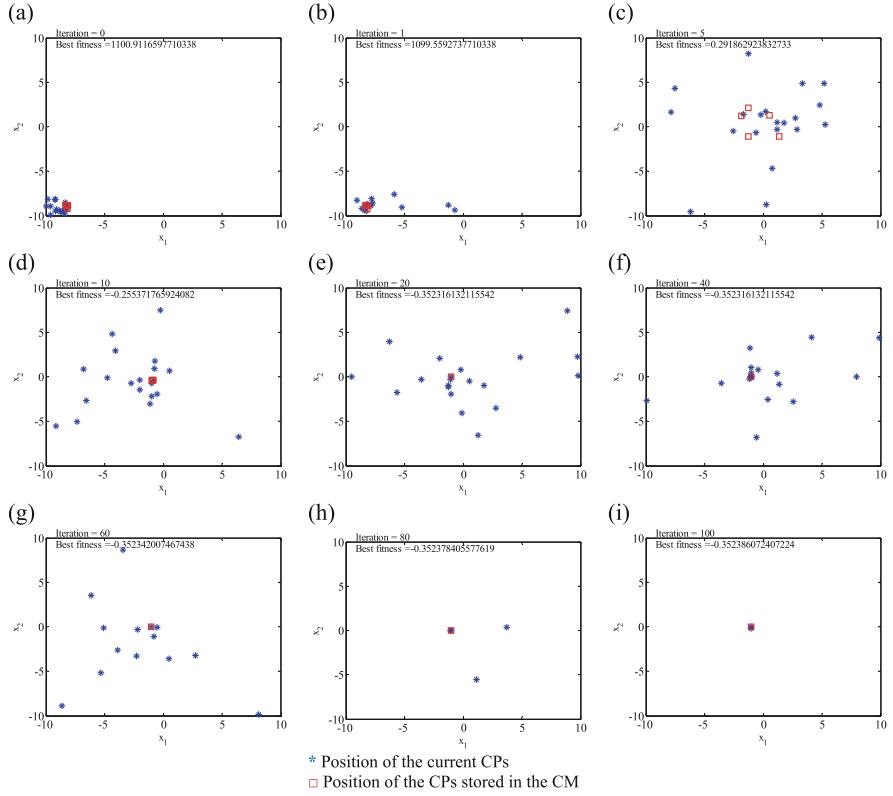
**Fig. 3.8** The positions of the current CPs and the stored CPs in the CM for the first example [1]. Asterisk Position of the current CPs. Square Position of the CPs stored in the CM

### 3.4.1.1 Constraint Conditions for Truss Structures

For truss structures, the constraints are as follows:

$$\begin{aligned} \delta_{\min} &\leq \delta_i \leq \delta_{\max} & i = 1, 2, \dots, m \\ \sigma_{\min} &\leq \sigma_i \leq \sigma_{\max} & i = 1, 2, \dots, n \\ \sigma_i^b &\leq \sigma_i \leq 0 & i = 1, 2, \dots, nc \end{aligned} \quad (3.29)$$

in which  $m$  is the number of nodes;  $nc$  denotes the number of compression elements;  $\sigma_i$  and  $\delta_i$  are the stress and nodal displacement, respectively; and  $\sigma_i^b$  represents allowable buckling stress in member  $i$  when it is in compression.



**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]. Asterisk Position of the current CPs. Square Position of the CPs stored in the CM

### 3.4.1.2 Constraint Conditions for Frame Structures

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

The stress limitations:

$$\frac{f_a}{F_a} + \frac{f_{bx}}{F_{bx}} + \frac{f_{by}}{F_{by}} \leq 1, \quad \text{For } \frac{f_a}{F_a} \leq 0.15 \quad (3.30)$$

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

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

The slenderness ratio limitation:

$$\begin{cases} \lambda_i = \frac{k_i L_i}{r_i} \leq 300 & \text{For tension members} \\ \lambda_i = \frac{k_i L_i}{r_i} \leq 200 & \text{For compression members} \end{cases} \quad (3.33)$$

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  $\psi$  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 buckling failure mode of the member according to the slenderness ratio:

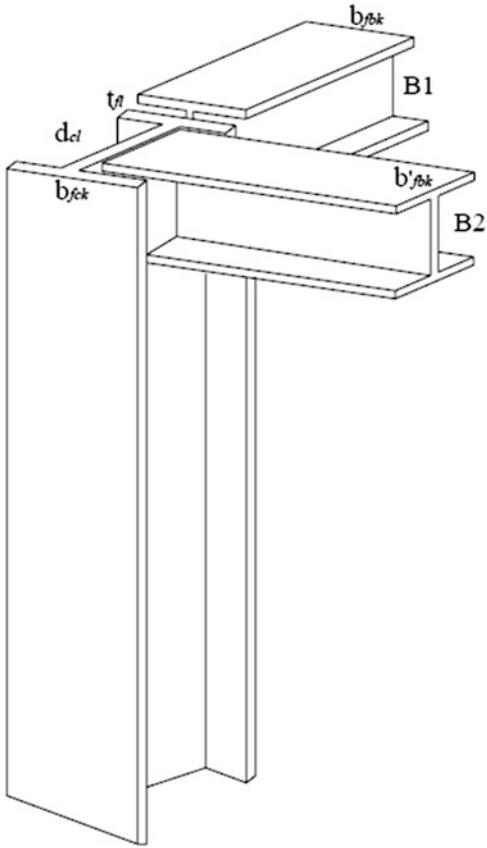
$$F_a = \begin{cases} \left[ \left( 1 - \frac{\lambda_i^2}{2C_C^2} \right) F_y \right] / \left( \frac{5}{3} + \frac{3\lambda_i}{8C_C} - \frac{\lambda_i^3}{8C_C^3} \right) & \text{For } \lambda_i < C_C \\ \frac{12\pi^2 E}{23\lambda_i^2} & \text{For } \lambda_i \geq C_C \end{cases} \quad (3.34)$$

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 ( $C_C = \sqrt{2\pi^2 E/F_y}$ );  $\lambda_i$  = 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 (3.23) 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]):

**Fig 3.10** Beam-column geometric constraints [2]



$$b_{fb} \leq b_{fc} \quad (3.35)$$

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

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, Eq. (3.36) guarantees that flange width of the beam B2 remains smaller than clear distance between the flanges of the column.

Maximum lateral displacement:

$$\frac{\Delta_T}{H} \leq R \quad (3.37)$$

Inter-story displacement constraints:

$$\frac{d_i}{h_i} \leq R_I, \quad i = 1, 2, \dots, ns \quad (3.38)$$

where  $\Delta_T$  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  $i$ th floor,  $ns$  represents the total number of stories, and  $R_I$  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  $2.88 \text{ kN/m}^2$  and a design live load of  $2.39 \text{ kN/m}^2$ . The roof is subjected to a design dead load of  $2.88 \text{ kN/m}^2$  plus snow load. The design snow load is computed using Equation (7-1) in ASCE 7-05 [11], resulting in a design snow pressure of  $0.75 \text{ kN/m}^2$ . 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:

$$p_w = (0.613K_zK_{zt}K_dV^2I)(GC_p) \quad (3.39)$$

where  $p_w$  is the design wind pressure in  $\text{kN/m}^2$ ;  $K_z$  ( $=1.07$ ) 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;  $V$  ( $=46.94 \text{ m/s}$ ) 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 ASD-AISC specification is considered as:

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

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. 3.3, 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  $N$  Charged Particles (CPs) with random positions are generated considering the variable bounds together with their associated velocities. The structures associated with the generated CPs are analyzed and the fitness functions values of the CPs are evaluated considering the weight of the structure and the penalty functions. Then, CPs are ranked in the increasing order of their fitness function values.  $CMS$  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 [Eq. (3.24)], the magnitude of the attracting force vector [Eq. (3.25)], and the motion laws [Eqs. (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 evaluate the fitness function values of their corresponding CPs 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 metaheuristic 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

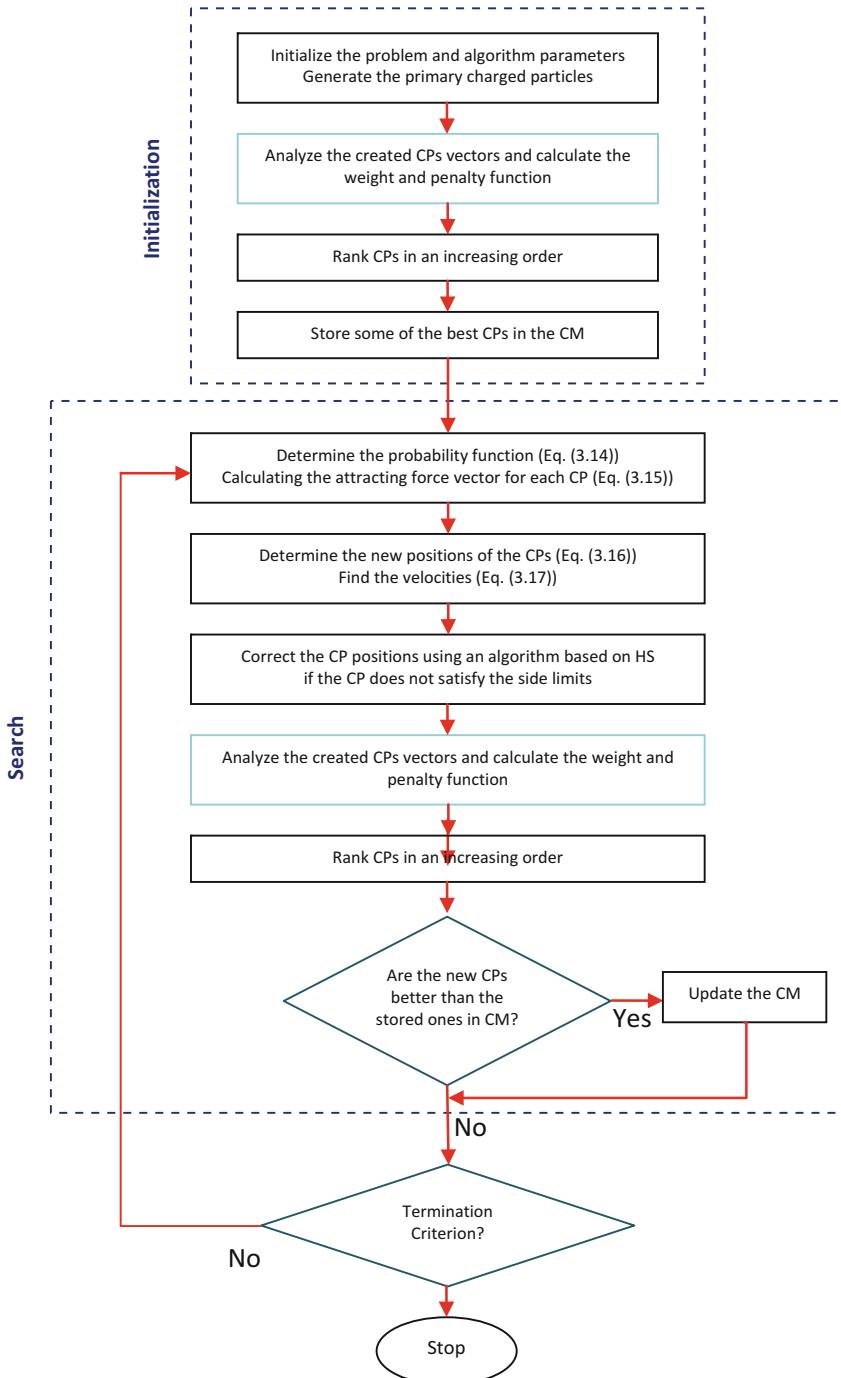


Fig. 3.11 The flowchart of the CSS for the truss structures [2]

of the previous velocity and the resultant force affecting a CP can be decreased or increased based on the values of the  $k_v$  and  $k_a$ . Here,  $k_v$  and  $k_a$  are defined as:

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

where  $iter$  is the iteration number,  $iter_{\max}$  is the maximum number of the iterations, and  $c$  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 increases. In this way, the balance between the exploration and the exploitation 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 is based on 20 evaluations. The algorithms are coded in MATLAB and structures are analyzed using the direct stiffness method.

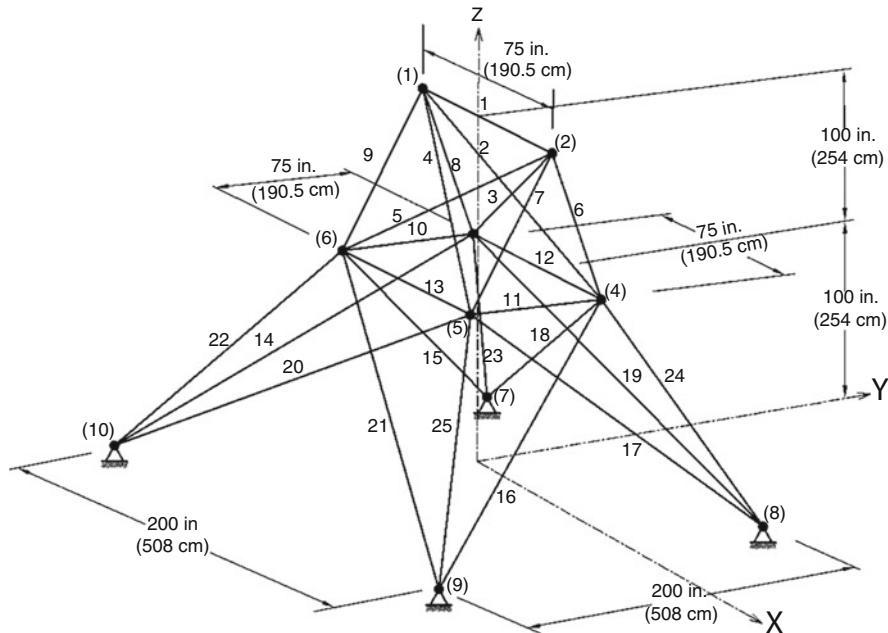
### 3.5.1 A Benchmark Truss

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<sup>3</sup> (2767.990 kg/m<sup>3</sup>), and the modulus of elasticity is taken as 10,000 ksi (68,950 MPa). The twenty-five members are categorized into eight groups as follows:

- (1) A<sub>1</sub>, (2) A<sub>2</sub>–A<sub>5</sub>, (3) A<sub>6</sub>–A<sub>9</sub>, (4) A<sub>10</sub>–A<sub>11</sub>, (5) A<sub>12</sub>–A<sub>13</sub>, (6) A<sub>14</sub>–A<sub>17</sub>, (7) A<sub>18</sub>–A<sub>21</sub>, and (8) A<sub>22</sub>–A<sub>25</sub>

This spatial truss is subjected to two loading conditions shown in Table 3.3. Maximum displacement limitations of  $\pm 0.35$  in ( $\pm 8.89$  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<sup>2</sup> (0.6452–21.94 cm<sup>2</sup>).

The CSS algorithm achieves the best solution after 7000 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 9875 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 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 metaheuristic algorithms.



**Fig. 3.12** Schematic of a 25-bar spatial truss [2]

**Table 3.3** Loading conditions for the 25-bar spatial truss

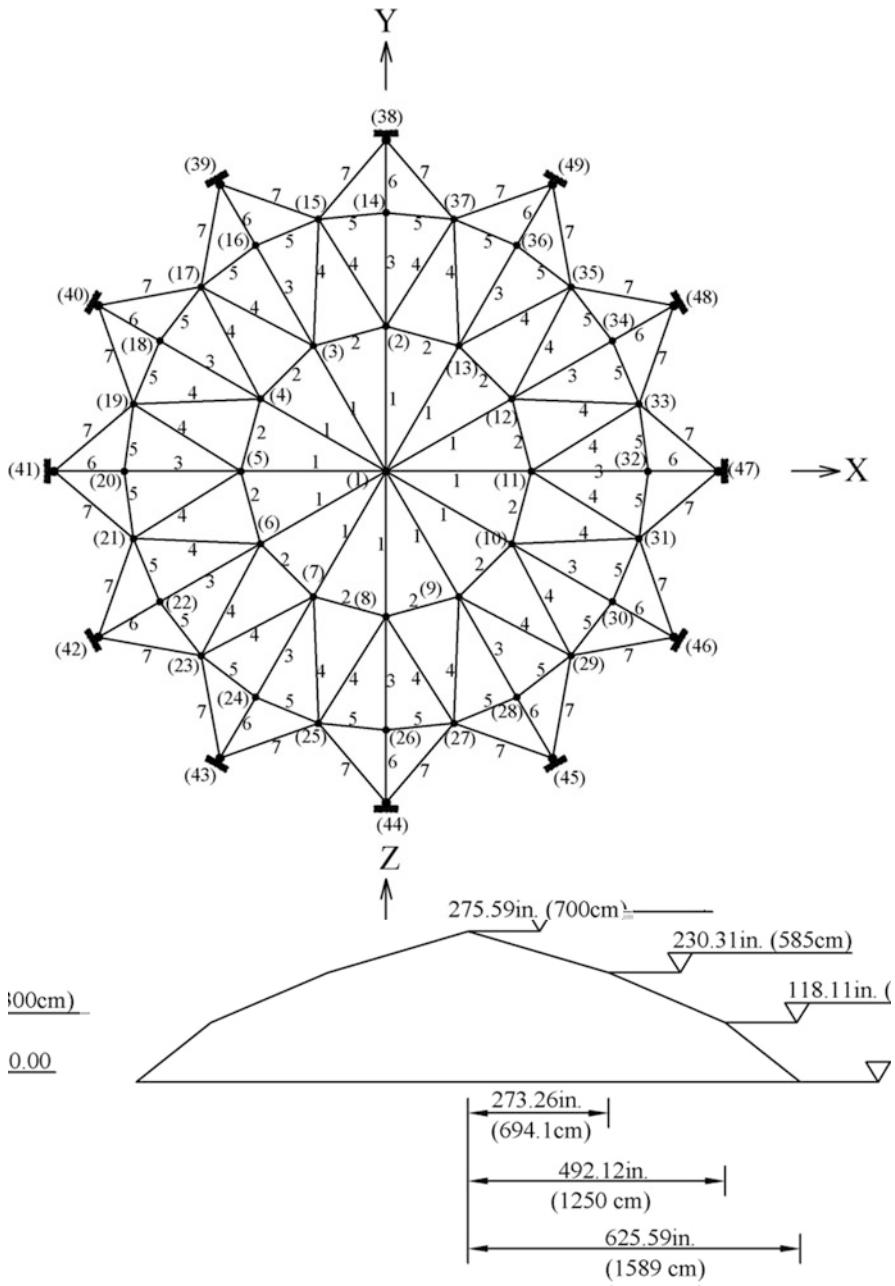
Node	Case 1			Case 2		
	P <sub>X</sub> kips (kN)	P <sub>Y</sub> kips (kN)	P <sub>Z</sub> kips (kN)	P <sub>X</sub> kips (kN)	P <sub>Y</sub> kips (kN)	P <sub>Z</sub> kips (kN)
1	0.0	20.0 (89)	-5.0 (22.25)	1.0 (4.45)	10.0 (44.5)	-5.0 (22.25)
2	0.0	-20.0 (89)	-5.0 (22.25)	0.0	10.0 (44.5)	-5.0 (22.25)
3	0.0	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 <sub>1</sub>	35.092 (241.96)	40.0 (275.80)
2	A <sub>2</sub> ~ A <sub>5</sub>	11.590 (79.913)	40.0 (275.80)
3	A <sub>6</sub> ~ A <sub>9</sub>	17.305 (119.31)	40.0 (275.80)
4	A <sub>10</sub> ~ A <sub>11</sub>	35.092 (241.96)	40.0 (275.80)
5	A <sub>12</sub> ~ A <sub>13</sub>	35.092 (241.96)	40.0 (275.80)
6	A <sub>14</sub> ~ A <sub>17</sub>	6.759 (46.603)	40.0 (275.80)
7	A <sub>18</sub> ~ A <sub>21</sub>	6.959 (47.982)	40.0 (275.80)
8	A <sub>22</sub> ~ A <sub>25</sub>	11.082 (76.410)	40.0 (275.80)

**Table 3.5** Performance comparison for the 25-bar spatial truss

Element group	Optimal cross-sectional areas ( $\text{in}^2$ )				Kaveh et al.	Kaveh and Talatieri [13]	PSACO [4]	HPSACO [4]	HB-B-BC [12]	$\text{in}^2$	$\text{cm}^2$	Present work [2]
	Rajeev and Krishnamoorthy	Schutte and Groenwold	Lee and Geem	PSO [15]								
1 A <sub>1</sub>	0.10	0.010	0.047	0.010	0.010	0.010	0.010	0.010	0.010	0.010	0.065	
2 A <sub>2</sub> ~A <sub>5</sub>	1.80	2.121	2.022	2.042	2.052	2.054	1.993	2.003	12.923			
3 A <sub>6</sub> ~A <sub>9</sub>	2.30	2.893	2.950	3.001	3.008	3.056	3.007	19.400				
4 A <sub>10</sub> ~A <sub>11</sub>	0.20	0.010	0.010	0.010	0.010	0.010	0.010	0.010	0.010	0.010	0.065	
5 A <sub>12</sub> ~A <sub>13</sub>	0.10	0.010	0.014	0.010	0.010	0.010	0.010	0.010	0.010	0.010	0.065	
6 A <sub>14</sub> ~A <sub>17</sub>	0.80	0.671	0.688	0.684	0.684	0.679	0.665	0.687	4.432			
7 A <sub>18</sub> ~A <sub>21</sub>	1.80	1.611	1.657	1.625	1.616	1.611	1.642	1.655	10.677			
8 A <sub>22</sub> ~A <sub>25</sub>	3.0	2.717	2.663	2.672	2.673	2.678	2.679	2.660	17.161			
Best weight (lb)	546	545.21	544.38	545.03	545.04	544.99	545.16	545.10	2424.7 N			
Average weight (lb)	N/A	546.84	N/A	545.74	N/A	545.52	545.66	545.58	2426.8 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	9596	15,000	3520	28,850	9875	12,500	7000				



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

### 3.5.2 A 120-Bar Dome Truss

The topology and group numbers of a 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<sup>3</sup> (7971.810 kg/m<sup>3</sup>). 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<sup>2</sup> (2 cm<sup>2</sup>), and the maximum cross-sectional area is taken as 20.0 in<sup>2</sup> (129.03 cm<sup>2</sup>). The constraints are considered as follows:

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

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

where  $\sigma_i^-$  is calculated considering the slenderness ratio [Eq. (3.34)].

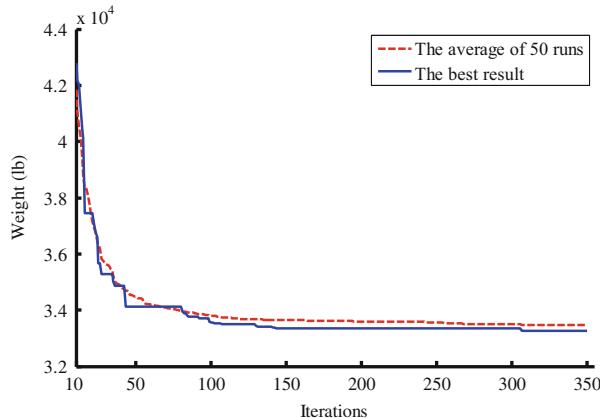
- 2) Displacement limitations of  $\pm 0.1969$  in ( $\pm 5$  mm) are imposed on all nodes in x, y, and z directions.

Table 3.6 illustrates the best solution vectors, the corresponding weights, and the required number of analyses for convergence in the present algorithm and some of other metaheuristic methods. Except IACS which uses two auxiliary mechanisms for searching, the CSS algorithm has the 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

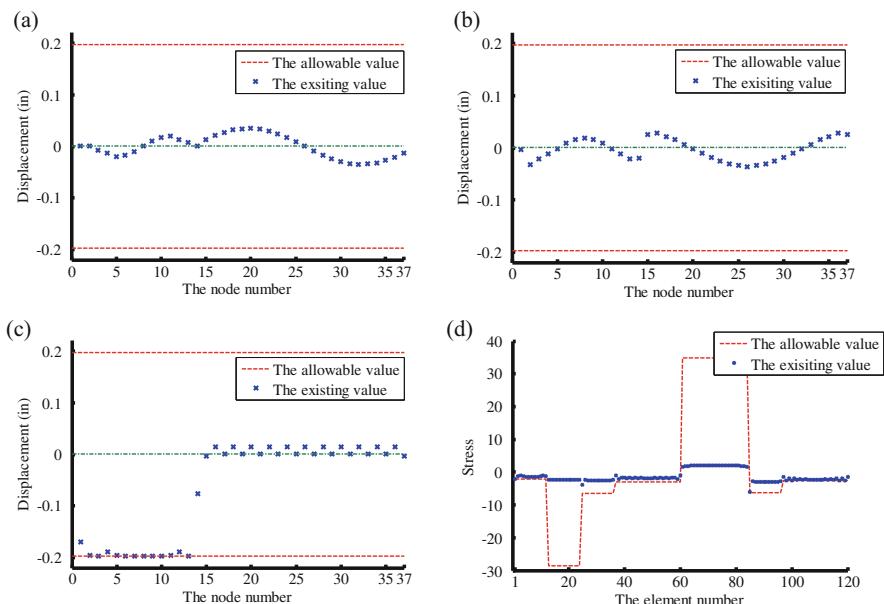
**Table 3.6** Performance comparison for the 120-bar dome truss

		Optimal cross-sectional areas (in <sup>2</sup> )					
Element group	Kaveh et al.	Kaveh and Talatahari				Present work [2]	
		IACS [13]	PSOPC [4]	PSACO [4]	HPSACO [4]	HBB-BC [12]	in <sup>2</sup>
1	A <sub>1</sub>	3.026	3.040	3.026	3.095	3.037	3.027
2	A <sub>2</sub>	15.06	13.149	15.222	14.405	14.431	14.606
3	A <sub>3</sub>	4.707	5.646	4.904	5.020	5.130	5.044
4	A <sub>4</sub>	3.100	3.143	3.123	3.352	3.134	3.139
5	A <sub>5</sub>	8.513	8.759	8.341	8.631	8.591	8.543
6	A <sub>6</sub>	3.694	3.758	3.418	3.432	3.377	3.367
7	A <sub>7</sub>	2.503	2.502	2.498	2.499	2.500	2.497
Best weight (lb)		33,320.52	33,481.2	33,263.9	33,248.9	33,287.9	33,251.9
No. of analyses		3250	150,000	32,600	10,000	10,000	7000

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 %.

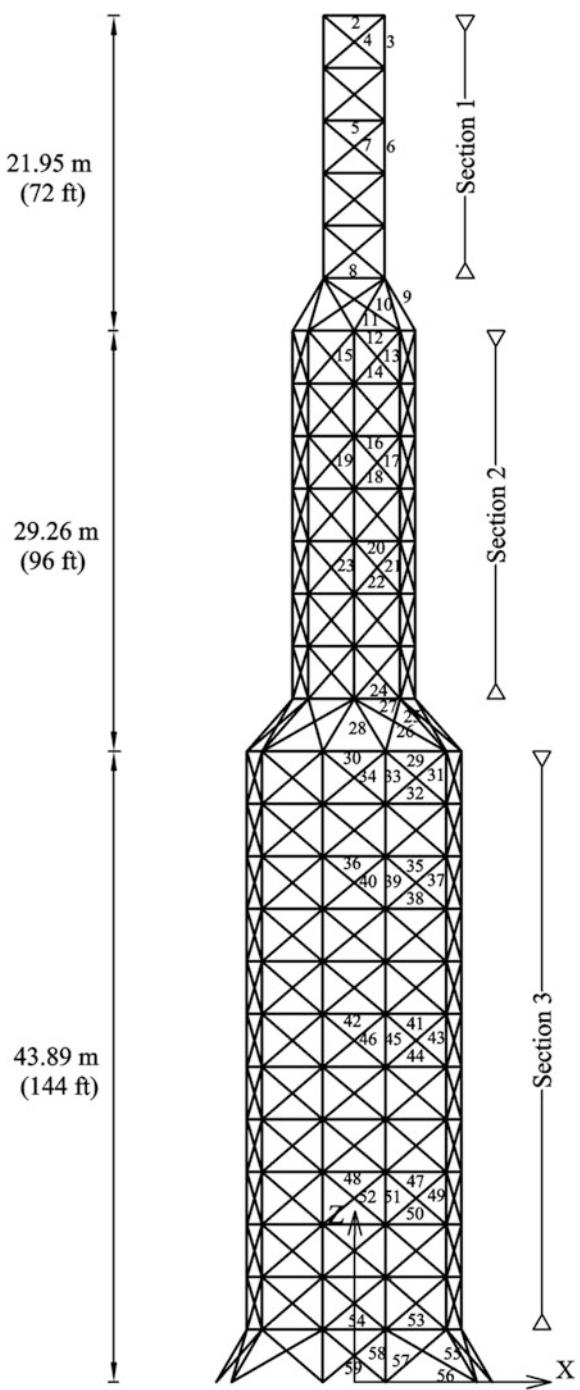


**Fig. 3.14** Convergence history of the 120-bar dome-shaped truss for the CSS algorithm [2]



**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. (d) Stress

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



### 3.5.3 A 26-Story-Tower Space Truss

The 26-story-tower space truss containing 942 elements and 244 nodes is considered as a 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  $0.1 \text{ lb/in}^3$  ( $2767.990 \text{ kg/m}^3$ ) and the modulus of elasticity is 10,000 ksi (68,950 MPa).

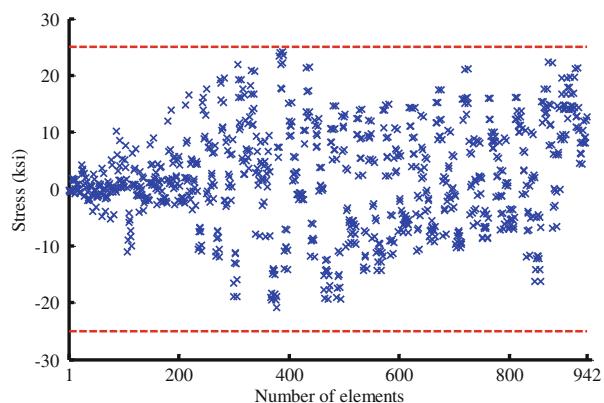
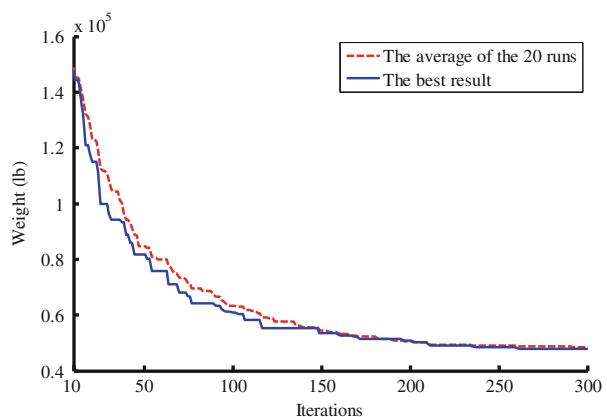
The members are subjected to the stress limits of  $\pm 25$  ksi (172.375 MPa), and the four nodes of the top level in the x, y, and z directions are subjected to the displacement limits of  $\pm 15.0$  in (38.10 cm) (about 1/250 of the total height of the tower). The allowable cross-sectional areas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

- 1) The vertical load at each node in the first section is equal to  $-3$  kips ( $-13.344 \text{ kN}$ ).
- 2) The vertical load at each node in the second section is equal to  $-6$  kips ( $-26.688 \text{ kN}$ ).
- 3) The vertical load at each node in the third section is equal to  $-9$  kips ( $-40.032 \text{ kN}$ ).
- 4) The horizontal load at each node on the right side in the x direction is equal to  $-1$  kips ( $-4.448 \text{ kN}$ ).
- 5) The horizontal load at each node on the left side in the x direction is equal to  $1.5$  kips ( $6.672 \text{ kN}$ ).
- 6) The horizontal load at each node on the front side in the y direction is equal to  $-1$  kips ( $-4.448 \text{ kN}$ ).
- 7) The horizontal load at each node on the back side in the x direction is equal to  $1$  kips ( $4.448 \text{ kN}$ ).

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 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.

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

	Kaveh and Talatahari [12]				Present work [2]
	GA	PSO	BB–BC	HBB–BC	
Best weight (lb)	56,343 (250,626 N)	60,385 (268,606 N)	53,201 (236,650 N)	52,401 (233,091 N)	47,371 (210,716 N)
Average weight (lb)	63,223 (281,230 N)	75,242 (334,693 N)	55,206 (245,568 N)	53,532 (238,122 N)	48,603 (216,197 N)
Std dev (lb)	6640.6 (29,539 N)	9906.6 (44,067 N)	2621.3 (11,660 N)	1420.5 (6318 N)	950.4 (4227 N)
No. of analyses	50,000	50,000	50,000	30,000	15,000
Optimization time (s)	4450	3640	3162	1926	1340

**Fig. 3.17** Comparison of the allowable and existing stress constraints for the 26-story-tower truss using the CSS [2]**Fig. 3.18** Convergence history of the 26-story-tower truss for the CSS algorithm [2]

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

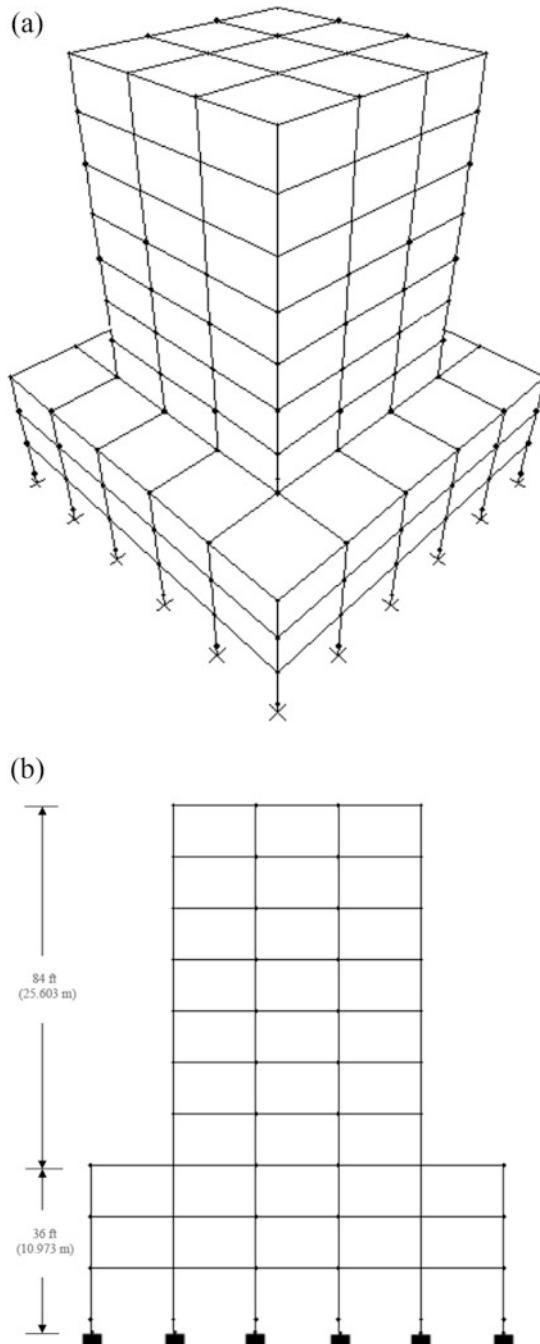
	Optimal cross-sectional areas ( $\text{cm}^2$ )							
	Members	Area		Members	Area		Members	Area
1	A <sub>1</sub>	0.962	21	A <sub>21</sub>	2.780	41	A <sub>41</sub>	0.417
2	A <sub>2</sub>	2.557	22	A <sub>22</sub>	0.430	42	A <sub>42</sub>	0.679
3	A <sub>3</sub>	1.650	23	A <sub>23</sub>	3.048	43	A <sub>43</sub>	19.584
4	A <sub>4</sub>	0.402	24	A <sub>24</sub>	5.112	44	A <sub>44</sub>	0.533
5	A <sub>5</sub>	0.657	25	A <sub>25</sub>	19.352	45	A <sub>45</sub>	1.640
6	A <sub>6</sub>	18.309	26	A <sub>26</sub>	0.476	46	A <sub>46</sub>	0.618
7	A <sub>7</sub>	0.346	27	A <sub>27</sub>	2.887	47	A <sub>47</sub>	0.531
8	A <sub>8</sub>	3.076	28	A <sub>28</sub>	19.500	48	A <sub>48</sub>	1.374
9	A <sub>9</sub>	2.235	29	A <sub>29</sub>	4.772	49	A <sub>49</sub>	19.656
10	A <sub>10</sub>	3.813	30	A <sub>30</sub>	5.063	50	A <sub>50</sub>	0.888
11	A <sub>11</sub>	0.856	31	A <sub>31</sub>	15.175	51	A <sub>51</sub>	4.456
12	A <sub>12</sub>	1.138	32	A <sub>32</sub>	1.176	52	A <sub>52</sub>	0.386
13	A <sub>13</sub>	3.374	33	A <sub>33</sub>	0.839	53	A <sub>53</sub>	10.398
14	A <sub>14</sub>	0.573	34	A <sub>34</sub>	1.394	54	A <sub>54</sub>	18.834
15	A <sub>15</sub>	19.530	35	A <sub>35</sub>	0.153	55	A <sub>55</sub>	18.147
16	A <sub>16</sub>	1.512	36	A <sub>36</sub>	0.247	56	A <sub>56</sub>	3.280
17	A <sub>17</sub>	2.667	37	A <sub>37</sub>	18.673	57	A <sub>57</sub>	2.972
18	A <sub>18</sub>	0.478	38	A <sub>38</sub>	0.696	58	A <sub>58</sub>	4.927
19	A <sub>19</sub>	17.873	39	A <sub>39</sub>	1.395	59	A <sub>59</sub>	0.288
20	A <sub>20</sub>	0.335	40	A <sub>40</sub>	0.422			
	Weight (N)	210,716						

### 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 an HS-based technique in real-size optimum design of steel frames 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 thereafter, as are 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



**Fig. 3.19** Schematic of an unbraced space frame [2]. (a) Three-dimensional view, (b) elevation, (c) plan, (d) member grouping

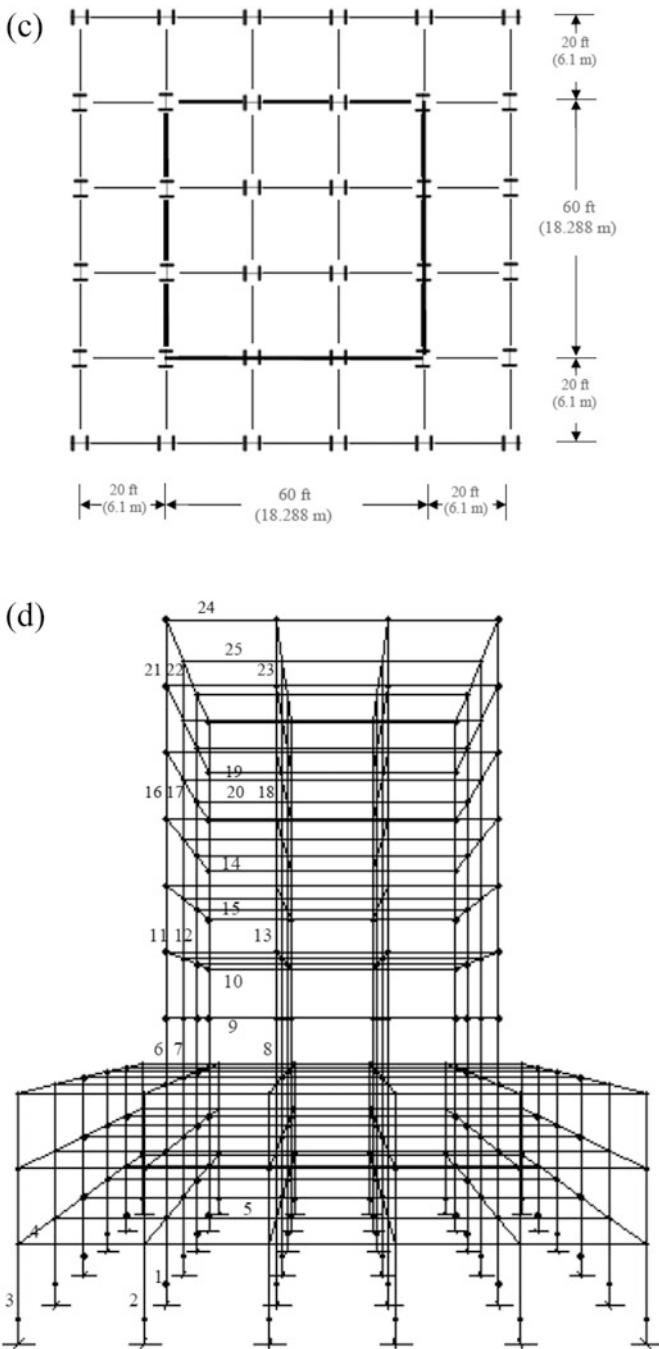
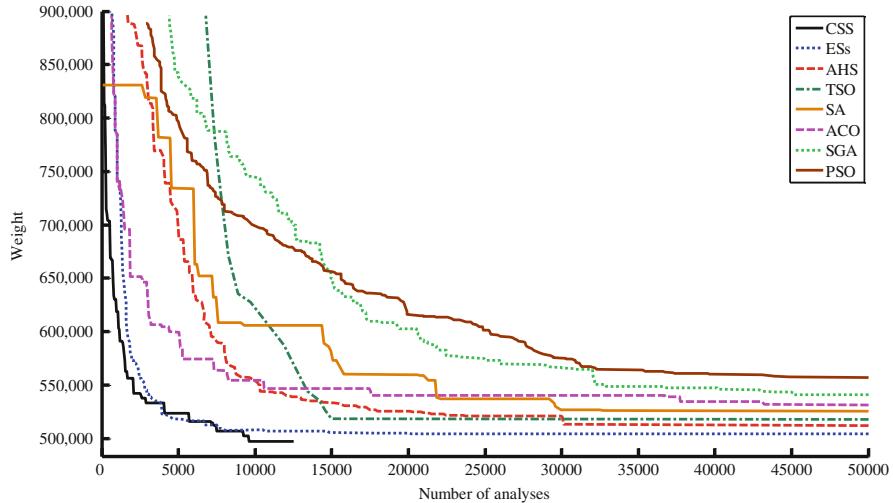


Fig. 3.19 (continued)



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

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 strategy 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.

### 3.5.5 A Braced Space Frame

The second frame example considered in this chapter is a 36-story braced space steel frame consisting of 814 joints and 1860 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.

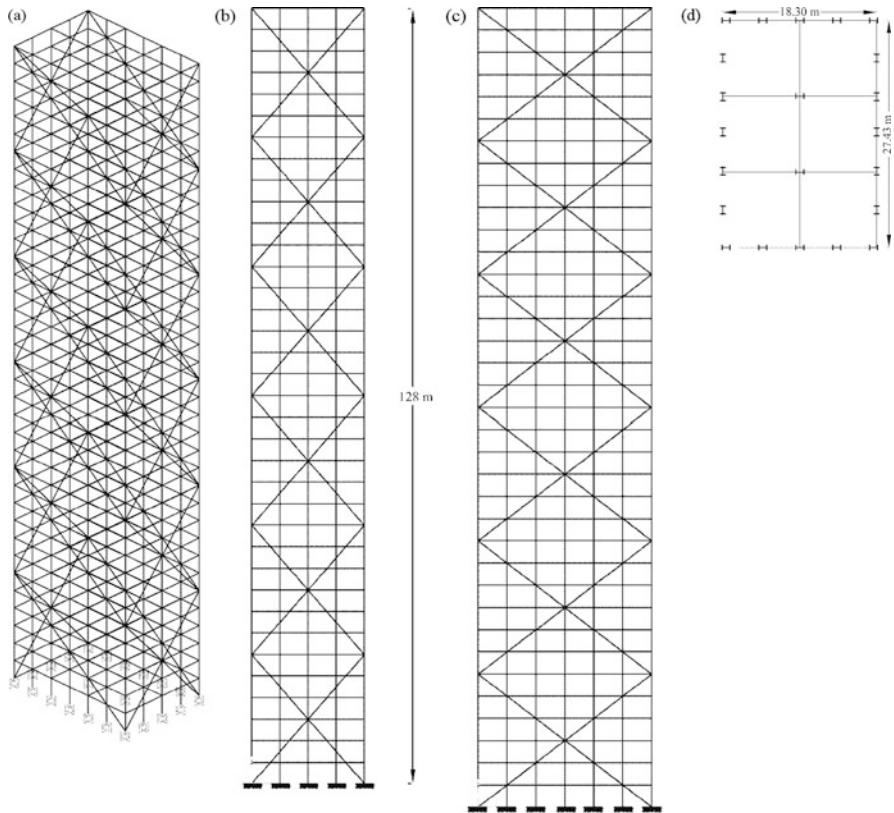
The 1860 frame members are collected in 72 different member groups, considering the symmetry of the structure and the practical fabrication requirements. That

**Table 3.9** Optimal design for the unbraced space frame

Element group	Optimal W-shaped sections Saka and Hasançebi [10]				Present work [2]
	SA	TSO	AHS	ESs	
1	W14 × 193	W14 × 193	W14 × 176	W14 × 193	W14 × 132
2	W8 × 48	W8 × 48	W14 × 48	W8 × 48	W21 × 55
3	W8 × 40	W8 × 40	W10 × 39	W10 × 39	W12 × 40
4	W10 × 22	W10 × 22	W10 × 22	W10 × 22	W10 × 33
5	W21 × 44	W21 × 50	W24 × 55	W21 × 50	W21 × 50
6	W12 × 65	W10 × 54	W12 × 65	W10 × 54	W12 × 65
7	W14 × 145	W14 × 120	W14 × 145	W14 × 109	W14 × 99
8	W14 × 145	W14 × 159	W14 × 159	W14 × 176	W14 × 120
9	W24 × 65	W21 × 44	W14 × 30	W18 × 40	W21 × 44
10	W24 × 55	W18 × 40	W18 × 40	W18 × 40	W21 × 44
11	W10 × 49	W10 × 45	W10 × 54	W10 × 49	W14 × 61
12	W14 × 90	W14 × 90	W14 × 90	W14 × 90	W10 × 88
13	W14 × 120	W12 × 120	W14 × 120	W14 × 109	W14 × 99
14	W16 × 36	W12 × 44	W14 × 34	W14 × 30	W18 × 35
15	W16 × 40	W16 × 36	W18 × 40	W16 × 36	W12 × 50
16	W12 × 40	W10 × 33	W8 × 31	W12 × 45	W21 × 68
17	W12 × 65	W12 × 65	W12 × 65	W12 × 65	W16 × 57
18	W12 × 26	W14 × 34	W18 × 35	W10 × 22	W24 × 68
19	W12 × 72	W12 × 79	W12 × 79	W12 × 79	W16 × 36
20	W16 × 36	W14 × 30	W14 × 30	W14 × 30	W16 × 31
21	W8 × 24	W10 × 39	W10 × 22	W8 × 35	W10 × 33
22	W10 × 49	W12 × 45	W10 × 45	W10 × 39	W16 × 31
23	W8 × 24	W12 × 35	W8 × 31	W8 × 31	W8 × 28
24	W12 × 26	W6 × 20	W10 × 22	W8 × 18	W8 × 18
25	W12 × 26	W12 × 26	W12 × 26	W14 × 30	W10 × 26
Weight (kg)	238,756.5	235,167.5	232,301.2	228,588.3	225,654.0

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 3-story deep members, and two bracing groups are specified in every six stories.

The minimum weight design of the frame is equal to 2301.69 t for the CSS algorithm, while it is 2383.60 and 4438.17 t 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.



**Fig. 3.21** Schematic of a braced space frame [2]. (a) 3D view of the frame, (b) Front view, (c) side view, (d) plan

## 3.6 Discussion

### 3.6.1 Efficiency of the CSS Rules

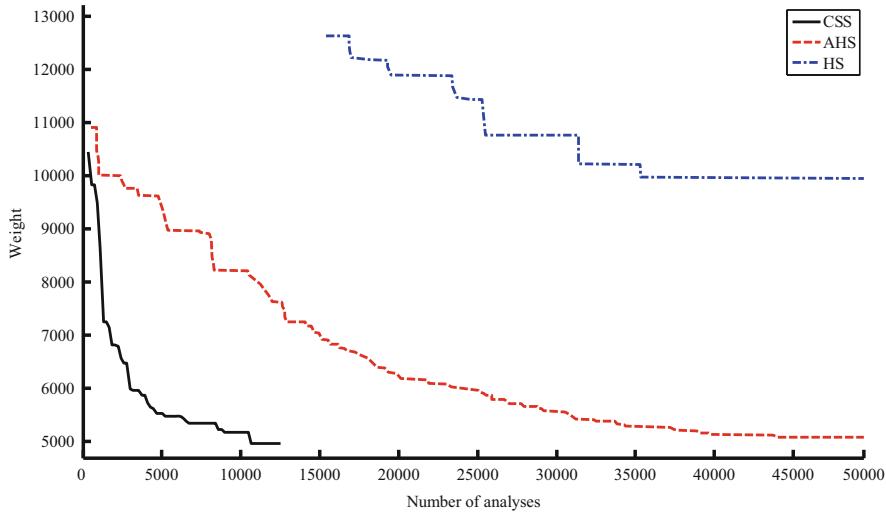
Solution of a number of design examples shows the superiority of the CSS algorithm to the other existing metaheuristics. To investigate the effect of some utilized rules, a number of the CSS-based algorithms are defined as follows:

**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 ( $p_{ij} = 1$ ).



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

**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

**Case 3:** Rule 4 is changed as:

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 shows the results of the 50 runs of the first example for 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, 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's movement. The focus of the PSO is placed upon finding 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's 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 recognize the end of the global phase and change the movement updating equation for the local phase to have a better balance between the exploration and exploitation. 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 metaheuristic 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 metaheuristic 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, New York

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.245–287
6. 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
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, Hasanc̄ebi O (2009) Design code optimization of steel structures using adaptive harmony search algorithm, Chapter 3. In Geem ZW (ed) *Harmony search algorithms for structural design*. Springer, Berlin. SCI 239:79–120
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):1233–1250
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 the 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 demonstrate 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 the governing laws of motion from the Newtonian mechanics.

### 4.2.1 Magnetic Laws

#### 4.2.1.1 Magnetic Fields

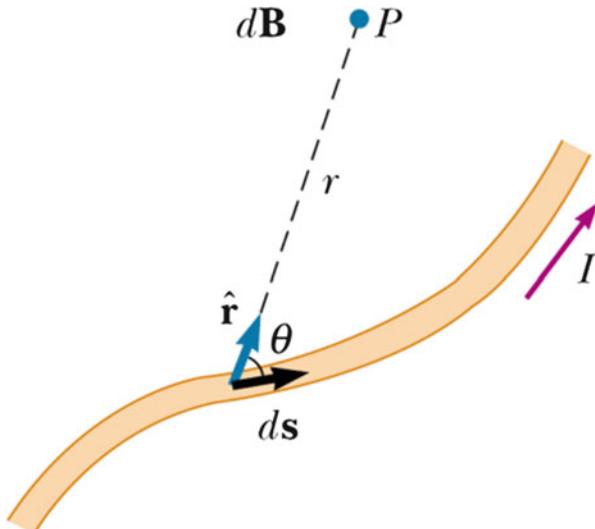
There is a relation between electric and magnetic forces, and these forces are called electromagnetic forces. The region surrounding any stationary or moving charged particle contains electric fields. In addition to electric field, the region surrounding any moving charged particle also contains magnetic 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 was deflected by a current-carrying conductor. Shortly after this discovery, Biot and Savar 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:

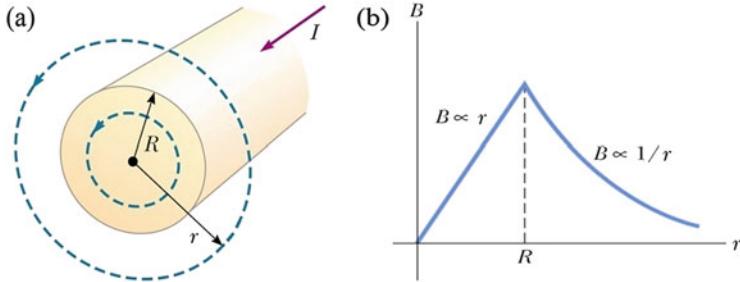
$$d\mathbf{B} = \frac{\mu_0}{4\pi} \frac{Id\mathbf{s} \times \hat{\mathbf{r}}}{r^2} \quad (4.1)$$

Here,  $d\mathbf{B}$  is the magnetic field at point  $P$ ,  $\mu_0$  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 the Biot–Savar law, the magnetic field produced by wire at a point like  $P$  outside the wire can be determined 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 the 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]

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

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

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

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  $r=R$ , Eqs. (4.2) and (4.3) have an overlap, and both give identical magnitude for the magnetic field. A plot of these two equations from Ref. [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:

$$B_P = \sum_{i=1}^n B_{ip} \quad (4.4)$$

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  $i$ th wire at point  $P$  which can be expressed as:

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

#### 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 it. Experiments on charged particles moving in a magnetic field result 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:

$$\mathbf{F}_B = q\mathbf{v} \times \mathbf{B} \quad (4.6)$$

where  $\mathbf{B}$  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  $\mathbf{B}$  can be calculated using Eq. (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 physical 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 governing laws of motion from the Newtonian mechanics, 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 each CP is:

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

where  $\text{fit}(i)$  is the objective function value of the  $i$ th CP and  $\text{fitbest}$  and  $\text{fitworst}$  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,

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

where  $\mathbf{X}_i$  and  $\mathbf{X}_j$  are the positions of the  $i$ th and  $j$ th CPs and  $r_{ij}$  is the separation distance of these CPs.  $\mathbf{X}_{best}$  is the position of the best current CP, and  $\epsilon$  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:

$$p_{ij} = \begin{cases} 1 & \Leftrightarrow \frac{fit(i) - fit_{best}}{fit(j) - fit(i)} > rand, or, fit(j) > fit(i) \\ 0 & \Leftrightarrow else. \end{cases} \quad (4.9)$$

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

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

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

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

$$\mathbf{V}_{j,new} = \frac{\mathbf{X}_{j,new} - \mathbf{X}_{j,old}}{\Delta t} \quad (4.12)$$

where  $rand_{j1}$  and  $rand_{j2}$  are two random numbers that are 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 functions:

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

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 CPs with better objective function values 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**

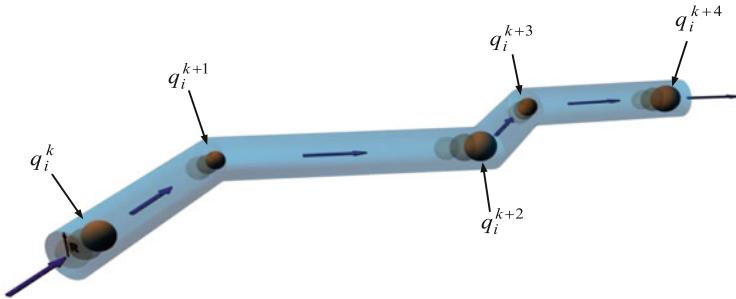
The inspiration of the standard CSS algorithm is based on a group of charged particles that exert electric forces on each other based on their charges and their separation distances. After computing the electric forces, each particle moves, and its movement is calculated by using the governing laws of motion from the Newtonian mechanics. 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  $\mathbf{B}$ , it 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], exerted on the charged particle is:

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

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 charged particle (CP) contains electrical charge. These CPs produce electric fields and exert electric forces on each other. When a



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

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 place that a wire changes its direction is the position of the CP at the end of its movement. 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 create 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. 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  $\Delta q$  is the amount of charge that passes through this area in a time interval  $\Delta t$ , the average current  $I_{avg}$  will be equal to the charge that passes through the cross section per unit time:

$$I_{avg} = \frac{\Delta q}{\Delta t} \quad (4.15)$$

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, Eq. (4.13) can be written as:

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

where  $(I_{avg})_{ik}$  is the average current in the  $i$ th wire (corresponding to the  $i$ th CP) in the  $k$ th movement (iteration), and  $q_i^{k-1}$  and  $q_i^k$  are the charges of the  $i$ th CP at the start and end of its  $k$ th movement, respectively. Equation (4.14) 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 a deterioration in the quality of the CP.

The charges of the CPs are defined by Eq. (4.6). This expression for computing electric charges results in values between 0 and 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:

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

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

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

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

where  $df_{i,k}$  is the variation of the objective function in the  $k$ th movement (iteration).  $fit_k(i)$  and  $fit_{k-1}(i)$  are the values of the objective function of the  $i$ th CP at the start of the  $k$ th 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 [Eqs. (4.16) and (4.17)] 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 Eq. (4.7). Thus,  $r_{ij}$  now means the distance between the  $i$ th wire and  $i$ th virtual CP to the  $j$ th charged particle.

In the expression for computing the magnetic force (Eq. (4.5)), 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:

$$\mathbf{v}_{rel} = \frac{\mathbf{X}_i - \mathbf{X}_j}{\Delta t} \quad (4.20)$$

where  $\mathbf{X}_i$  and  $\mathbf{X}_j$  are the positions of the  $i$ th and  $j$ th CPs and the  $\Delta t$  is the time step that is set to unity. Therefore the relative velocity can be rewritten as:

$$\mathbf{v}_{rel} = \mathbf{X}_i - \mathbf{X}_j \quad (4.21)$$

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

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

where  $q_i$  is the charge of the  $i$ th 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  $i$ th wire (CP) on the  $j$ th CP. This term can be computed by the following expression:

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

where  $fit(i)$  and  $fit(j)$  are the objective values of the  $i$ th and  $j$ th CPs, 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 Eq. (4.8). 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 the  $i$ th virtual wire, is negative, according to the concept of the electric current, a negative value means that the  $i$ th CP did not experience 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  $i$ th CP experiences an improvement in its movement, it will attract the  $j$ th 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:

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

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

The quantity  $R$  is the radius of the virtual wires, and if a charged particle is placed outside or inside of a virtual wire, the magnetic force that is 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 (Eq. (4.3)). This means that the formulation of the magnetic force for global and local phases should be separated (Eq. (4.24)). A suitable value for  $R$  in Eq. (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 Eq. (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 Eq. (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 Eq. (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 experiences an improvement in its value of the objective function, it should attract other CPs, regardless of 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:

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

$$j = 1, 2, \dots, N$$

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

Consider the  $i$ th CP among all of the CPs; this CP has a charge which is larger than the charges of a number of CPs. Considering the rules of the CSS, the  $i$ th 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  $i$ th CPs also moved to a new position. In this movement, the  $i$ th particle may experience deterioration in its objective function value. Due to this decrease, the new charge of the  $i$ th particle will be decreased, but its charge may still be larger than a number of CPs. According to the CSS algorithm, the  $i$ th particle still attracts all other CPs with smaller charges regardless of the failure of the  $i$ th 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 from which the algorithm benefits. By this concept, the  $i$ th 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 exerted on CPs by the  $i$ th CP are 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 Eq. (4.11) 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 with 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. 5.1. In Sect. 5.2 the results of the MCSS are presented and compared to those of the CSS and other optimization algorithms in the literature. Finally, in Sect. 5.3 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 Ref. [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 Ref. [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 Ref. [6], and GA and some of its variants are 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.,  $\epsilon = |f_{\min} - f_{final}| = 10^{-4}$ . 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 of CPs and the number of iterations are set to small values. Thus, another

**Table 4.1** Description of the mathematical benchmarks

Function name	Interval	Function	Global minimum
Alfifi-Pentini	$\mathbf{X} \in [-10, 10]^2$	$f(\mathbf{X}) = \frac{1}{4}x_1^4 - \frac{1}{2}x_1^2 + \frac{1}{10}x_1 + \frac{1}{2}x_2^2$	-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	$\mathbf{X} \in [-50, 50]^2$	$f(\mathbf{X}) = x_1^2 + 2x_2^2 - \frac{3}{10} \cos(3\pi x_1) \cdot \cos(4\pi x_2) + \frac{3}{10}$	0.0
Becker and Lago	$\mathbf{X} \in [-10, 10]^2$	$f(\mathbf{X}) = ( x_1  - 5)^2 + ( x_2  - 5)^2$	0.0
Branin	$0 \leq x_2 \leq 15,$ $-5 \leq x_1 \leq 10$	$f(\mathbf{X}) = (x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1)^2 + 10 \cdot (1 - \frac{1}{8\pi}) \cos(x_1) + 10$	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	$\mathbf{X} \in [-5, 5]^2$	$f(\mathbf{X}) = 2x_1^2 - 1.05x_1^4 + \frac{1}{6}x_1^6 + x_1x_2 + x_2^2$	0.0
Cosine mixture	$\mathbf{X} \in [-1, 1]^n, n = 4$	$f(\mathbf{X}) = \sum_{i=1}^n x_i^2 - \frac{1}{10} \sum_{i=1}^n \cos(5\pi x_i)$	-0.4
DeJong	$\mathbf{X} \in [-5.12, 5.12]^3$	$f(\mathbf{X}) = x_1^2 + x_2^2 + x_3^2$	0.0
Exponential	$\mathbf{X} \in [-1, 1]^n,$ $n = 2, 4, 8$	$f(\mathbf{X}) = -\exp\left(-0.5 \cdot \sum_{i=1}^n x_i^2\right)$	-1.0
Goldstein and Price	$\mathbf{X} \in [-2, 2]^2$	$f(\mathbf{X}) = \left[1 + (x_1 + x_2 + 1)^2 \cdot (19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)\right] \times \left[30 + (2x_1 - 3x_2)^2 \cdot (18 - 32x_1 - 12x_1 + 48x_2 - 36x_1x_2 + 27x_2^2)\right]$	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	$\mathbf{X} \in [-30, 30]^3$	$f(\mathbf{X}) = -\sum_{i=1}^4 c_i \exp\left(-\sum_{j=1}^3 a_{ij}(x_j - p_{ij})^2\right), \quad a = \begin{bmatrix} 3 & 10 & 30 \\ 0.1 & 10 & 35 \\ 3 & 10 & 35 \\ 0.1 & 10 & 35 \end{bmatrix}, \quad c = \begin{bmatrix} 1 \\ 1.2 \\ 3 \\ 3.2 \end{bmatrix}$ $p = \begin{bmatrix} 0.3689 & .117 & 0.2673 \\ 0.4699 & 0.4387 & 0.747 \\ 0.1091 & 0.8732 & 0.5547 \\ 0.03815 & 0.5743 & 0.8828 \end{bmatrix}$	-3.862782

(continued)

Table 4.1 (continued)

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

**Table 4.2** Performance comparison for the benchmark problems

Function	GEN	GEN-S	GEN-S-M	GEN-S-M-LS	CSS	MCSS
AP	1360 (0.99)	1360	1277	1253	804	437
Bf1	3992	3356	1640	1615	1187	542
Bf2	20,234	3373	1676	1636	742	556
BL	19,596	2412	2439	1436	423	481
Branin	1442	1418	1404	1257	852	351
Camel	1358	1358	1336	1300	575	384
Cb3	9771	2045	1163	1118	436	288
CM	2105	2105	1743	1539	1563	538
Dejoung	9900	3040	1462	1281	630	387
Exp2	938	936	817	807	132	183
Exp4	3237	3237	2054	1496	867	317
Exp8	3237	3237	2054	1496	1426	659
Goldstein and Price	1478	1478	1408	1325	682	450
Griewank	18,838 (0.91)	3111 (0.91)	1764	1652 (0.99)	1551	1272
Hartman 3	1350	1350	1332	1274	860	344
Hartman 6	2562 (0.54)	2562 (0.54)	2530 (0.67)	1865 (0.68)	1783	908
Rastrigin	1533 (0.97)	1523 (0.97)	1392	1381	1402	1252
Rosenbrock	9380	3739	1675	1462	1452	1424
Total	112,311 (96.7)	41,640 (96.7)	29,166 (98.16)	25,193 (98.16)	17,367	10,773

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 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

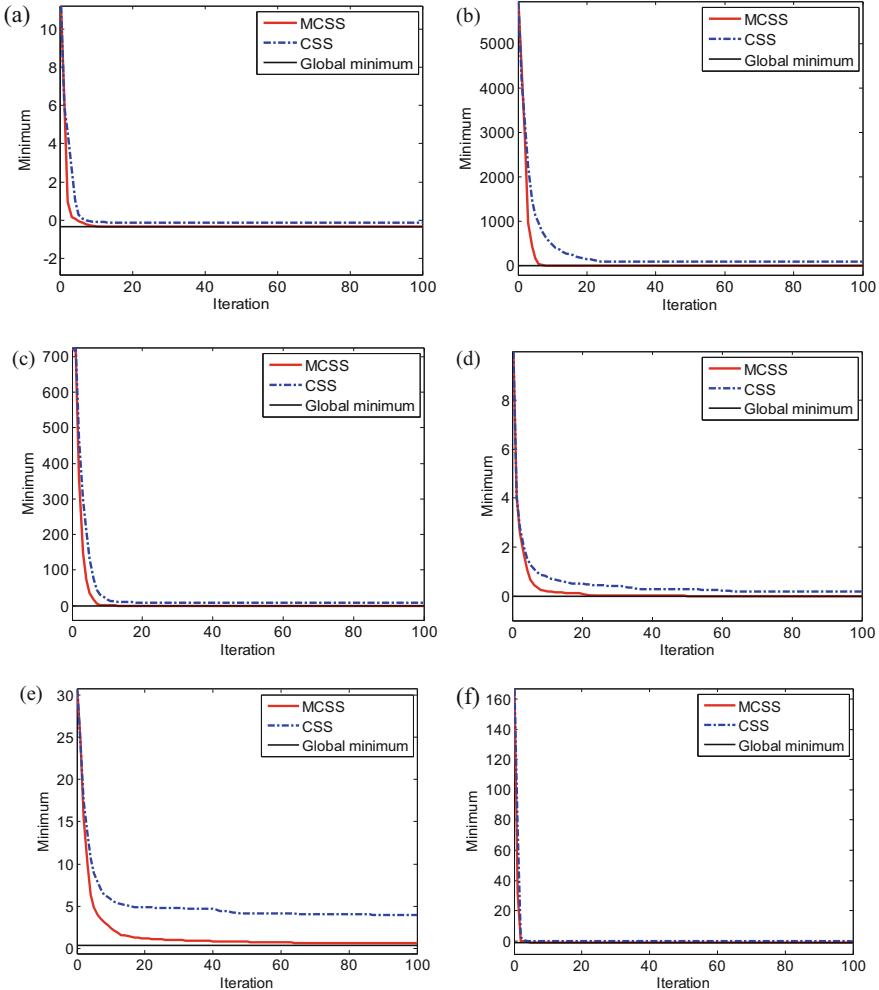
Now in the following we want to ensure that the results of MCSS in Table 4.3 are better than CSS algorithm. For this purpose, we apply a multiproblem analysis using statistical tests. We apply the test on the obtained errors by each algorithm. If we have the normality condition for our sample of results, a parametric pair *t*-test can be suitable. We first analyze a safe usage of parametric tests. We utilized two

**Table 4.3** Numerical comparison of CSS and MCSS algorithms. Number of CPs = 10, maximum number of iterations = 100

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
Hartman 3	-3.862782	-3.491627	-3.816318	0.371155	0.046464
Hartman 6	-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

normality tests including Kolmogorov–Smirnov and Shapiro–Wilks 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  $<0.05$ . 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 are 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 the Wilcoxon test that is a nonparametric test for pairwise comparisons. The method of this test is described in Ref. [7]. The result of this test can be summarized as:

- The  $p$ -value obtained by the Wilcoxon test is 0.00. Consequently, the Wilcoxon 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.

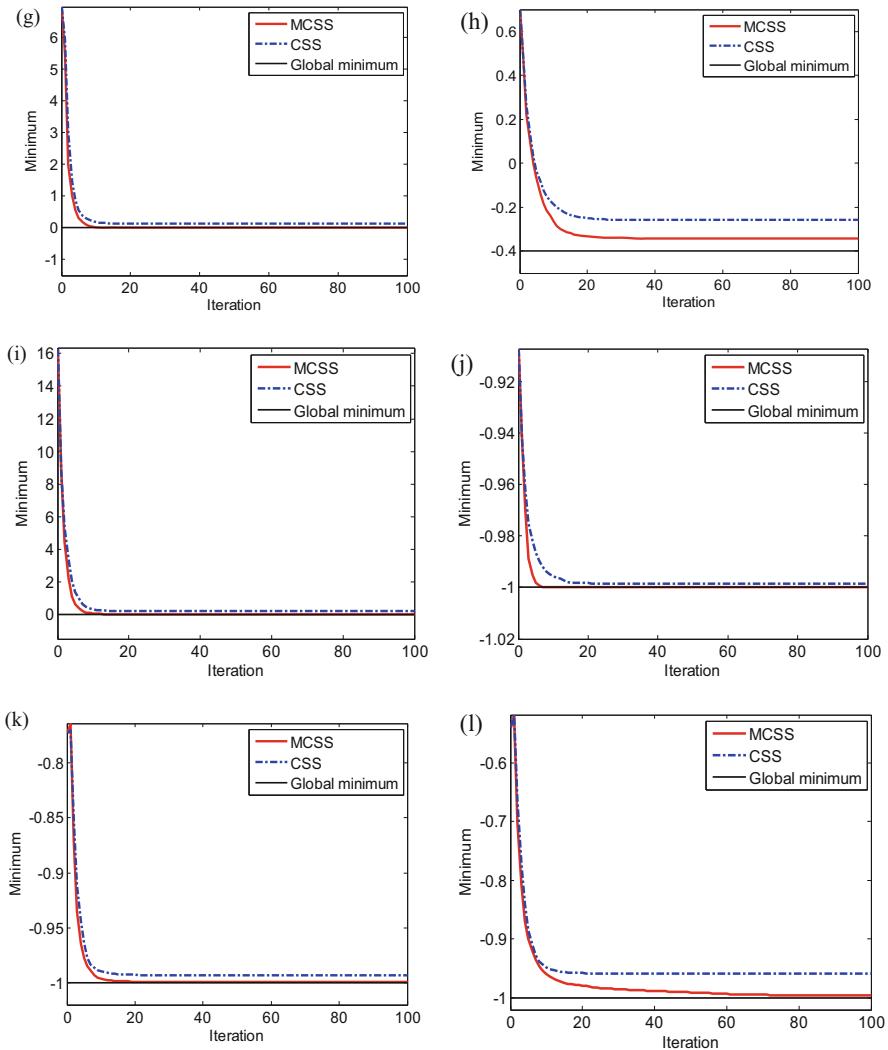


**Fig. 4.4** Comparison of the convergence rate of optimizing mathematical benchmarks [1]: (a) AP, (b) Bf1, (c) Bf2, (d) BL, (e) Branin, (f) Camel, (g) Cb3, (h) CM, (i) Dejoung, (j) Exp2, (k) Exp4, (l) Exp8, (m) Goldstein and price, (n) Griewank, (o) Hartman3, (p) Hartman6, (q) Rastrigin, (r) Rosenbrock

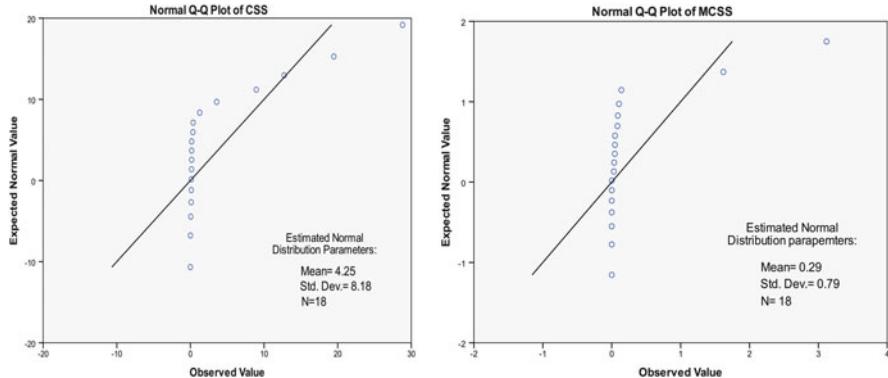
#### 4.2.4.2 Comparison Between MCSS and Other State-of-the-Art Algorithms

##### 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

**Fig. 4.4** (continued)**Table 4.4** Normality tests and their  $p$ -values over multiproblem 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]

description of test functions is presented by Suganthan et al. [8]. The set of these test functions consists of the following functions:

- Five displaced unimodal 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 multimodal functions (f6–f7)
  - Seven 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.
  - Two 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 are 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  $< 10^{-8}$ . Table 4.5 shows the

**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	4771.0	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	48.5	1E-09	0.012	1E-09	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.
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	1050	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	1060	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	2144.922	272.4173	189.7254	203.5414	213.4814	152.6623	273.1534	70635.3	194.261	191.12	167.97

**Table 4.6** The Wilcoxon test results

MCSS versus	R <sup>+</sup>	R <sup>-</sup>	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	.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

official results of the participated algorithms obtained from Garcia et al. [9]. The description of each algorithm is given in Ref. [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 with other algorithms. We want to find out whether the results of MCSS have a significant difference in comparison with the other algorithms. This is a multiproblem analysis; therefore, a nonparametric test is more suitable in this case. We utilized Wilcoxon's test. This test performs pairwise comparisons between two algorithms. In this test, MCSS is compared to some other algorithms.

Table 4.6 summarizes the results of applying the Wilcoxon test. Table 4.6 includes sum of ranking and *p*-value of each comparison. The method of this test is simply described in Ref. [7]. The significance level  $\alpha$  is considered as 0.05. In each comparison when the corresponding *p*-value is  $<0.05$ , 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 independent from another one. If we draw a conclusion involving more than one pairwise comparison in Wilcoxon's analysis, an accumulated error which is merged up by combination of pairwise comparisons will be obtained. In statistical 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:

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

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 Ref. [9].

Considering the values of Table 4.6, the  $p$ -value of all of the comparisons except MCSS versus G-CMA-ES is less than significance level  $\alpha = 0.05$ , and 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 independent 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 Eq. (4.22):

$$p = 1 - ((1 - 0.16) \cdot (1 - 0.0) \cdot (1 - 0.0) \cdot (1 - 0.05) \cdot (1 - 0.24) \cdot (1 - 0.001) \cdot (1 - 0.025) \cdot (1 - 0.048) \cdot (1 - 0.027) \cdot (1 - 0.001)) = 0.17765 \quad (4.27)$$

Based on this analysis 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  $100(1 - \alpha) = 82.23\%$ .

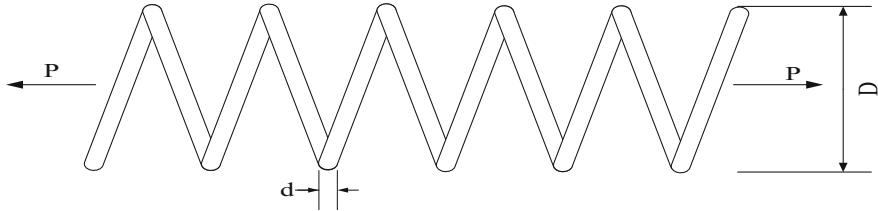
#### 4.2.5 Engineering Examples

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. For handling constraints, a simple penalty function is utilized to prevent adding the effect of a robust constraint 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 constraints, 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:



**Fig. 4.6** Schematic of the tension/compression spring with indication of design variables

**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 Talatahari (CSS) [6]	0.051744	0.358532	11.165704	0.0126384
Present work [1]	0.051645	0.356496	11.271529	0.0126192

$$f_{cost}(\mathbf{X}) = (x_3 + 2)x_2x_1^2 \quad (4.28)$$

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

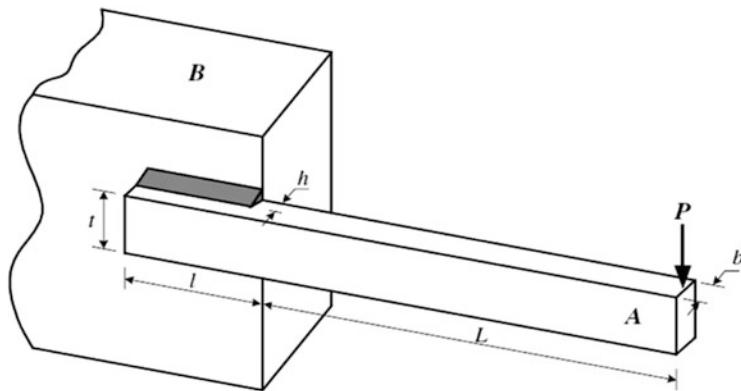
The decision variables are limited as:

$$\begin{aligned} 0.05 &\leq x_1 \leq 2.0, \\ 0.25 &\leq x_2 \leq 1.3, \\ 2.0 &\leq x_3 \leq 15. \end{aligned} \quad (4.30)$$

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 Talatahari [14, 26]. The results of the best solutions found by different methods are presented in Table 4.7. From Table 4.7 it can be understood that the best solution found by MCSS is better than other

**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 Talatahari (CSS) [6]	0.0126384	0.012852	0.013626	8.3564e-5
Present work [1]	0.0126192	0.012794	0.013962	5.3491e-5

**Fig. 4.7** Schematic of the welded beam system

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 constraints, such as shear ( $\tau$ ) and bending ( $\sigma$ ) stresses, buckling load ( $P_c$ ), end deflection ( $\delta$ ), and end side constraint. Design variables are  $h(=x_1)$ ,  $l(=x_2)$ ,  $t(=x_3)$ , and  $b(=x_4)$ . By considering the setup, welding labor, and the materials costs, the cost function can be expressed as:

$$f_{\cos t}(\mathbf{X}) = 1.1047x_1^2x_2 + 0.04811x_3x_4 \cdot (14.0 + x_2) \quad (4.31)$$

Subjected to the following constraints:

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

where

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

and variable boundaries are:

$$\begin{aligned}
0.1 &\leq x_1 \leq 2.0, \\
0.1 &\leq x_2 \leq 10, \\
0.1 &\leq x_3 \leq 10, \\
0.1 &\leq x_3 \leq 2.0.
\end{aligned} \tag{4.34}$$

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 Talatahari [16] by ACO. This problem is also solved by Kaveh and Talatahari [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

**Table 4.9** 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 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 Talatahari [16]	0.205700	3.471131	9.036683	0.205731	1.724918
Kaveh and Talatahari (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

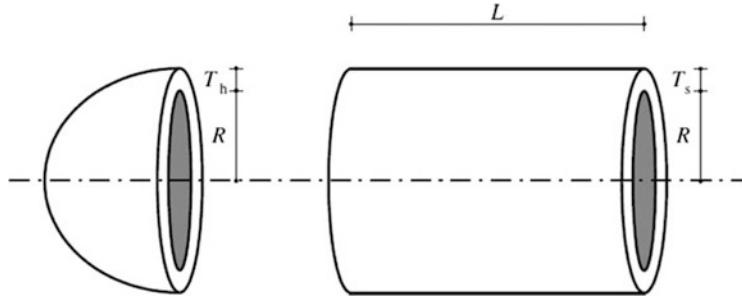
**Table 4.10** Statistical results of different methods for the design of welded beam

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 Talatahari (CSS) [6]	1.724866	1.739654	1.759479	0.008064
Present work [1]	1.724853	1.735438	1.753681	0.009527

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 0.0625 in the available thickness of the rolled steel plates, but  $R$  and  $L$  are continuous variables. The mathematical expression of the cost function is:



**Fig. 4.8** Schematic of the pressure vessel and its design variables

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

The constraint areas are as follows:

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

The search space is defined as:

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

Various types of methods have been used to solve this problem. Some of these approaches are a branch and bound method [19], an augmented Lagrangian multiplier approach [20], a genetic adaptive search [21], a GA-based algorithm [12], a feasibility-based tournament selection scheme [13], a coevolutionary 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.

**Table 4.11** Optimum results for the design of welded beam

Methods	Optimal design variables				
	$x_1(T_s)$	$x_2(T_h)$	$x_3(R)$	$x_4(L)$	$f_{cost}$
Sandgren [19]	1.125000	0.625000	47.700000	117.701000	8129.1036
Kannan and Kramer [20]	1.125000	0.625000	58.291000	43.690000	7198.0428
Deb and Gene [21]	0.937500	0.500000	48.329000	112.679000	6410.3811
Coello [12]	0.812500	0.437500	40.323900	200.000000	6288.7445
Coello and Montes [13]	0.812500	0.437500	42.097398	176.654050	6059.9463
He and Wang [14]	0.812500	0.437500	42.091266	176.746500	6061.0777
Montes and Coello [15]	0.812500	0.437500	42.098087	176.640518	6059.7456
Kaveh and Talatahari [16]	0.812500	0.437500	42.098353	176.637751	6059.7258
Kaveh and Talatahari (CSS) [6]	0.812500	0.437500	42.103624	176.572656	6059.0888
Present work [1]	0.812500	0.437500	42.107406	176.525589	6058.6233

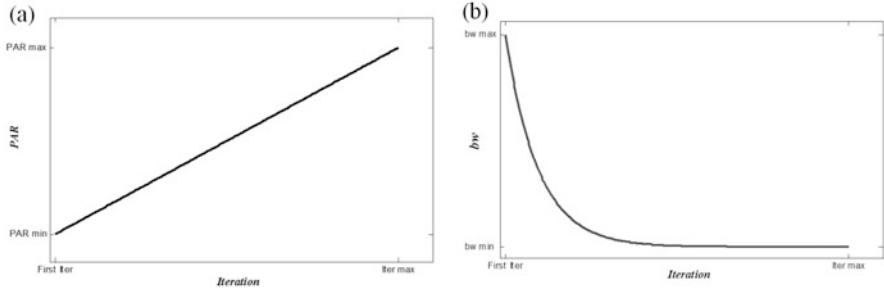
**Table 4.12** Statistical results of different methods for the design of welded beam

Methods	Best	Mean	Worst	Standard deviation
Sandgren [19]	8129.1036	N/A	N/A	N/A
Kannan and Kramer [20]	7198.0428	N/A	N/A	N/A
Deb and Gene [21]	6410.3811	N/A	N/A	N/A
Coello [12]	6288.7445	6293.8432	6308.1497	7.4133
Coello and Montes [13]	6059.9463	6177.2533	6469.3220	130.9297
He and Wang [14]	6061.0777	6147.1332	6363.8041	86.4545
Montes and Coello [15]	6059.7456	6850.0049	7332.8798	426.0000
Kaveh and Talatahari [16]	6059.7258	6081.7812	6150.1289	67.2418
Kaveh and Talatahari (CSS) [6]	6059.0888	6067.9062	6085.4765	10.2564
Present work [1]	6058.6233	6073.5931	6108.5479	24.6712

### 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 lead 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



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

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 lead to the improvement of best solutions in final iterations and convergence of the algorithm to optimal solution vector. To improve the performance of the HS scheme and eliminate the drawbacks which correspond 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 follows: [22]:

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

and

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

$$c = \frac{\ln(bw_{\min}/bw_{\max})}{iter_{\max}}, \quad (4.40)$$

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 follows:

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

where  $Fix(X)$  is a function which rounds each element 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:

$$\begin{aligned} & \text{Find} && X = [x_1, x_2, x_3, \dots, x_n] \\ & \text{to minimize} && Mer(X) = f_{\text{penalty}}(X) \times W(X) \\ & \text{subject to} && \sigma_{\min} < \sigma_i < \sigma_{\max} \quad i = 1, 2, \dots, nm \\ & && \delta_{\min} < \delta_i < \delta_{\max} \quad i = 1, 2, \dots, nn \end{aligned} \quad (4.42)$$

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;  $f_{\text{penalty}}(X)$  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;  $\sigma_i$  and  $\delta_i$  are the stress of members and nodal displacements, respectively; and  $\min$  and  $\max$  mean the lower and upper bounds of constraints, respectively. The cost function can be expressed as:

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

where  $\rho_i$  is the material density of the member  $i$ ,  $L_i$  is the length of the  $i$ th member, and  $A_i$  is the cross-sectional area of the member  $i$ .

The penalty function can be defined as:

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

where  $np$  is the number of multiple loadings. Here  $\varepsilon_1$  is taken as unity, and  $\varepsilon_2$  is set to 1.5 in the first iterations of the search process, but gradually it is increased to 3 [24].  $\varphi_{\sigma}^k$  and  $\varphi_{\delta}^k$  are the summation of stress penalties and nodal displacement penalties for  $k$ th charged particle which are mathematically expressed as:

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

$$\varphi_{\delta} = \sum_{i=1}^{nn} \max \left( \left| \frac{\delta_i}{\bar{\delta}_i} \right| - 1, 0 \right), \quad (4.46)$$

where  $\sigma_i$  and  $\bar{\sigma}_i$  are the stress and allowable stress in member  $i$ , respectively, and  $\delta_i$  and  $\bar{\delta}_i$  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, are shown in Fig. 4.10.

There are ten design variables in this example and a set of pseudo-variables ranging from 0.1 to 35.0 in<sup>2</sup> (0.6452–225.806 cm<sup>2</sup>).

In this problem two cases are considered:

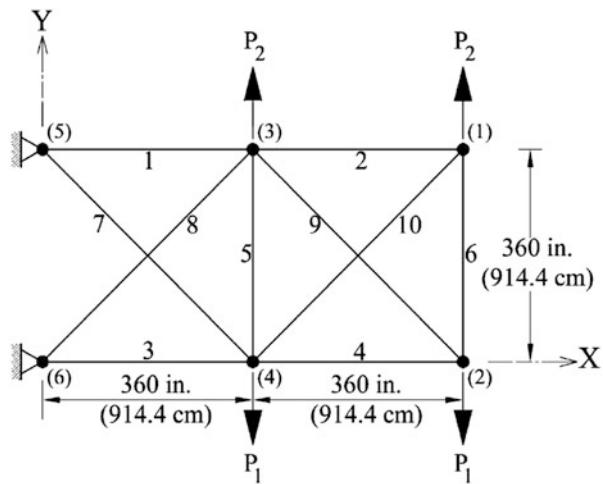
Case 1,  $P_1 = 100$  kips (444.8 kN) and  $P_2 = 0$ , and Case 2,  $P_1 = 150$  kips (667.2 kN) and  $P_2 = 50$  kips (222.4 kN).

**Table 4.13** The allowable steel pipe sections taken from AISC code

No.	Area (in <sup>2</sup> )	Area (mm <sup>2</sup> )	No.	Area (in <sup>2</sup> )	Area (mm <sup>2</sup> )
1	0.111	71.613	33	3.84	2477.414
2	0.141	90.968	34	3.87	2496.769
3	0.196	126.451	35	3.88	2503.221
4	0.25	161.29	36	4.18	2696.769
5	0.307	198.064	37	4.22	2722.575
6	0.391	252.258	38	4.49	2896.768
7	0.442	285.161	39	4.59	2961.284
8	0.563	363.225	40	4.8	3096.768
9	0.602	388.386	41	4.97	3206.445
10	0.766	494.193	42	5.12	3303.219
11	0.785	506.451	43	5.74	3703.218
12	0.994	641.289	44	7.22	4658.055
13	1	645.16	45	7.97	5141.925
14	1.228	792.256	46	8.53	5503.215
15	1.266	816.773	47	9.3	5999.988
16	1.457	939.998	48	10.85	6999.986
17	1.563	1008.385	49	11.5	7419.43
18	1.62	1045.159	50	13.5	8709.66
19	1.8	1161.288	51	13.9	8967.724
20	1.99	1283.868	52	14.2	9161.272
21	2.13	1374.191	53	15.5	9999.98
22	2.38	1535.481	54	16	10,322.56
23	2.62	1690.319	55	16.9	10,903.2
24	2.63	1696.771	56	18.8	12,129.01
25	2.88	1858.061	57	19.9	12,838.68
26	2.93	1890.319	58	22	14,193.52
27	3.09	1993.544	59	22.9	14,774.16
28	1.13	729.031	60	24.5	15,806.42
29	3.38	2180.641	61	26.5	17,096.74
30	3.47	2238.705	62	28	18,064.48
31	3.55	2290.318	63	30	19,354.8
32	3.63	2341.931	64	33.5	21,612.86

The material density is 0.1 lb/in<sup>3</sup> (2767.990 kg/m<sup>3</sup>), 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 all nodes in both vertical and horizontal directions are subjected to the displacement limits of  $\pm 2.0$  in (5.08 cm). Figure 4.11 shows a comparison of the convergence history of both cases for MCSS and IMCSS algorithms.

**Fig. 4.10** Schematic of a 10-bar planar truss structure



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 reaches its best solutions after 20,000 analyses and the PSO and PSOPC algorithms after 3000 iterations (150,000 analyses). The HPSACO algorithm finds the best solution after 10,650 and 9925 analyses, for Case 1 and Case 2, respectively.

The MCSS and IMCSS algorithms achieve the best solutions after 355 iterations (8875 analyses) and 339 iterations (8475 analyses), respectively. The best weights of IMCSS are 5064.6 lb for Case 1 and 4679.15 for Case 2.

As seen in both tables, although the best weights of IMCSS in both cases are a little bigger than the HPSACO, it has lower penalty values compared to 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].

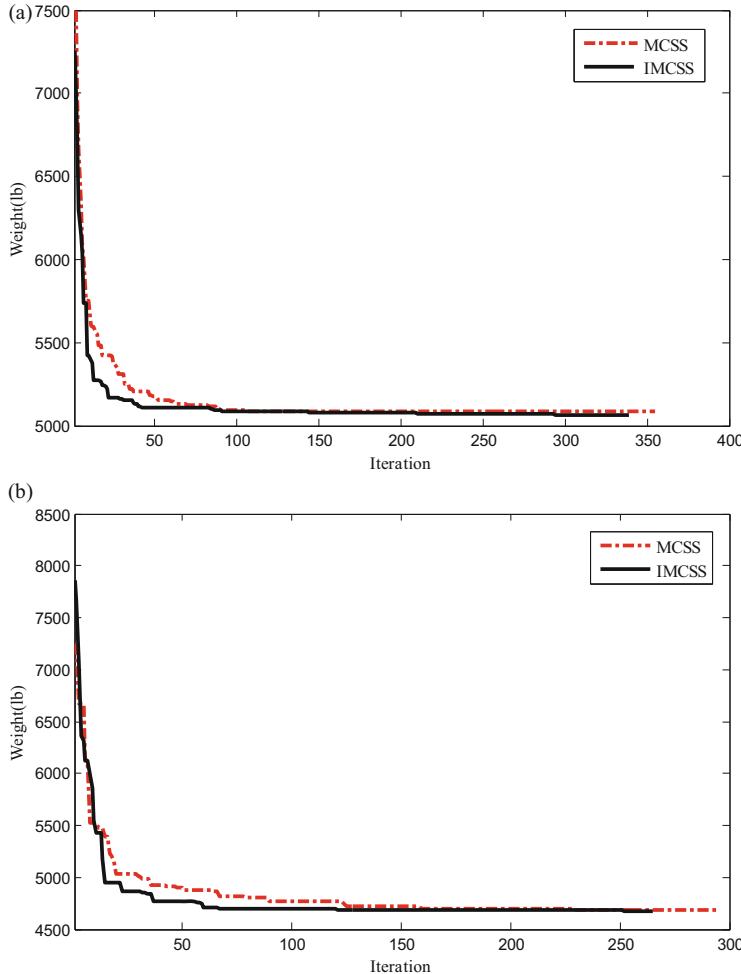
The members of this structure are divided into 12 groups: (1) A1–A4, (2) A5–A10, (3) A11–A13, (4) A14–A17, (5) A18–A23, (6) A24–A26, (7) A27–A30, (8) A31–A36, (9) A37–A39, (10) A40–A43, (11) A44–A49, and (12) A50–A52.

The material density is 7860.0 kg/m<sup>3</sup> and the modulus of elasticity is  $2.07 \times 10^5$  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 weights of MCSS and IMCSS algorithms are 1904.05 lb and 1902.61 lb, respectively, while for DHPSCAO is 1904.83 lb.

The MCSS and IMCSS algorithms find the best solutions after 4225 and 4075 analyses, respectively, but the DHPSCAO reaches a good solution in 5300 analyses. As it can be seen in the results of Table 4.16, the IMCSS algorithm achieves better



**Fig. 4.11** Convergence curves for the 10-bar planar truss structure using MCSS and IMCSS [2]. (a) Case 1 and (b) Case 2

optimal results than previous methods like MCSS, PSO, PSOPC, HPSO, and DHPSACO algorithms.

### 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  $0.1 \text{ lb/in}^3$  ( $2767.990 \text{ kg/m}^3$ ), and the modulus of elasticity is 10,000 ksi ( $68,950 \text{ MPa}$ ). The nodes are subjected to the displacement limits of  $\pm 0.25$  in ( $\pm 0.635 \text{ cm}$ ), and the members are subjected to the stress limits of  $\pm 25$  ksi ( $\pm 172.375 \text{ MPa}$ ).

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

Element group	Camp et al. [26]		Lee and Geem [27]		Li et al. [28]		Kaveh and Talatahari [29]		Present work [2]	
	GA	HS	PSO	PSOPC	HPSO	HPSACO	MCSS	IMCSS	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)	5076.31	5057.88	5529.5	5061	5060.92	5056.56	5086.9	5064.6		
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	8875	8475		

**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		HPSACO	MCSS
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)		4668.81	4679.47	4677.7	4677.29	4675.78	4686.47	4679.15
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	9625	7350	6625

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, (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  $D = \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2.0, 2.1, 2.2, 2.3, 2.4, 2.5, 2.6, 2.7, 2.8, 2.9, 3.0, 3.1, 3.2\}$  ( $\text{in}^2$ ) or  $\{0.65, 1.29, 1.94, 2.58, 3.23, 3.87, 4.52, 5.16, 5.81, 6.45, 7.10, 7.74, 8.39, 9.03, 9.68, 10.32, 10.97, 12.26, 12.90, 13.55, 14.19, 14.84, 15.48, 16.13, 16.77, 17.42, 18.06, 18.71, 19.36, 20.00, 20.65\}$  ( $\text{cm}^2$ ).

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 of 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 is 385.54 lb (174.88 kg), while it is 389.49 lb, 388.94 lb, 387.94 lb, and 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

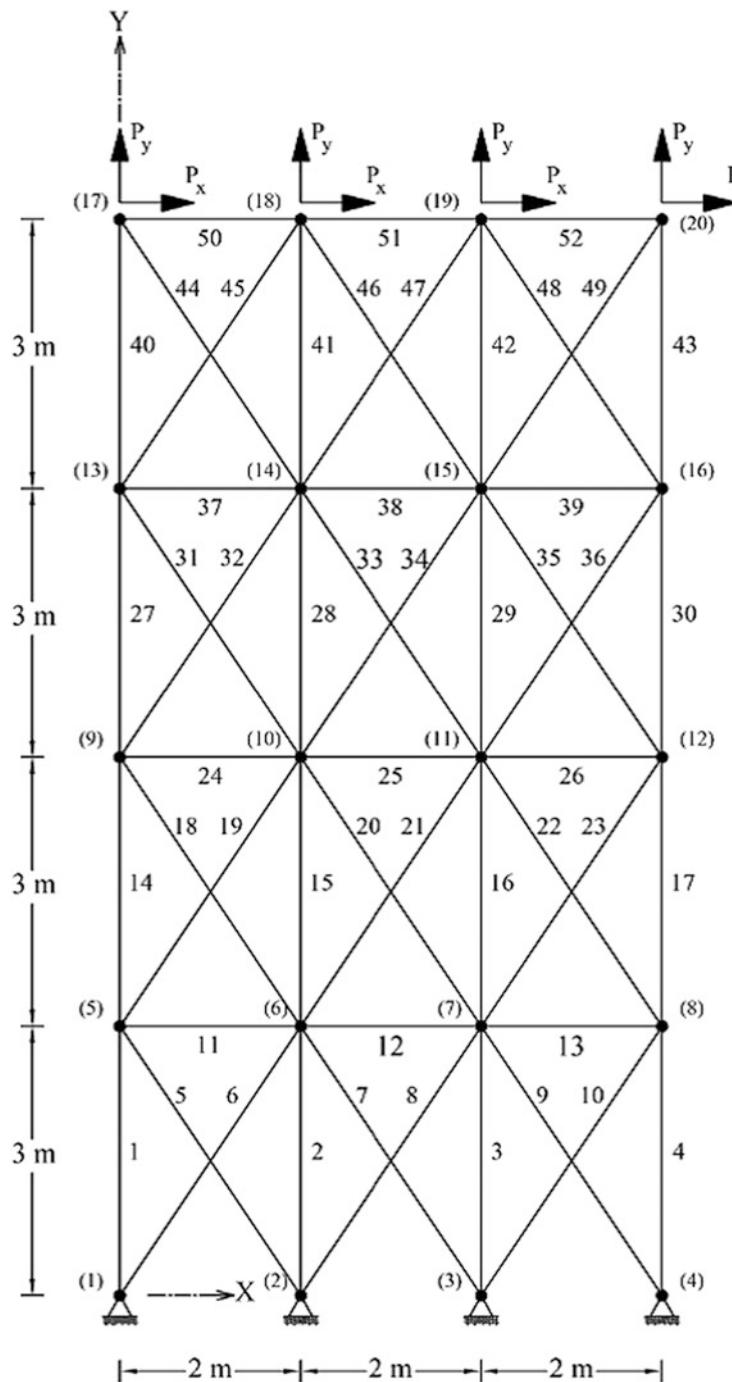
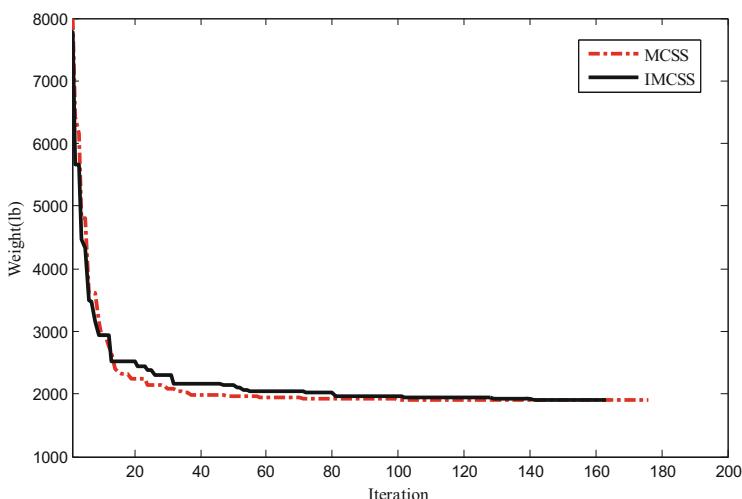


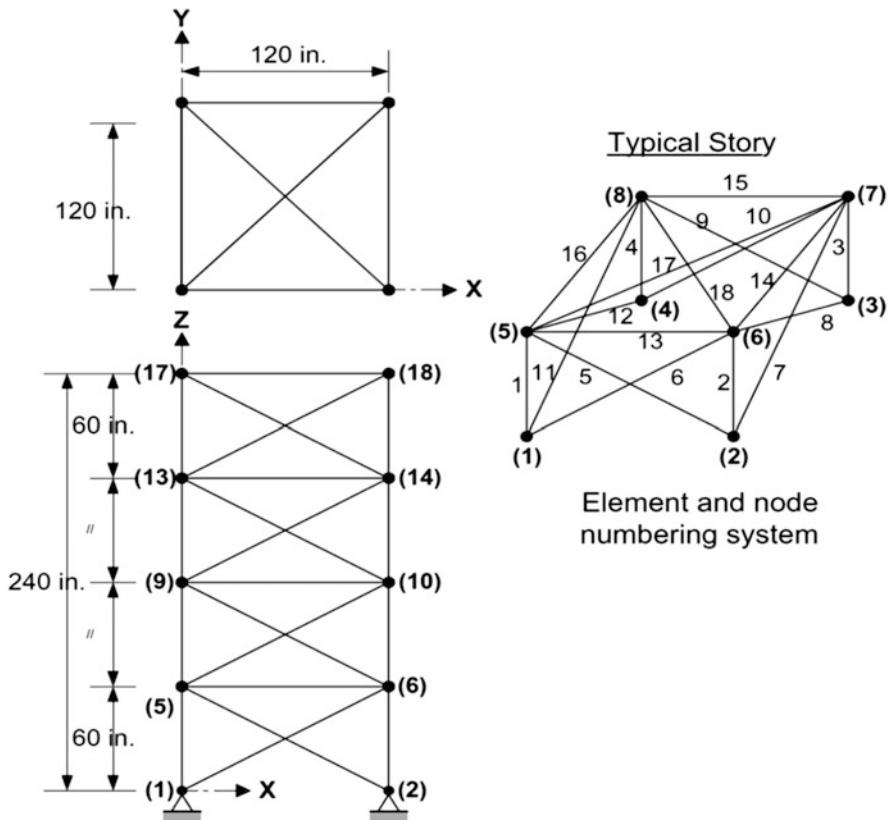
Fig. 4.12 Schematic of a 52-bar planar truss

**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	4658.055	4658.055	5999.988	4658.055	4658.055	4658.055	4658.055
2	1161.288	1374.19	1008.38	1161.288	1161.288	1161.288	1161.288
3	506.451	1858.06	2696.38	363.225	494.193	363.225	494.193
4	3303.219	3206.44	3206.44	3303.219	3303.219	3303.219	3303.219
5	940	1283.87	1161.29	940	1008.385	939.998	939.998
6	494.193	252.26	729.03	494.193	285.161	506.451	494.193
7	2290.318	3303.22	2238.71	2238.705	2290.318	2238.705	2238.705
8	1008.385	1045.16	1008.38	1008.385	1008.385	1008.385	1008.385
9	2290.318	126.45	494.19	388.386	388.386	388.386	494.193
10	1535.481	2341.93	1283.87	1283.868	1283.868	1283.868	1283.868
11	1045.159	1008.38	1161.29	1161.288	1161.288	1161.288	1161.288
12	506.451	1045.16	494.19	792.256	506.451	729.031	494.193
Weight (kg)	1906.76	2230.16	2146.63	1905.49	1904.83	1904.05	1902.61
No. of analyses	N/A	N/A	N/A	50,000	5300	4225	4075

**Fig. 4.13** Convergence curves for the 52-bar planar truss structure using MCSS and IMCSS [2]

reached. The IMCSS algorithm gets the best solution after 145 iterations (3625 analyses), while it takes for MCSS and DHPSACO 216 iterations (5400 analyses) and 213 iterations (5330 analyses), respectively.



**Fig. 4.14** Schematic of a 72-bar spatial truss

**Table 4.17** Loading conditions for the 72-bar spatial truss

Node	Case 1			Case 2		
	P <sub>X</sub> kips (kN)	P <sub>y</sub> kips (kN)	P <sub>z</sub> kips (kN)	P <sub>X</sub> kips (kN)	P <sub>y</sub> kips (kN)	P <sub>z</sub> 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)

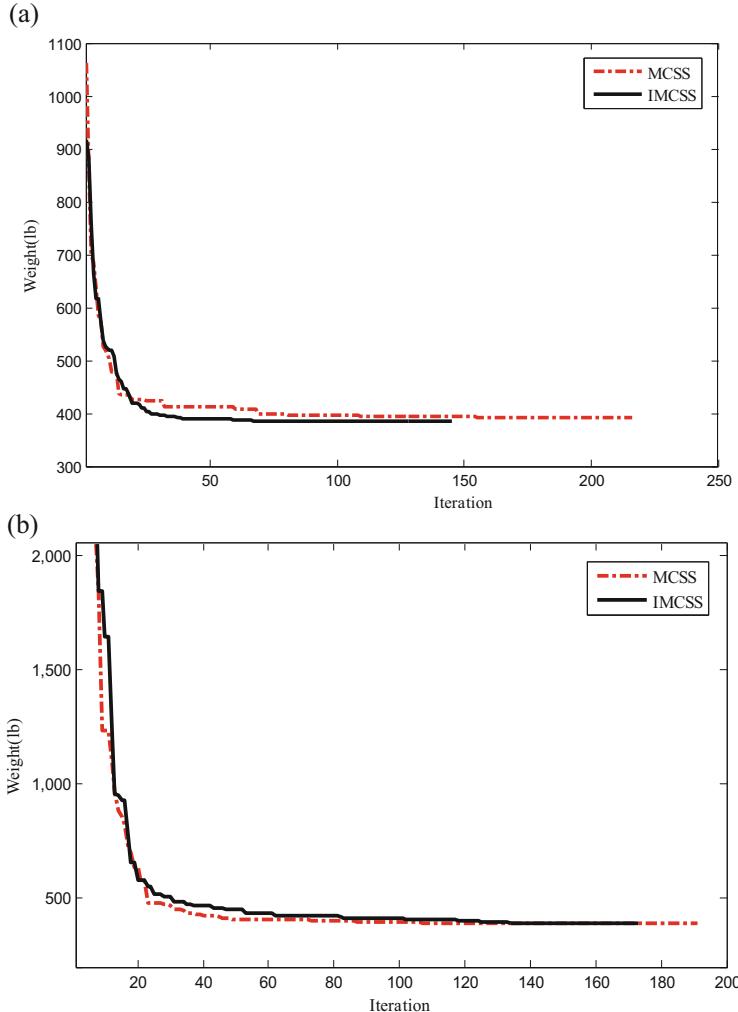
In Case 2, the best obtained weight from IMCSS is 389.60 lb, but it is 393.13 lb, 389.87 lb, 392.84 lb, 393.06 lb, and 393.38 lb for MCSS, CS, ICA, CSS, and HPSACO algorithms, respectively. IMCSS algorithm finds the best solutions after 173 iterations (4325 analyses), while MCSS, CS, ICA, CSS, and HPSACO algorithms need 4775, 4840, 4500, 7000, and 5330 analyses to find the best solutions.

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

Element group	GA	Wu and Chow [30]	Lee and Geem [27]	Li et al. [28]	PSOPC	HFSO	Kaveh and Talatahari [31]	MCSS	Present work [2]
		HS					DHPSACO		
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	1089.88	1069.79	388.94	385.54	389.49	385.54	385.54
No. of analyses	N/A	N/A	N/A	150,000	50,000	5330	5400	3625	3625

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

Element group	Wu and Chow [30]			Li et al. [28]			Kaveh and Talatahari			Kaveh and Bakhtpoori [33]			Present work [2]	
	GA	PSO	PSOPC	HPSO	DHPSACO [31]	CSS [23]	ICA [32]	CS	MCSS	MCSS	IMCSS	MCSS	IMCSS	
1 AA4	0.196	7.22	4.49	4.97	1.8	1.99	1.99	1.8	1.8	1.8	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	0.563	0.563	0.563	
3 A13~A16	0.307	1.13	0.111	0.141	0.111	0.111	0.111	0.111	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	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	1.266	1.266	1.266	1.228	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	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	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	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	0.563	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	0.442	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	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	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	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	0.602	0.602	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	0.391	0.391	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	0.563	0.563	0.563	0.766	0.563	
Weight (lb)	427.20	1209	941.82	933.09	393.38	393.06	392.84	389.87	389.87	389.87	389.87	393.13	389.6	
No. of analyses	N/A	150,000	50,000	5330	7000	4500	4840	4840	4840	4840	4840	4775	4325	

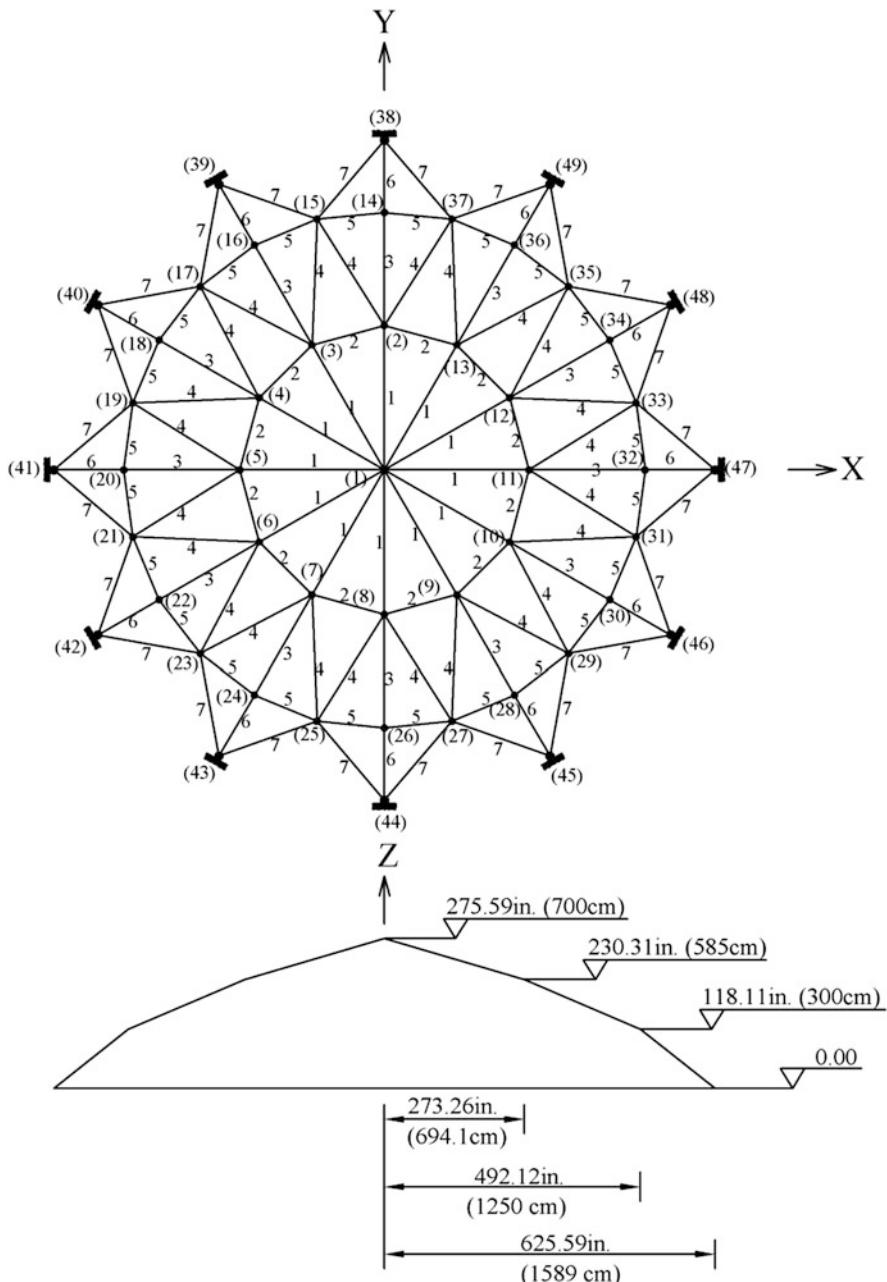


**Fig. 4.15** Convergence curves for the 72-bar truss structure using MCSS and IMCSS [2]. (a) Case 1 and (b) Case 2

#### 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<sup>3</sup> (7971.810 kg/m<sup>3</sup>). The yield stress of steel is taken as 58.0 ksi (400 MPa).



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

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

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

where  $\sigma_i^-$  is calculated according to the slenderness ratio:

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

where  $E$  is the modulus of elasticity,  $F_y$  is the yield stress of steel,  $C_c$  is the slenderness ratio ( $\lambda_i$ ) dividing the elastic and inelastic buckling regions ( $C_c = \sqrt{2\pi^2 E/F_y}$ ),  $\lambda_i$  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 chapter, 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,  $-6.744$  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 $^2$ ).

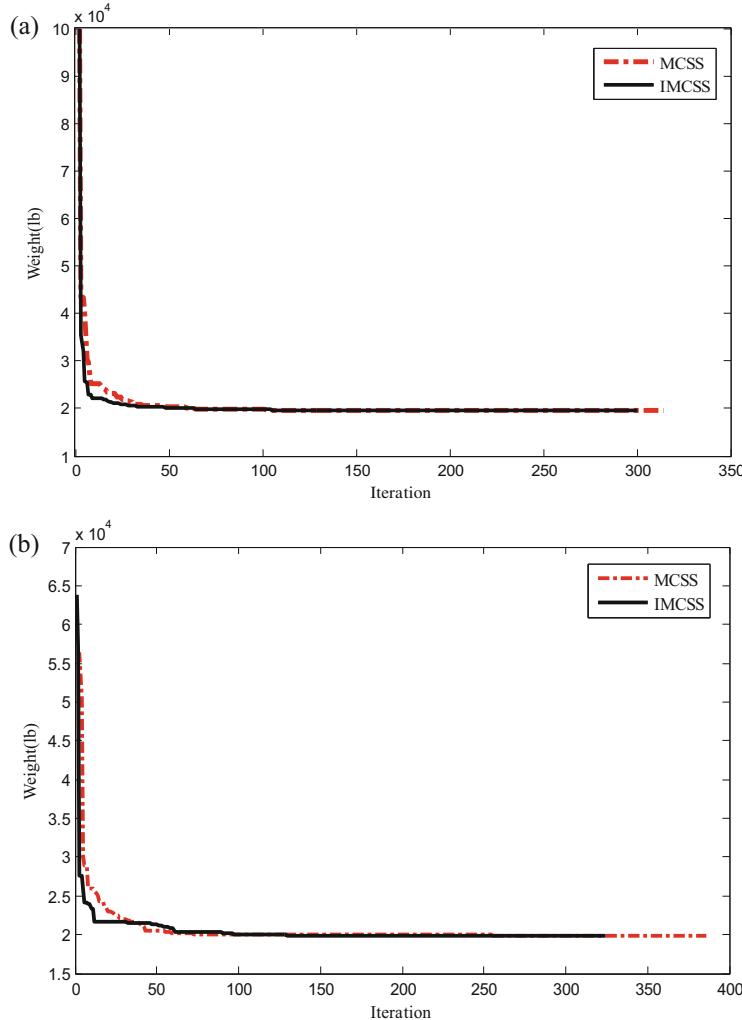
In this example, two cases of constraints are considered:

Case 1, with stress constraints and no displacement constraints, and Case 2, with stress constraints and displacement limitations of  $\pm 0.1969$  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 both 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 solutions in 314 iterations (7850 analyses) and 299 iterations (7475 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 (9650 analyses) and 324 iterations (8100 analyses) to find the best solutions, respectively, while for Ray and HPSACO algorithms, 19,950 and 10,075 analyses are required. The best weights obtained from MCSS and IMCSS algorithms are  $19,928$  lb and



**Fig. 4.17** Comparison of the convergence curves 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 comparing 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

**Table 4.20** Optimal design comparison for the 120-bar dome truss (two cases) optimal cross-sectional areas (in<sup>2</sup>)

Case 1						
Element group	Lee and Geem [27]			Kaveh and Khayatazar [37]		
	HS	PSO	PSOPC	HPSACO	Ray	Present work [2]
1	3.295	3.147	3.235	3.311	3.128	3.1108
2	3.396	6.376	3.37	3.438	3.357	3.3903
3	3.874	5.957	4.116	4.147	4.114	4.106
4	2.571	4.806	2.784	2.831	2.783	2.7757
5	1.15	0.775	0.777	0.775	0.775	0.9674
6	3.331	13.798	3.343	3.474	3.302	3.2981
7	2.784	2.452	2.454	2.551	2.453	2.4417
Weight (lb)	19,707.77	32,432.9	19,618.7	19,491.3	19,476.19	19,476.92
No. of analyses	35,000	N/A	125,000	10,025	19,950	7475
Case 2						
Element group	Lee and Geem [27]			Kaveh and Khayatazar [37]		
	HS	PSO	PSOPC	HPSACO	Ray	Present work
1	3.296	15.978	3.083	3.779	3.084	MCSS
2	2.789	9.599	3.639	3.377	3.360	IMCSS
3	3.872	7.467	4.095	4.125	4.093	3.3187
4	2.57	2.79	2.765	2.734	2.762	2.4746
5	1.149	4.324	1.776	1.609	1.593	4.2768
6	3.331	3.294	3.779	3.533	3.294	4.2882
7	2.781	2.479	2.438	2.539	2.434	2.8103
Weight (lb)	19,893.34	41,052.7	20,681.7	20,078	20,071.9	0.7753
No. of analyses	35,000	N/A	125,000	10,075	19,950	3.5168

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.

## 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 Multidiscip 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 Heuristics* 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 Intel* 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 Ser 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, Kissimmee, 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, Chicago, IL
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 Constr 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. *Iran 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* 116:2599–2620
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, magnetic, or electric 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 abovementioned metaheuristics. It seems the enhanced method opens a new horizon for the concept of time or iteration for the metaheuristics.

In order to investigate the efficiency 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 number 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.

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 an 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 ( $\mathbf{G}$ ) in addition to the sizing variables for its different members ( $\mathbf{A}$ ). The problem can be expressed as follows:

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

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);  $\rho_i$  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;  $\sigma_i$  and  $\delta_i$  are the stress and nodal deflection, respectively; and  $\sigma_i^b$  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 fields in physics such as temperature fields, air pressure fields, Newtonian gravitational fields, electric fields, 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:

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

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 monopole strength. According to Eq. (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 Eq. (5.2) as:

$$\begin{aligned} E_{ij} &= G \frac{m_i}{r_{ij}^2} \\ E_{ij} &= k_e \frac{q_i}{r_{ij}^2} \\ E_{ij} &= U \frac{M_i}{r_{ij}^2} \end{aligned} \quad (5.3)$$

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 Gauss's law as:

$$E_{ij} = k_e \frac{q_i}{a^3} r_{ij} \quad (5.4)$$

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

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

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

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

$$E_j = \sum_{i=1, i \neq j}^N E_{ij} \frac{\mathbf{r}_{ij}}{||\mathbf{r}_{ij}||} \quad (5.6)$$

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

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

## 5.4 Necessary Definitions for a FOF-Based Model

Here, some principles 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 sometimes 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 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 an 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 An 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 in the space-time and finds a place under the effect of the 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 can not make a problem in generality of this pseudo-code.

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

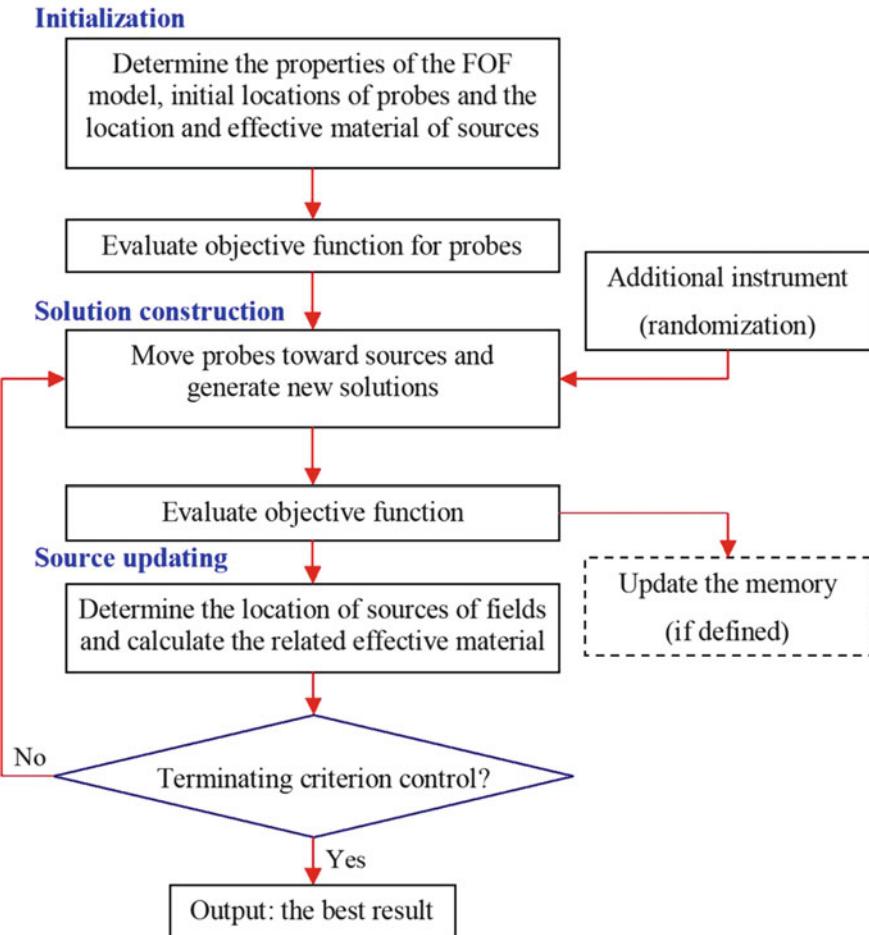


Fig. 5.1 Flowchart of the FOF-based model [1]

## 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.

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

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

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

$$r_{ij} = \frac{\|\mathbf{X}_i - \mathbf{X}_j\|}{\|(\mathbf{X}_i + \mathbf{X}_j)/2 - \mathbf{X}_{best}\| + \epsilon} \quad (5.9)$$

where  $\mathbf{X}_{best}$  is the position of the best current CP and  $\epsilon$  is a small positive number.

In Eq. (5.8),  $p_{ij}$  is the probability of moving each CP toward the others and is equal to

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

- New position creation. Each CP moves to the new position and the velocities are found as:

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

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

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 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.

- Force determination. The force vector for the  $j$ th CP is calculated as Eq. (5.8).
- New position creation. Each CP moves to the new position as defined in Eqs. (5.11) and (5.12). It should be noted that in order to determine the location of each CP using Eq. (5.11), the recent location of the previous agents is utilized instead of the previous ones, and this leads to the use of the previous 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 size 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 analyze the structures.

### 5.7.1 18-Bar Planar Truss

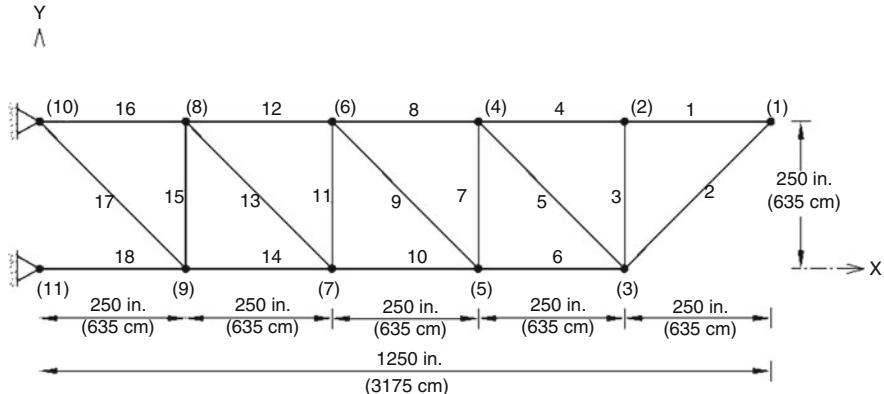
The initial configuration of an 18-bar cantilever planar truss is shown in Fig. 5.2 which has been previously analyzed by many authors to obtain the optimal design. The material density is  $2768 \text{ kg/m}^3$  ( $0.1 \text{ lb/in.}^3$ ) and the modulus of elasticity is  $68,950 \text{ MPa}$  ( $10,000 \text{ ksi}$ ). The members are subjected to stress limitations of  $\pm 137.9 \text{ MPa}$  ( $\pm 20 \text{ ksi}$ ). Also, an Euler buckling compressive stress limitation is imposed for truss member  $i$ , according to:

$$\sigma_i^b = \frac{-kEA_i}{L_i^2} \quad (5.13)$$

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  $-89 \text{ N}$  ( $-20 \text{ kips}$ ) 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)  $A_3, A_7, A_{11}, A_{15}$
- (4)  $A_5, A_9, A_{13}, A_{17}$



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

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:

$$\begin{aligned} -571.5 \text{ cm} (-225 \text{ in.}) &\leq y_3, y_5, y_7, y_9 \leq 622.3 \text{ cm} (245 \text{ in.}); \\ 1968.5 \text{ cm} (775 \text{ in.}) &\leq x_3 \leq 3111.5 \text{ cm} (1225 \text{ in.}); \\ 1333.5 \text{ cm} (525 \text{ in.}) &\leq x_5 \leq 2476.5 \text{ cm} (975 \text{ in.}); \\ 698.5 \text{ cm} (275 \text{ in.}) &\leq x_7 \leq 1841.5 \text{ cm} (725 \text{ in.}); \\ 63.5 \text{ cm} (25 \text{ in.}) &\leq x_9 \leq 1206.5 \text{ cm} (475 \text{ in.}). \end{aligned}$$

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

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 8000 structural analyses and the HS algorithm (Lee and Geem [11]) converges to the optimum point after 24,805 analyses, while CSS needs 4000 FEM analyses to reach the result. Figure 5.3 shows the convergence curve 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 indicate the limits of the stress

**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 <sub>1</sub>	11.24	12.554	12.040	12.65	12.476
A <sub>2</sub>	15.68	18.029	17.847	7.22	17.831
A <sub>3</sub>	7.93	5.114	7.969	6.17	5.277
A <sub>5</sub>	6.49	3.571	4.726	3.55	3.726
x <sub>3</sub>	891.10	912.969	944.882	903.10	911.698
y <sub>3</sub>	143.60	188.067	150.000	174.30	185.788
x <sub>5</sub>	608.20	646.450	664.961	630.30	643.917
y <sub>5</sub>	105.40	150.617	122.441	136.30	147.640
x <sub>7</sub>	381.70	416.624	414.961	402.10	414.18
y <sub>7</sub>	57.10	102.526	77.559	90.50	98.507
x <sub>9</sub>	181.00	204.282	192.520	195.30	202.444
y <sub>9</sub>	-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

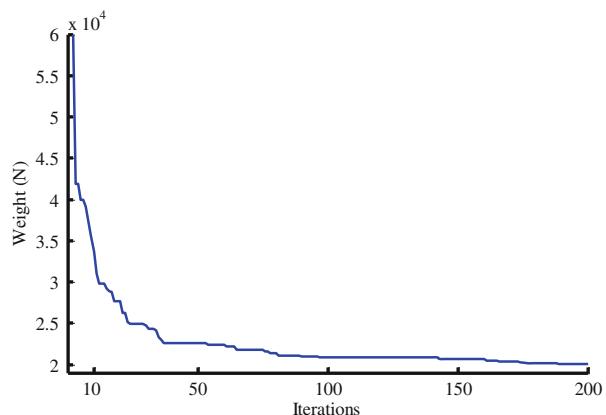
constraints, and it is equal to 20.00 ksi when the type of stress is tension, and the limits for elements in compression are obtained by using Eq. (5.13). The maximum tension stress is equal to 20.00 ksi in the 16th element, and the maximum compression stress is  $-17.0152$  ksi in the last element, while the allowable buckling stress equals to  $-17.0154$  ksi.

### 5.7.2 25-Bar Spatial Truss

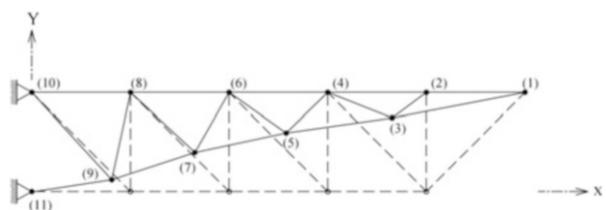
Figure 5.6 shows the initial topology of a 25-bar spatial truss structure. This example has been frequently studied in sizing and configuration optimization using mathematical approaches. The material density is  $2768 \text{ kg/m}^3$  ( $0.1 \text{ lb/in.}^3$ ), and the modulus of elasticity is 68,950 MPa (10,000 ksi). Two cases are considered:

- Case 1. The cross sections are continuous, and the bounds on the member cross-sectional areas are  $0.065\text{--}6.45 \text{ cm}^2$  ( $0.01\text{--}1.0 \text{ in.}^2$ ). 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 Eq. (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 follows:  $\{0.645I \ (I=1,\dots,26), 18.064, 19.355, 20.645, 21.935\} \text{ cm}^2$  or  $\{0.1I \ (I=1,\dots,26), 2.8, 3.0, 3.2, 3.4\} \text{ in.}^2$

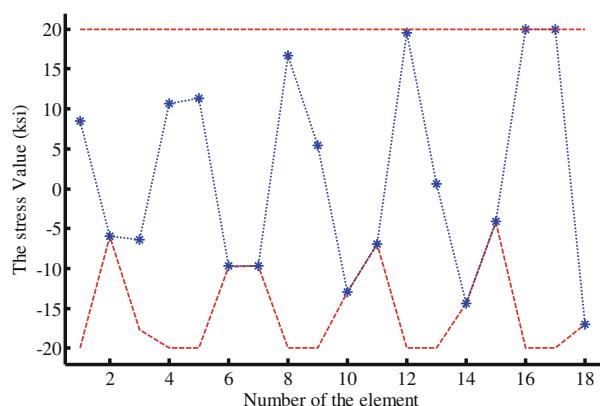
**Fig. 5.3** Convergence curve for the 18-bar truss for the CSS algorithm [1]



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

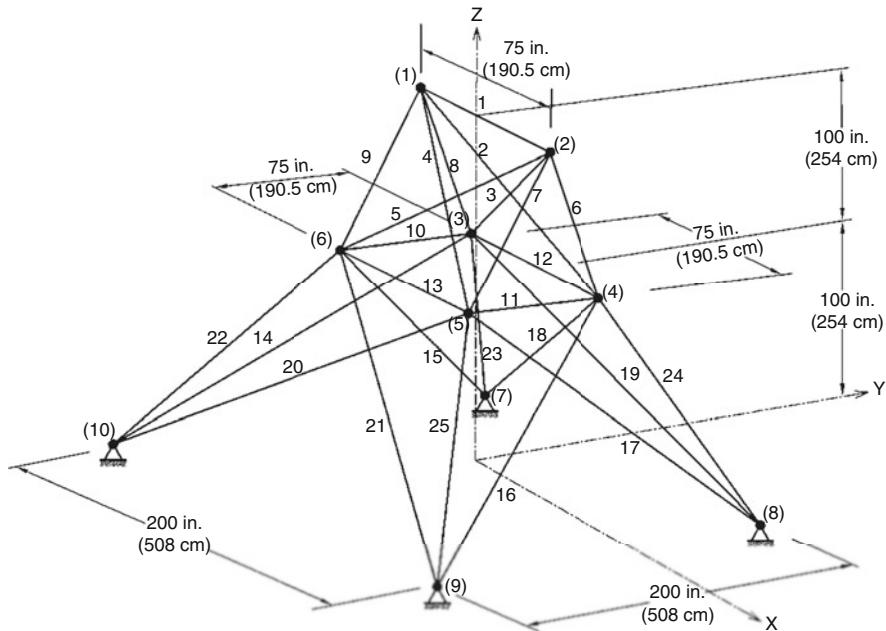


**Fig. 5.5** Comparison of the allowable and existing stress values for the 18-bar truss using the CSS algorithm [1]



which has 30 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  $\pm 275.6$  MPa or  $\pm 40$  ksi) for all members.

The structure was required to be doubly symmetric about  $x$ - and  $y$ -axes; this condition grouped the truss members as follows: (1)  $A_1$ , (2)  $A_2 \sim A_5$ , (3)  $A_6 \sim A_9$ ,



**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)

Node	Case 1			Case 2		
	P <sub>X</sub> kips (kN)	P <sub>Y</sub> kips (kN)	P <sub>Z</sub> kips (kN)	P <sub>X</sub> kips (kN)	P <sub>Y</sub> kips (kN)	P <sub>Z</sub> 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 <sub>X</sub> kips (kN)	P <sub>Y</sub> kips (kN)	P <sub>Z</sub> 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

(4)  $A_{10} \sim A_{11}$ , (5)  $A_{12} \sim A_{13}$ , (6)  $A_{14} \sim A_{17}$ , (7)  $A_{18} \sim A_{21}$ , and (8)  $A_{22} \sim A_{23}$ . 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:

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

Considering Case 1, this example was solved by different methods. Vanderplaats and Moses [12] and Felix [8] 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 weights of these methods are 593.8, 571.6, 610.3, 590.9, 584.0, and 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 needs 4000 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 weights 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  $-126.4 \text{ MPa}$  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.

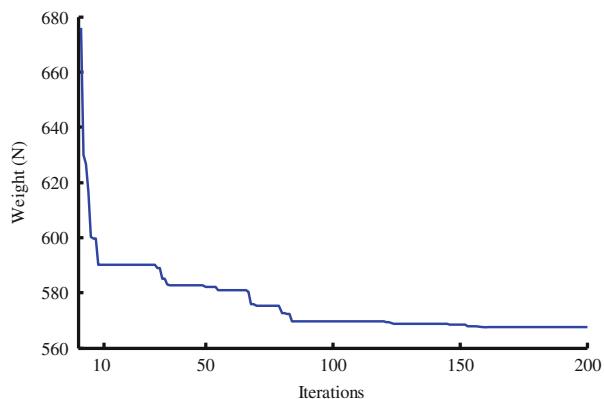
In addition to previous cases, when the range of cross-sectional areas varies from 0.01 to  $3.4 \text{ in}^2$  ( $0.6452\text{--}21.94 \text{ cm}^2$ ) and only size optimization is considered, a statistical study on the results of different algorithms is performed. The detailed information for constraint 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, and the standard deviation are compared.

### 5.7.3 120-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 original 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  $7971.810 \text{ kg/m}^3$  ( $0.288 \text{ lb/in.}^3$ ). 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  $-60 \text{ kN}$  ( $-13.49 \text{ kips}$ ) at node

**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 <sub>1</sub>	0.08	0.07	0.58	0.58	0.11
A <sub>2</sub>	2.67	3.14	2.84	4.97	2.33
A <sub>3</sub>	5.43	5.39	5.81	4.39	5.64
A <sub>4</sub>	0.21	0.16	0.32	0.52	0.25
A <sub>5</sub>	0.65	0.79	0.71	0.065	0.70
A <sub>6</sub>	0.78	0.54	1.36	0.1	0.80
A <sub>7</sub>	4.77	4.50	4.52	3.1	5.34
A <sub>8</sub>	3.57	3.54	3.61	3.1	3.69
x <sub>4</sub>	54.6	60.20	55.8	32.0	51.57
y <sub>4</sub>	122.7	125.2	110.7	222.0	96.00
z <sub>4</sub>	254.8	248.2	246.0	254.0	262.77
x <sub>8</sub>	54.1	69.9	35.9	159.0	45.65
y <sub>8</sub>	244.7	244.9	206.1	254.0	198.71
Best weight (N)	593.8	571.6	590.9	583.8	567.5

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

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<sup>2</sup> (0.775 in.<sup>2</sup>), and the maximum cross-sectional area is taken as 129.03 cm<sup>2</sup> (20.0 in.<sup>2</sup>). 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 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 original CSS and enhanced CSS. When pure size optimization is considered, the enhanced CSS can find a better result in a less

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

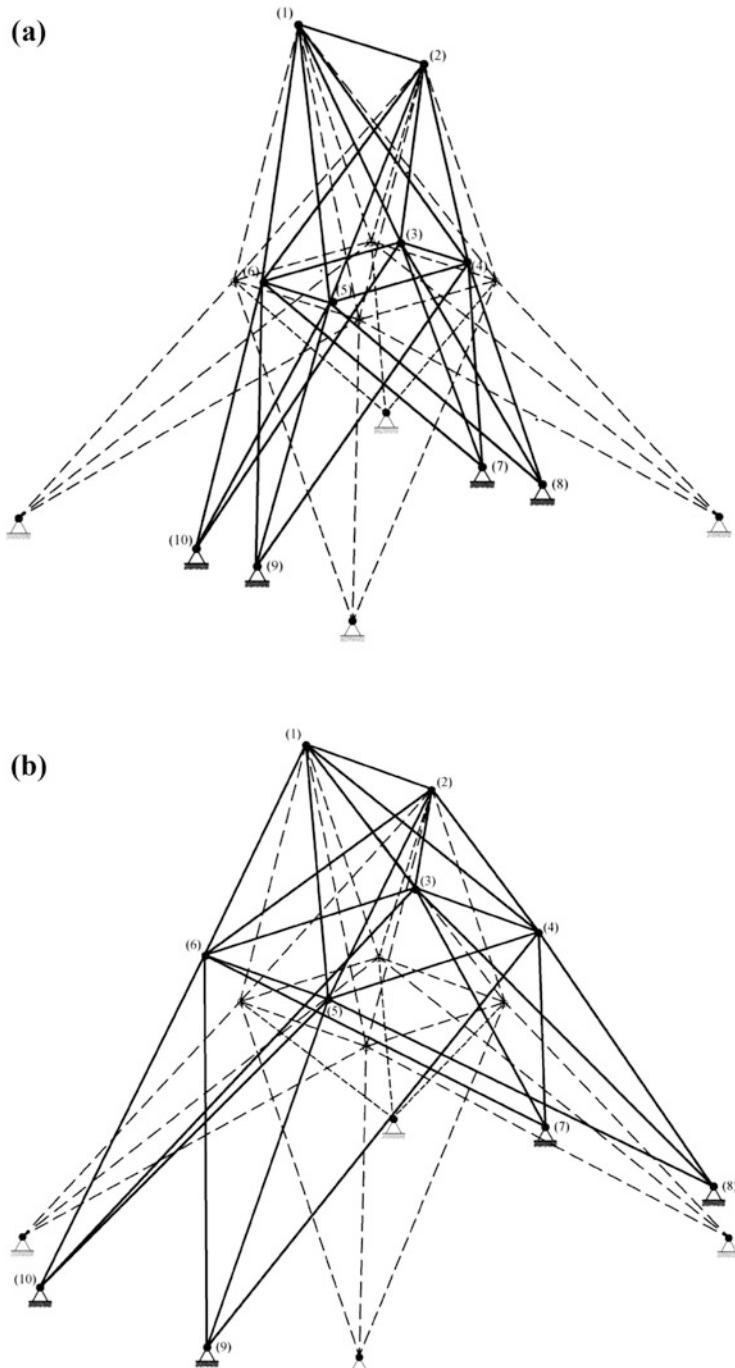
Design variables	Wu and Chow [16]	Kaveh and Kalatjari [17]	Rahami et al. [9]	Lee and Geem [11]	Kaveh and Talatahari [1]
A <sub>1</sub>	0.1	0.1	0.1	0.2	0.1
A <sub>2</sub>	0.2	0.1	0.1	0.1	0.1
A <sub>3</sub>	1.1	1.1	1.1	0.9	0.9
A <sub>4</sub>	0.2	0.1	0.1	0.1	0.1
A <sub>5</sub>	0.3	0.1	0.1	0.1	0.1
A <sub>6</sub>	0.1	0.1	0.1	0.1	0.1
A <sub>7</sub>	0.2	0.1	0.2	0.2	0.1
A <sub>8</sub>	0.9	1.0	0.8	1.0	1.0
x <sub>4</sub>	41.07	36.23	33.049	31.88	36.762
y <sub>4</sub>	53.47	58.56	53.566	83.57	56.920
z <sub>4</sub>	124.6	115.59	129.909	126.35	124.863
x <sub>8</sub>	50.8	46.46	43.783	40.43	49.767
y <sub>8</sub>	131.48	127.95	136.838	130.64	136.757
Best weight (N)	605.85	551.58	534.30	550.55	528.58

number of analyses. The enhanced CSS needs 4000 analyses to find the optimum result, while it is 7000 for the original CSS as reported by the authors (Kaveh and Talatahari [5]). Figure 5.10 shows the convergence curves 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 original 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:

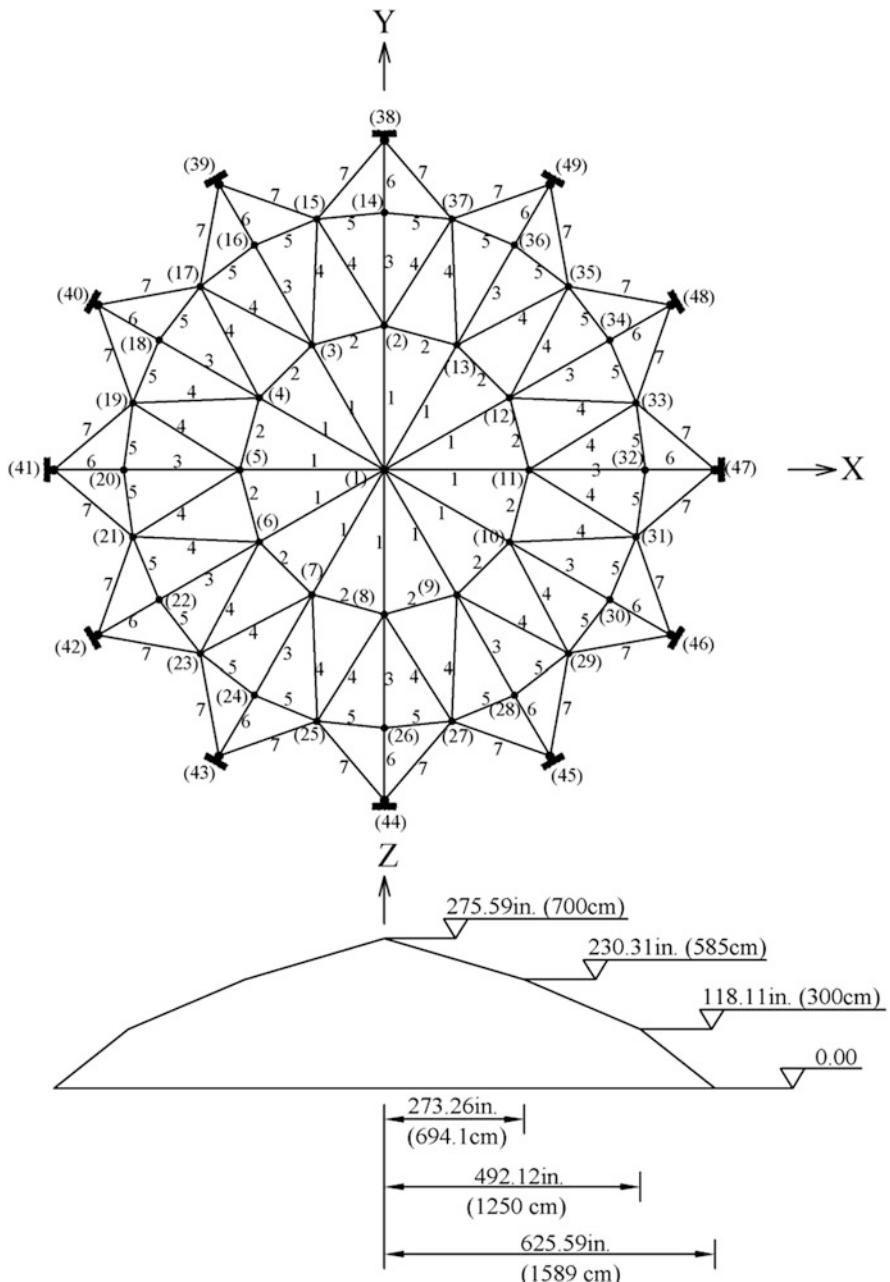
- Initialization, where one considers some probes and sources and determines the first location of the probes (initial solutions) using a uniform field. Also, the amounts of the effective material for the sources based on the utilized approach are determined.
- Solution construction, in which each probe 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.



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

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

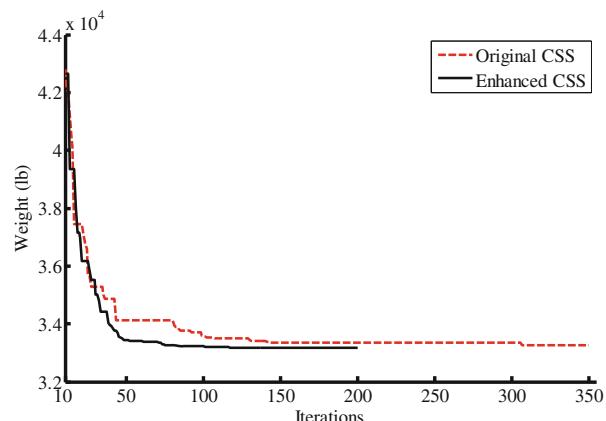
	Rajeev and Krishnamoorthy	Schutte and Groenwold	Lee and Geem	Kaveh and Talatahari				
	GA [17]	PSO [18]	HS [10]	PSACO [19]	HPSACO [19]	HBB-BC [20]	CSS [4]	Kaveh and Talatahari [1]
Best weight (lb)	546	545.21	544.38	545.04	544.99	545.16	545.10	544.92
Average weight (lb)	N/A	546.84	N/A	N/A	545.52	545.66	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

**Table 5.7** Performance comparison for the 120-bar truss

Design variables	Pure size optimization		Configuration optimization	
	Original CSS (Kaveh and Talatahari [5])	Present work [1]	Original CSS	Kaveh and Talatahari [1]
A <sub>1</sub>	3.027	3.032	3.103	3.235
A <sub>2</sub>	14.606	15.335	7.328	4.875
A <sub>3</sub>	5.044	4.767	4.350	4.303
A <sub>4</sub>	3.139	3.030	2.731	2.764
A <sub>5</sub>	8.543	8.252	1.719	2.438
A <sub>6</sub>	3.367	3.723	3.739	3.637
A <sub>7</sub>	2.497	2.502	2.452	2.505
z <sub>1</sub>	–	–	286.505	259.569
z <sub>2</sub>	–	–	209.295	207.017
z <sub>3</sub>	–	–	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)	2963	2456	3365	2884

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

- Terminating criterion control, which determines the end time of the search process.

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 of 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 original 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 spacetime for other metaheuristic algorithms opens a new horizon for the concept of the time or iteration for metaheuristics, and it is expected that this continuous space–time can be extended to other algorithms. As a future work, one can change the manner of using other concepts of the FOF model to reach 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* 31:452–461
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 optimization, edit. Z.W. Geem. Springer, Berlin, Heidelberg
21. American Institute of Steel Construction (AISC) (1989) Manual of steel construction—allowable stress design, 9th edn. AISC, Chicago, IL

# Chapter 6

## Dolphin Echolocation Optimization

### 6.1 Introduction

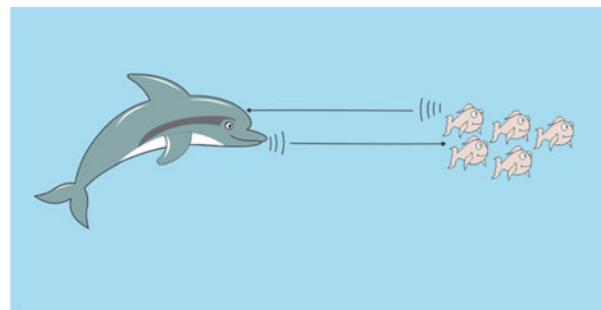
Nature has provided inspiration for most of the man-made technologies. Scientists believe that dolphins are the second to humans in smartness and intelligence. Echolocation is the biological sonar used by dolphins and several kinds of other animals for navigation and hunting in various environments. 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 (DE) 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 changes 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

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



and a few birds. The best studied echolocation in marine mammals is that of the bottlenose dolphins, (Au [3]).

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 toward 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 enables 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 more 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 Ref. [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 produced 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 metaheuristics [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, the search space should be sorted using the following rule:

**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 %
<i>CF</i>	55 %			

**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 to each other, as the columns of a matrix, the *Matrix Alternatives*<sub>MA\*NV</sub> 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:

$$PP(Loop_i) = PP_1 + (1 - PP_1) \frac{Loop_i^{Power} - 1}{(LoopsNumber)^{Power} - 1} \quad (6.1)$$

*PP*: Predefined probability

*PP*<sub>1</sub>: Convergence factor of the first loop in which the answers are selected randomly

*Loop<sub>i</sub>*: Number of the current loop

*Power*: Degree of the curve. As it can be seen, the curve in Eq. (6.1) is a polynomial 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.

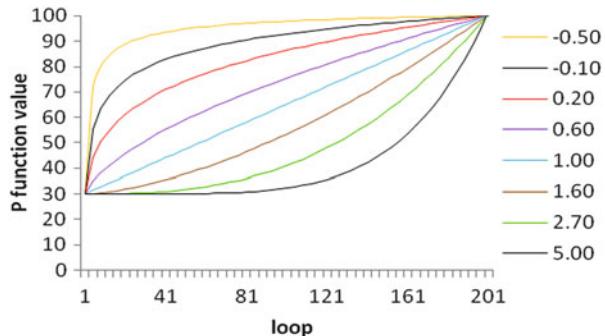
Figure 6.2 shows the variation of *PP* by the changes of the *Power*, using the proposed formula, Eq. (6.1).

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

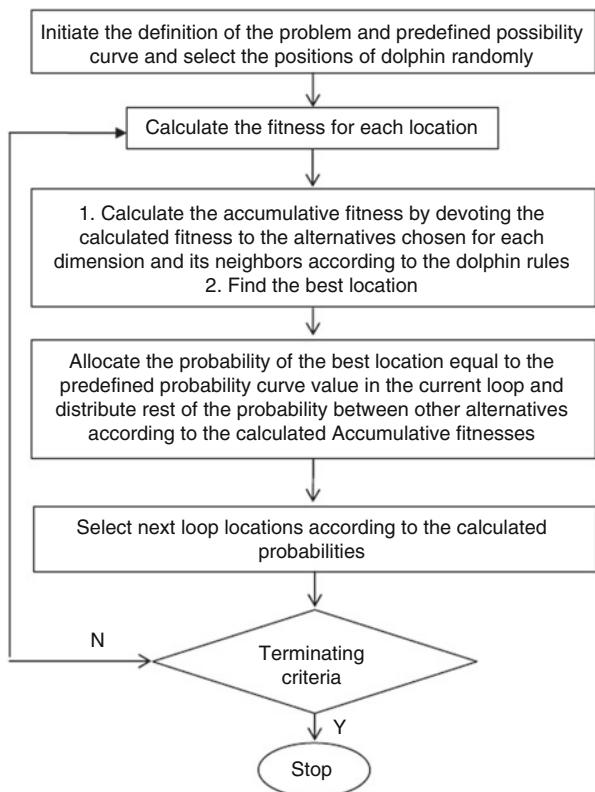
1. Initiate *NL* locations for a dolphin randomly.

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).

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



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



2. Calculate the  $PP$  of the loop using Eq. (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.

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  $R_e$

$$AF_{(A+k)j} = \frac{1}{R_e} * (R_e - |k|) Fitness(i) + AF_{(A+k)j} \quad (6.2)$$

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 Alternatives matrix) to be chosen for the  $j$ th variable and  $R_e$  is the effective radius in which accumulative fitness of the alternative  $A$ 's neighbors is affected from its fitness. This radius is recommended to be not more than 1/4 of the search space;  $Fitness(i)$  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  $A + k > LA_j$ ), 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  $\epsilon$  is added to all the arrays as  $AF = AF + \epsilon$ . Here,  $\epsilon$  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( $j$ )

$$AF_{ij} = 0 \quad (6.3)$$

end

end

end

5. For variable  $j_{(j=1 \text{ to } NV)}$ , calculate the probability of choosing alternative  $i_{(i=1 \text{ to } AL_j)}$ , according to the following relationship:

$$P_{ij} = \frac{AF_{ij}}{\sum_{i=1}^{LA_j} AF_{ij}} \quad (6.4)$$

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)

$$P_{ij} = PP \quad (6.5)$$

Else

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

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 quits 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 abovementioned 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  $\frac{1}{4}$  of the size of the search space.

(d)  $\epsilon$

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:

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

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 Eq. (6.1), utilizing  $Power = 1$ , the loops number = 8, and  $PPI = 0.1$  as follows:

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

It should be noted that the  $PPI$  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 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 Eq. (6.8).
3. Calculate fitness for each location. In this example as the objective function is defined by Eq. (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 fitness values 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 Eq. (6.2). As discussed before the alternatives should be sorted in an ascending order. The  $Alternatives_{MA \times NV}$  ( $MA$  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:

$$Alternatives = \begin{bmatrix} -20 & -20 & -20 & -20 \\ -19 & -19 & -19 & -19 \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ 19 & 19 & 19 & 19 \\ 20 & 20 & 20 & 20 \end{bmatrix} \quad (6.9)$$

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  $j$ th column of the Alternatives matrix and name it as  $A$ .

for  $k = -10$  to 10

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

end

end

end

Equation (6.10) can also be stated as:

for  $j = \{1,2,3,4\}$

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

for  $k = -10$  to 10

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

end

end

Considering  $\epsilon$  as the worth possible fitness, it will be  $\epsilon = 1/(4 * 20^2)$  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, the search space should be 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:

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

$$AF = AF + 0.000625$$

Figure 6.4 shows the result of performing the explained process for all four variables of this location.

Performing Step 4 for all the randomly selected answers, the final accumulative fitness of the first loop is achieved.

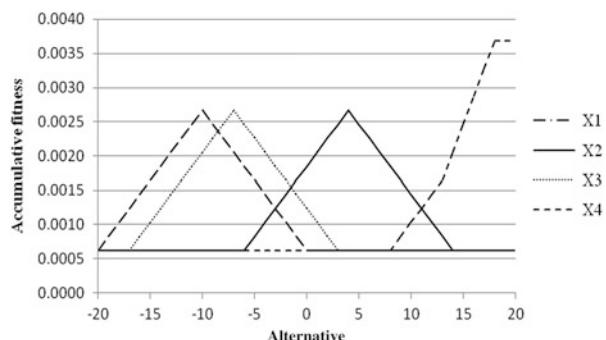
5. For variable  $j_{(j=1 \text{ to } 4)}$ , calculate the probability of choosing alternative  $i_{(i=1 \text{ to } 40)}$ , according to the following relationship:

$$P_{ij} = \frac{AF_{ij}}{\sum_{i=1}^{40} AF_{ij}} \quad (6.13)$$

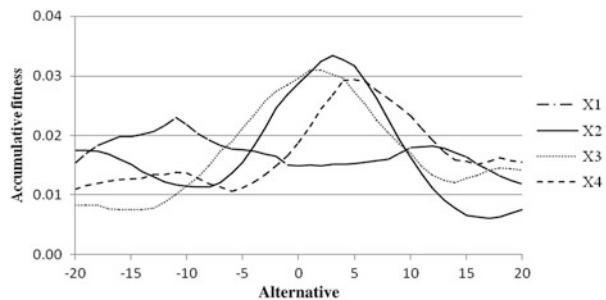
and consequently the probability will be according to Figs. 6.5 and 6.6.

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  $X1 = -11$ ,  $X2 = 3$ ,  $X3 = 4$ , and  $X4 = 4$ . On the other hand, according to Eq. (6.8), PP for the first loop is equal to 10 %. As a result all variables in

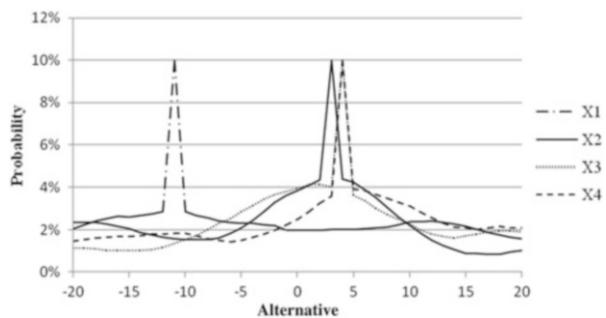
**Fig. 6.4** Accumulative fitness resulted from sample location of the mathematical example [1]



**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]



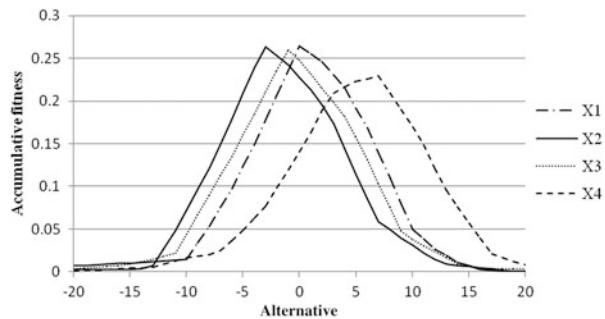
their best placement are equal to 10 % probability of the other alternatives is defined distributing remaining value of probability equal to 90 % to the other alternatives, using the following formula:

$$P_{ij} = (1 - 0.1)P_{ij} = 0.9P_{ij} \quad (6.14)$$

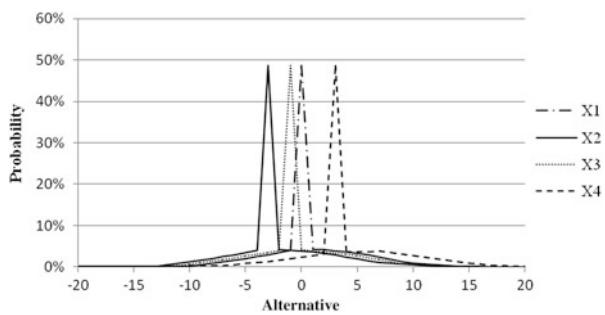
Since the number of loops is equal to 8, Steps 2–6 should be repeated eight times. Figures 6.7, 6.8, 6.9, and 6.10 show the accumulative fitness and the probability of alternatives in loops 4 and 8, respectively. It can be seen from these figures that the probability changes in a way that in eight loops DE reaches the best answer.

**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 probabilities.

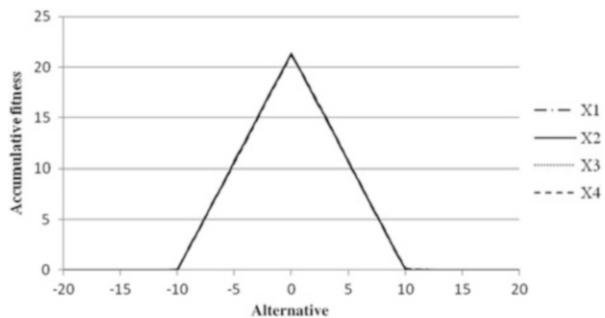
**Fig. 6.7** Accumulative fitness of all four variables in the fourth loop of DE in of mathematical example [1]



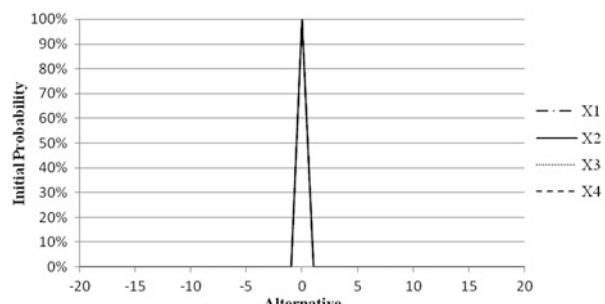
**Fig. 6.8** Probability curve of all four variables in the fourth loop of DE in mathematical example



**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]



## 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:*

$$W = \rho \sum_{i=1}^M A_i L_i \quad (6.15)$$

*Subjected to:*

$$\begin{aligned} KU - P &= 0 \\ g_1 \geq 0, g_2 \geq 0, \dots, g_n \geq 0 \end{aligned} \quad (6.16)$$

where  $g_1, g_2, \dots, g_n$  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:

$$F = F_0 - w^* (1 + K_p \cdot V) \quad (6.17)$$

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 algorithms.

### 6.5.1.1 A 25-Bar Spatial Truss

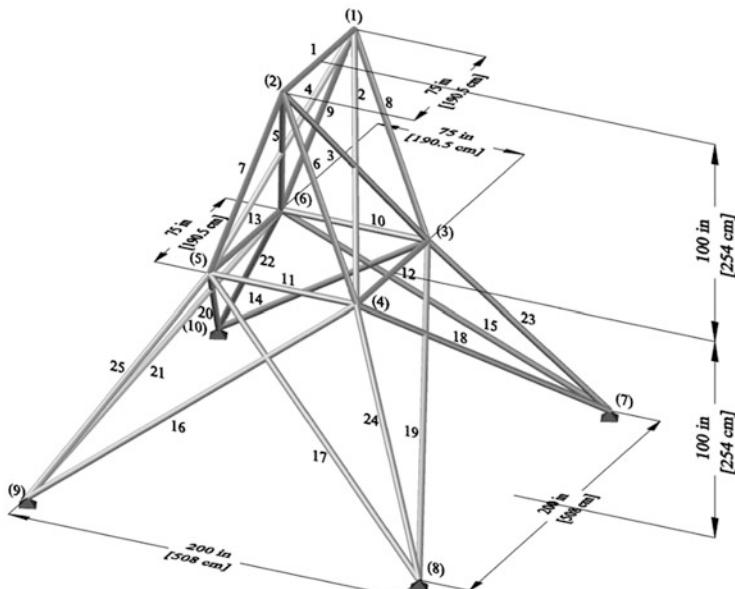
The 25-bar spatial truss structure shown in Fig. 6.11 has been studied in [8–11]. The material density is  $0.1 \text{ lb/in}^3$  ( $2767.990 \text{ kg/m}^3$ ), and the modulus of elasticity is 10,000 ksi (68,950 MPa). The stress limitations of the members are  $\pm 40 \text{ kpsi}$  ( $\pm 275.80 \text{ MPa}$ ). All nodes in three directions are subjected to displacement limitations of  $\pm 0.35 \text{ inch}$  (in) ( $\pm 8.89 \text{ mm}$ ) imposed on every node in each direction. The structure includes 25 members, which are divided into eight groups as follows: (1) A<sub>1</sub>, (2) A<sub>2</sub>–A<sub>5</sub>, (3) A<sub>6</sub>–A<sub>9</sub>, (4) A<sub>10</sub>–A<sub>11</sub>, (5) A<sub>12</sub>–A<sub>13</sub>, (6) A<sub>14</sub>–A<sub>17</sub>, (7) A<sub>18</sub>–A<sub>21</sub>, and (8) A<sub>22</sub>–A<sub>25</sub>. Two optimization cases are implemented.

**Case 1:** The discrete variables are selected from the set  $D = \{0.01, 0.4, 0.8, 1.2, 1.6, 2.0, 2.4, 2.8, 3.2, 3.6, 4.0, 4.4, 4.8, 5.2, 5.6, 6.0\}$  ( $\text{in}^2$ ) or  $\{0.065, 2.58, 5.16, 7.74, 10.32, 12.90, 15.48, 18.06, 20.65, 23.22, 25.81, 28.39, 30.97, 33.55, 36.13, 38.71\}$  ( $\text{cm}^2$ ).

**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.

For solving this problem by the use of DE, the loops number is set to 80. Convergence curve is according to Eq. (6.1) considering  $PP_I = 0.15$  and  $Power = 1$ .  $R_e$  and  $\epsilon$  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 with GA, PSO, PSOPC, and HPSO.



**Fig. 6.11** Schematic of a 25-bar spatial truss

**Table 6.2** The available cross-sectional areas of the AISC code

No.	in <sup>2</sup>	mm <sup>2</sup>	No.	in <sup>2</sup>	mm <sup>2</sup>
1	0.111	(71.613)	33	3.840	(2477.414)
2	0.141	(90.968)	34	3.870	(2496.769)
3	0.196	(126.451)	35	3.880	(2503.221)
4	0.250	(161.290)	36	4.180	(2696.769)
5	0.307	(198.064)	37	4.220	(2722.575)
6	0.391	(252.258)	38	4.490	(2896.768)
7	0.442	(285.161)	39	4.590	(2961.284)
8	0.563	(363.225)	40	4.800	(3096.768)
9	0.602	(388.386)	41	4.970	(3206.445)
10	0.766	(494.193)	42	5.120	(3303.219)
11	0.785	(506.451)	43	5.740	(3703.218)
12	0.994	(641.289)	44	7.220	(4658.055)
13	1.000	(645.160)	45	7.970	(5141.925)
14	1.228	(792.256)	46	8.530	(5503.215)
15	1.266	(816.773)	47	9.300	(5999.988)
16	1.457	(939.998)	48	10.850	(6999.986)
17	1.563	(1008.385)	49	11.500	(7419.430)
18	1.620	(1045.159)	50	13.500	(8709.660)
19	1.800	(1161.288)	51	13.900	(8967.724)
20	1.990	(1283.868)	52	14.200	(9161.272)
21	2.130	(1374.191)	53	15.500	(9999.980)
22	2.380	(1535.481)	54	16.000	(10,322.560)
23	2.620	(1690.319)	55	16.900	(10,903.204)
24	2.630	(1696.771)	56	18.800	(12,129.008)
25	2.880	(1858.061)	57	19.900	(12,838.684)
26	2.930	(1890.319)	58	22.000	(14,193.520)
27	3.090	(1993.544)	59	22.900	(14,774.164)
28	1.130	(729.031)	60	24.500	(15,806.420)
29	3.380	(2180.641)	61	26.500	(17,096.740)
30	3.470	(2238.705)	62	28.000	(18,064.480)
31	3.550	(2290.318)	63	30.000	(19,354.800)
32	3.630	(2341.931)	64	33.500	(21,612.860)

**Table 6.3** Loading conditions for the 25-bar spatial truss

Node	Case 1			Case 2		
	P <sub>X</sub> kips (kN)	P <sub>Y</sub> kips (kN)	P <sub>Z</sub> kips (kN)	P <sub>X</sub> kips (kN)	P <sub>Y</sub> kips (kN)	P <sub>Z</sub> 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 6.4** Optimal design comparison for the 25-bar spatial truss (Case 1)

		Optimal cross-sectional areas ( $\text{in}^2$ )									
		Wu and Chow [8]		Lee and Geem [9]		Li et al. [10]		Kaveh and Talatahari [11]		HPSACO	
		GA	HS	PSO	FSOPC	HPSO		cm $^2$	in $^2$	cm $^2$	in $^2$
1	A1	0.40	0.01	0.01	0.01	0.01	0.01	0.07	0.01	0.07	0.07
2	A2-A5	2.00	2.00	2.00	2.00	2.00	1.60	10.32	1.60	10.32	10.32
3	A6-A9	3.60	3.60	3.60	3.60	3.60	3.20	20.65	3.20	20.65	20.65
4	A10-A11	0.01	0.01	0.01	0.01	0.01	0.01	0.07	0.01	0.07	0.07
5	A12-A13	0.01	0.01	0.40	0.01	0.01	0.01	0.07	0.01	0.07	0.07
6	A14-A17	0.80	0.80	0.80	0.80	0.80	0.80	5.16	0.80	5.16	5.16
7	A18-A21	2.00	1.60	1.60	1.60	1.60	2.00	12.90	2.00	12.90	12.90
8	A22-A25	2.40	2.40	2.40	2.40	2.40	2.40	15.48	2.40	15.48	15.48
Weight (lb)		563.52	560.59	566.44	560.59	560.59	551.6	250.2 kg	551.6	250.2 kg	250.2 kg

**Table 6.5** Optimal design comparison for the 25-bar spatial truss (Case 2)

		Optimal cross-sectional areas ( $\text{in}^2$ )								
		Wu and Chow [8]			Li et al. [10]		Kaveh and Talatahari [11] HPSACO		Present work [1]	
Element group		GA	PSO	PSOPC	HPSO	$\text{in}^2$	$\text{cm}^2$	$\text{in}^3$	$\text{cm}^3$	
1	A1	0.31	1.00	0.11	0.11	0.11	0.72	0.11	0.72	
2	A2–A5	1.99	2.62	1.56	2.13	2.13	13.74	2.13	13.74	
3	A6–A9	3.13	2.62	3.38	2.88	2.88	18.58	2.88	18.58	
4	A10–A11	0.11	0.25	0.11	0.11	0.11	0.72	0.11	0.72	
5	A12–A13	0.14	0.31	0.11	0.11	0.11	0.72	0.11	0.72	
6	A14–A17	0.77	0.60	0.77	0.77	0.77	4.94	0.77	4.94	
7	A18–A21	1.62	1.46	1.99	1.62	1.62	10.45	1.62	10.45	
8	A22–A25	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	

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

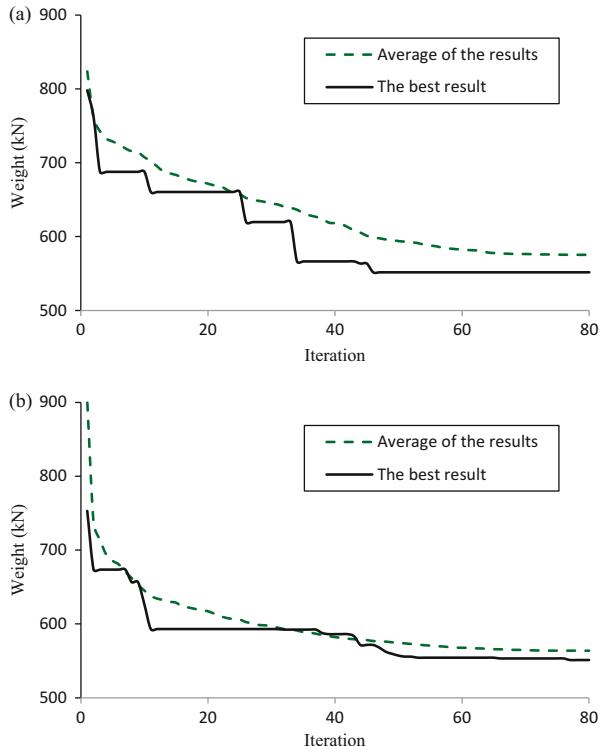
For the 72-bar spatial truss structure shown in Fig. 6.14, the material density is 0.1 lb/ $\text{in}^3$  (2767.990 kg/m $^3$ ), and the modulus of elasticity is 10,000 ksi (68,950 MPa). The members are subjected to the stress limits of  $\pm 25$  ksi ( $\pm 172.375$  MPa). The nodes are subjected to the displacement limits of  $\pm 0.25$  in ( $\pm 0.635$  cm).

The 72 structural members of this spatial truss are sorted into 16 groups using symmetry: (1) A<sub>1</sub>–A<sub>4</sub>, (2) A<sub>5</sub>–A<sub>12</sub>, (3) A<sub>13</sub>–A<sub>16</sub>, (4) A<sub>17</sub>–A<sub>18</sub>, (5) A<sub>19</sub>–A<sub>22</sub>, (6) A<sub>23</sub>–A<sub>30</sub>, (7) A<sub>31</sub>–A<sub>34</sub>, (8) A<sub>35</sub>–A<sub>36</sub>, (9) A<sub>37</sub>–A<sub>40</sub>, (10) A<sub>41</sub>–A<sub>48</sub>, (11) A<sub>49</sub>–A<sub>52</sub>, (12) A<sub>53</sub>–A<sub>54</sub>, (13) A<sub>55</sub>–A<sub>58</sub>, (14) A<sub>59</sub>–A<sub>66</sub> (15), A<sub>67</sub>–A<sub>70</sub>, and (16) A<sub>71</sub>–A<sub>72</sub>.

Two optimization cases are implemented.

**Case 1:** The discrete variables are selected from the set  $D = \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2.0, 2.1, 2.2, 2.3, 2.4, 2.5, 2.6, 2.7, 2.8, 2.9, 3.0, 3.1, 3.2\}$  ( $\text{in}^2$ ) or  $\{0.65, 1.29, 1.94, 2.58, 3.23, 3.87, 4.52, 5.16, 5.81, 6.45, 7.10, 7.74, 8.39, 9.03, 9.68, 10.32, 10.97, 12.26, 12.90, 13.55, 14.19, 14.84, 15.48, 16.13, 16.77, 17.42, 18.06, 18.71, 19.36, 20.00, 20.65\}$  ( $\text{cm}^2$ ).

**Fig. 6.12** The optimum answer convergence history for the 25-bar truss using DE [1]. (a) Case 1, (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 DHPACO [11].

Solving the problem using DE, the loops number is set to 200. Convergence curve is according to Eq. (6.1) considering  $PP_1 = 0.15$  and  $Power = 1$ .  $R_e$  and  $\varepsilon$  are equal to 5 and 1, respectively.

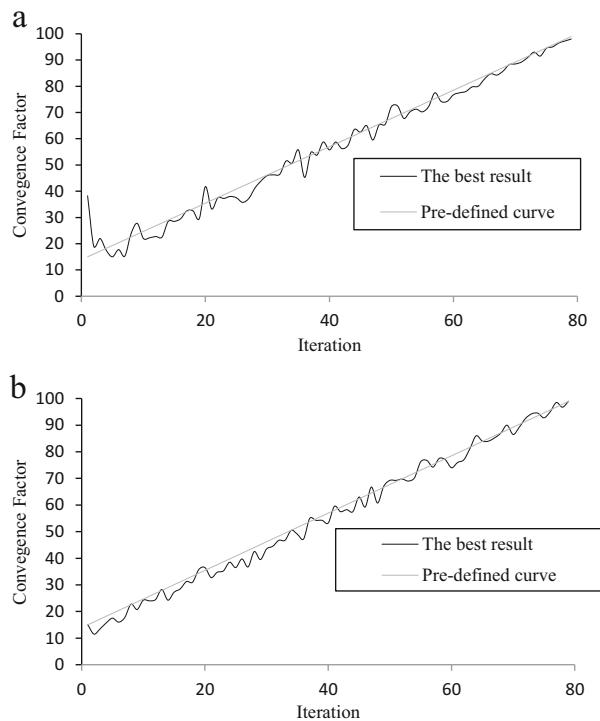
It can be seen from Table 6.7 that in Case 1 the best answer is achieved using DE that is better than GA and HS and although it is the same as DHPACO, but the penalty of the optimum answer is less than that of the DHPACO.

Moreover Table 6.8 shows that in Case 2, the DE achieves better results in comparison with the previously published works. Figure 6.15 shows that the DE can converge to the best answer in 200 loops and that it has a 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.

Figure 6.17 shows the allowable and existing displacements for the nodes of the 72-bar truss structure using the DE.

**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, (b) Case 2



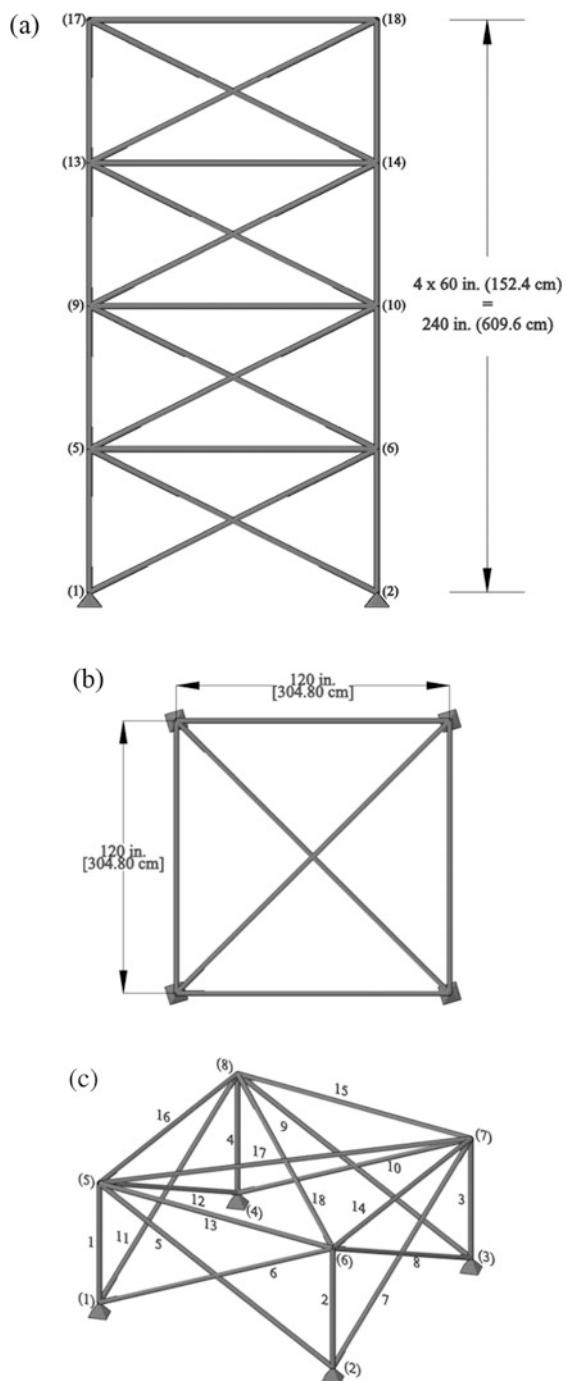
### 6.5.1.3 A 582-Bar Tower Truss

The 582-bar tower truss shown in Fig. 6.18 is chosen from Ref. [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  $6.16 \text{ in}^2$  ( $39.74 \text{ cm}^2$ ) and  $215.0 \text{ in}^2$  ( $1387.09 \text{ cm}^2$ ), respectively. The stress limitations of the members are imposed according to the provisions of ASD-AISC [12] as follows:

$$\begin{cases} \sigma_i^+ = 0.6F_y & \text{for } \sigma_i \geq 0 \\ \sigma_i^- & \text{for } \sigma_i < 0 \end{cases} \quad (6.18)$$

**Fig. 6.14** Schematic of a 72-bar spatial truss [1]. (a) Front view, (b) top view, (c) element and node numbering system of a typical story



**Table 6.6** Loading conditions for the 72-bar spatial truss

Node	Case 1			Case 2		
	P <sub>x</sub> kips (kN)	P <sub>y</sub> kips (kN)	P <sub>z</sub> kips (kN)	P <sub>x</sub> kips (kN)	P <sub>y</sub> kips (kN)	P <sub>z</sub> 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)

Element group	Optimal cross-sectional areas (in <sup>2</sup> )					
	Wu and Chow [8]	Lee and Geem [9]	Kaveh et al. [11]		Present work [1]	
	GA	HS	DHPSACO	DE	in <sup>2</sup>	cm <sup>2</sup>
in <sup>2</sup>	in <sup>2</sup>	in <sup>2</sup>	cm <sup>2</sup>	in <sup>2</sup>	cm <sup>2</sup>	
1 A1–A4	1.5	1.9	1.9	12.26	2.0	12.90
2 A5–A12	0.7	0.5	0.5	3.23	0.5	3.23
3 A13–A16	0.1	0.1	0.1	0.65	0.1	0.65
4 A17–A18	0.1	0.1	0.1	0.65	0.1	0.65
5 A19–A22	1.3	1.4	1.3	8.39	1.3	8.39
6 A23–A30	0.5	0.6	0.5	3.23	0.5	3.23
7 A31–A34	0.2	0.1	0.1	0.65	0.1	0.65
8 A35–A36	0.1	0.1	0.1	0.65	0.1	0.65
9 A37–A40	0.5	0.6	0.6	3.87	0.5	3.23
10 A41–A48	0.5	0.5	0.5	3.23	0.5	3.23
11 A49–A52	0.1	0.1	0.1	0.65	0.1	0.65
12 A53–A54	0.2	0.1	0.1	0.65	0.1	0.65
13 A55–A58	0.2	0.2	0.2	1.29	0.2	1.29
14 A59–A66	0.5	0.5	0.6	3.87	0.6	3.87
15 A67–A70	0.5	0.4	0.4	2.58	0.4	2.58
16 A71–A72	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

where  $\sigma_i^-$  is calculated according to the slenderness ratio:

$$\sigma_i^- = \begin{cases} \left[ \left( 1 - \frac{\lambda_i^2}{2C_i^2} \right) F_y \right] / \left( \frac{5}{3} + \frac{5\lambda_i}{8C_C} - \frac{\lambda_i^3}{8C_C^3} \right) & \text{for } \lambda_i < C_C \\ \frac{12\pi^2 E}{23\lambda_i^2} & \text{for } \lambda_i \geq C_C \end{cases} \quad (6.19)$$

**Table 6.8** Optimal design comparison for the 72-bar spatial truss (Case 2)

Element group		Optimal cross-sectional areas (in <sup>2</sup> )			
		Wu et al. [8]	Kaveh et al. [11]	Present work [1]	
		GA	DHPSCAO	DE	
		in <sup>2</sup>	in <sup>2</sup>	cm <sup>2</sup>	in <sup>2</sup>
1	A1–A4	0.196	1.800	11.610	2.130
2	A5–A12	0.602	0.442	2.850	0.442
3	A13–A16	0.307	0.141	0.910	0.111
4	A17–A18	0.766	0.111	0.720	0.111
5	A19–A22	0.391	1.228	7.920	1.457
6	A23–A30	0.391	0.563	3.630	0.563
7	A31–A34	0.141	0.111	0.720	0.111
8	A35–A36	0.111	0.111	0.720	0.111
9	A37–A40	1.800	0.563	3.630	0.442
10	A41–A48	0.602	0.563	3.630	0.563
11	A49–A52	0.141	0.111	0.720	0.111
12	A53–A54	0.307	0.250	1.610	0.111
13	A55–A58	1.563	0.196	1.270	0.196
14	A59–A66	0.766	0.563	3.630	0.563
15	A67–A70	0.141	0.442	2.850	0.307
16	A71–A72	0.111	0.563	3.630	0.563
Weight (lb)		427.203	393.380	178.4 kg	391.329
					177.47 kg

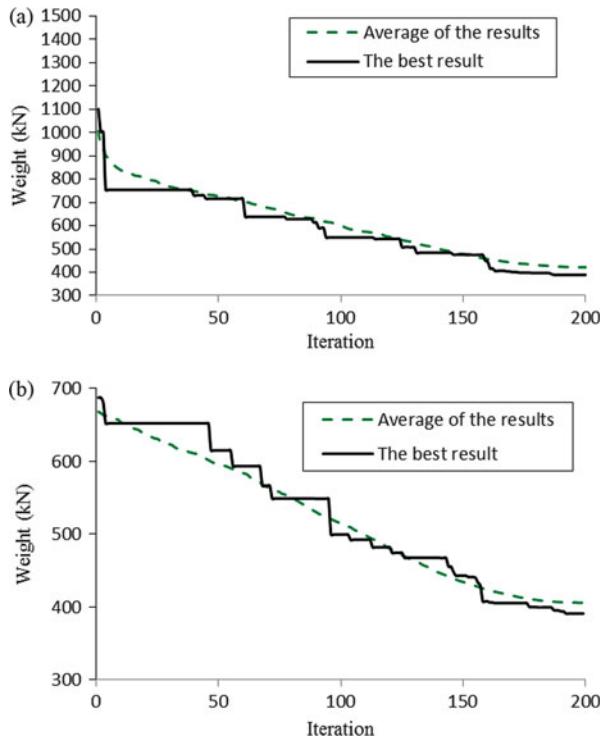
where  $E$  = the modulus of elasticity;  $F_y$  = the yield stress of A36 steel;  $C_C = \sqrt{2\pi^2 E/F_y}$ ;  $\lambda_i$  = the slenderness ratio ( $kL_i/r_i$ );  $k$  = the effective length factor;  $L_i$  = the member length; and  $r_i$  = the radius of gyration. The other constraint is the limitation 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 [12] design code provisions.

The problem was solved later by Kaveh and Talatahari [14] and Sonmez [15]. Two cases for analyzing are used according to Ref. [15] as follows:

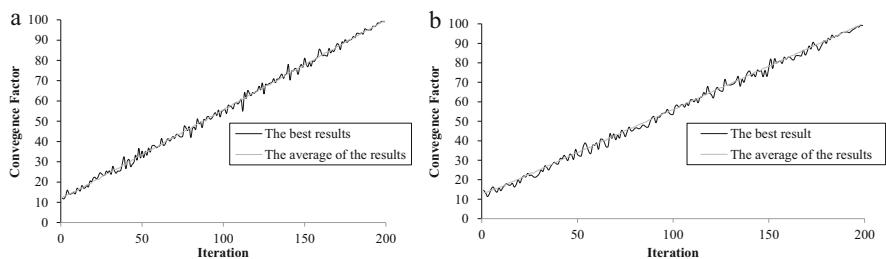
**Case 1:** All members are selected from a set of 140 W-shaped profiles according to Ref. [13], and the maximum number of evaluations is set to 50,000. For the DE, 25,000 evaluations are considered for this case to demonstrate the efficiency of the algorithm.

**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 Eq. (6.1) considering  $PP_I = 15\%$  and  $Power = 0.2$ .  $R_e$  and  $\epsilon$  are equal to 10 and 1, respectively.

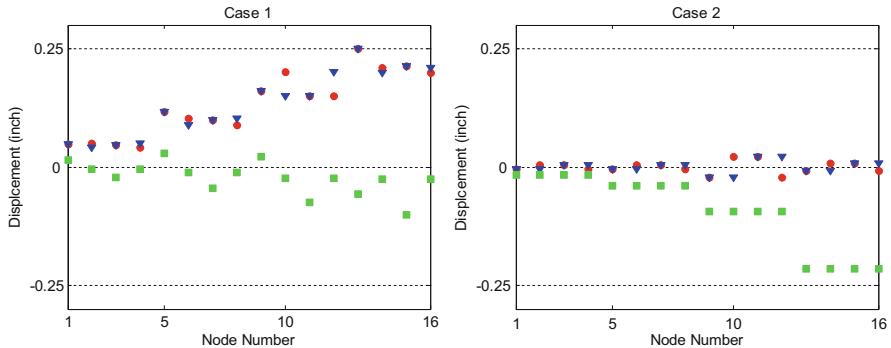


**Fig. 6.15** The optimum answer and average answers' convergence history for the 72-bar truss using the DE [1]. (a) Case 1, (b) Case 2



**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, (b) Case 2

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 Case 2, the DE result is 1.6% better than that 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 constraints 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 that, 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.

#### 6.5.1.4 Frame Structures

The displacement and AISC combined strength constraints are the performance constraints of the frames as follows:

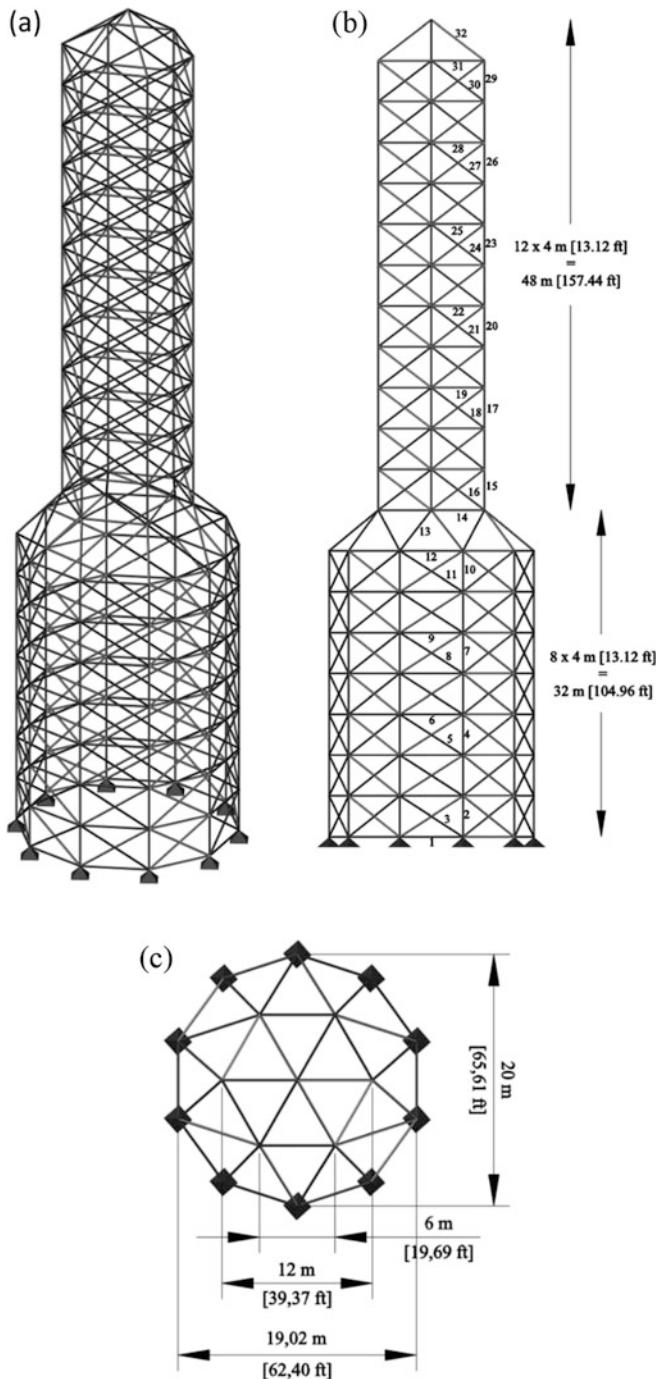
(a) Maximum lateral displacement:

$$\frac{\Delta_T}{H} < R \quad (6.20)$$

where  $\Delta_T$  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.

(b) The inter-story displacements:

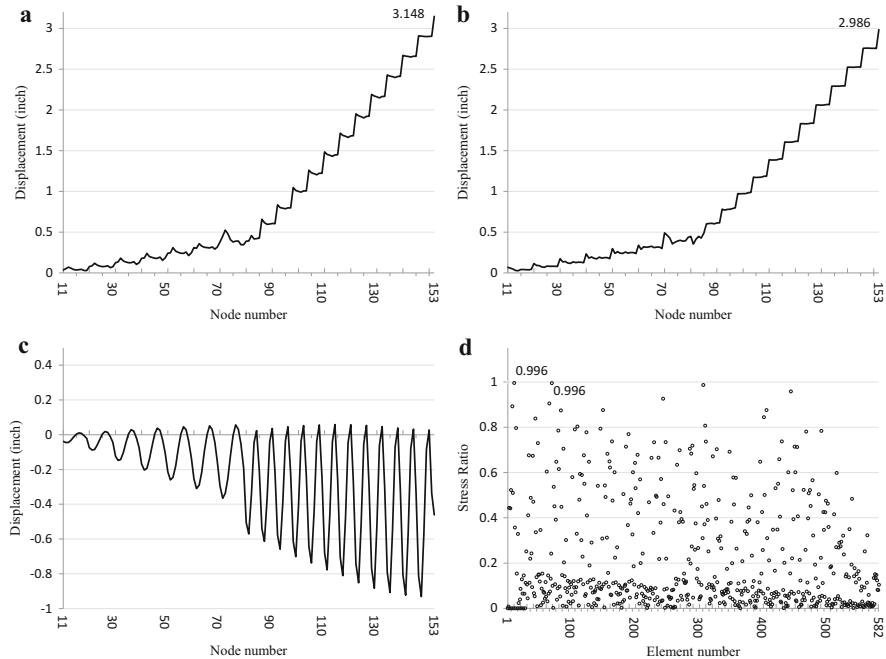
$$\frac{d_j}{h_j} < R_I, \quad j = 1, 2, \dots, ns \quad (6.21)$$



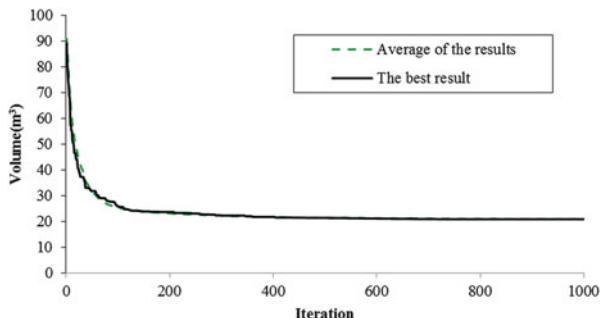
**Fig. 6.18** Schematic of a 582-bar tower truss. (a) Three-dimensional view, (b) side view, (c) top view

**Table 6.9** Optimal design comparison for the 582-bar spatial truss

Element group	Optimal cross-section					
	Case 1				Case 2	
	Hasançebi et al. [13]	Sonmez [15]	Kaveh et al. [14]	Present work [1]	Sonmez [15]	Present work [1]
	(PSO)	(ABC)	(DHP SACO)	(DE)	(ABC)	(DE)
Ready section	Ready section	Ready section	Ready section	Ready section	Ready section	Ready section
1	W8 × 21	W8 × 22	W8 × 24	W8 × 21	W8 × 22	W8 × 21
2	W12 × 79	W12 × 97	W12 × 72	W12 × 96	W10 × 78	W27 × 94
3	W8 × 24	W8 × 25	W8 × 28	W8 × 24	W8 × 25	W8 × 24
4	W10 × 60	W12 × 59	W12 × 58	W12 × 58	W14 × 62	W12 × 58
5	W8 × 24	W8 × 24	W8 × 24	W8 × 24	W8 × 24	W8 × 24
6	W8 × 21	W8 × 21	W8 × 24	W8 × 21	W8 × 21	W8 × 21
7	W14 × 48	W12 × 46	W10 × 49	W12 × 45	W12 × 51	W12 × 50
8	W8 × 24	W8 × 24	W8 × 24	W8 × 24	W8 × 24	W8 × 24
9	W8 × 21	W8 × 21	W8 × 24	W8 × 21	W8 × 21	W8 × 21
10	W10 × 45	W12 × 46	W12 × 40	W12 × 45	W10 × 50	W12 × 45
11	W8 × 24	W8 × 22	W12 × 30	W8 × 21	W8 × 25	W8 × 21
12	W10 × 68	W12 × 66	W12 × 72	W12 × 65	W10 × 69	W12 × 72
13	W14 × 74	W10 × 77	W18 × 76	W10 × 77	W18 × 77	W14 × 74
14	W14 × 48	W10 × 49	W10 × 49	W10 × 49	W14 × 49	W12 × 50
15	W18 × 76	W14 × 83	W14 × 82	W14 × 82	W10 × 78	W10 × 68
16	W8 × 31	W8 × 32	W8 × 31	W8 × 31	W8 × 32	W8 × 31
17	W16 × 67	W12 × 53	W14 × 61	W10 × 60	W21 × 62	W14 × 61
18	W8 × 24	W8 × 24	W8 × 24	W8 × 24	W8 × 24	W8 × 24
19	W8 × 21	W8 × 21	W8 × 21	W8 × 21	W8 × 21	W8 × 21
20	W8 × 40	W16 × 36	W12 × 40	W12 × 45	W14 × 43	W14 × 43
21	W8 × 24	W8 × 24	W8 × 24	W8 × 21	W8 × 24	W8 × 21
22	W8 × 21	W10 × 22	W14 × 22	W8 × 21	W8 × 21	W8 × 21
23	W10 × 22	W10 × 22	W8 × 31	W10 × 22	W8 × 24	W6 × 25
24	W8 × 24	W6 × 25	W8 × 28	W8 × 21	W8 × 24	W8 × 21
25	W8 × 21	W8 × 21	W8 × 21	W8 × 21	W8 × 21	W8 × 21
26	W8 × 21	W8 × 21	W8 × 21	W8 × 21	W8 × 21	W8 × 21
27	W8 × 24	W8 × 24	W8 × 24	W8 × 21	W8 × 24	W8 × 21
28	W8 × 21	W8 × 21	W8 × 28	W8 × 21	W8 × 21	W8 × 21
29	W8 × 24	W8 × 22	W16 × 36	W8 × 21	W8 × 21	W8 × 21
30	W8 × 21	W10 × 23	W8 × 24	W8 × 21	W8 × 21	W8 × 21
31	W8 × 21	W8 × 25	W8 × 21	W8 × 21	W8 × 24	W8 × 21
32	W8 × 24	W6 × 26	W8 × 24	W8 × 21	W8 × 24	W8 × 21
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	8500	25,000	100,000	50,000
Differences compared to DE	0.95 %	2.25 %	5.72 %		1.60 %	



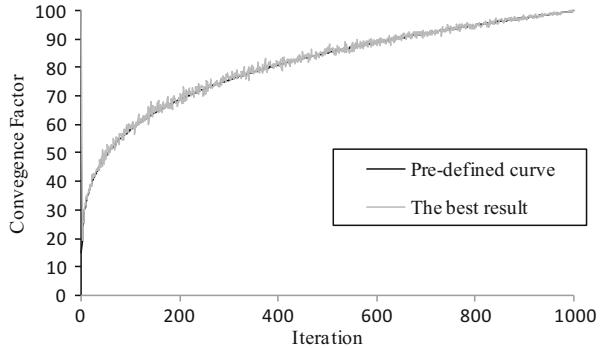
**Fig. 6.19** Comparison of the allowable and existing constraints for the 582-bar truss, Case 2 using DE [1]. (a) Displacement in the x direction. (b) Displacement in y direction. (c) Displacement in z direction. (d) Stress ratios



**Fig. 6.20** Convergence history of optimum result and average results for the 582-bar tower truss, Case 2, using DE [1]

where  $d_j$  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; and  $R_I$  is the inter-story drift index which is equal to 1/300 according to the ANSI/AISC 360-05 (2005), Ref. [16].

**Fig. 6.21** Convergence factor history for the 582-bar tower truss [1], Case 2 using DE



(c) Element forces:

$$\begin{aligned} \frac{P_u}{2\phi_c P_n} + \frac{M_u}{\phi_b M_n} &< 1 \quad \text{for} \quad \frac{P_u}{\phi_c P_n} < 0.2 \\ \frac{P_u}{\phi_c P_n} + \frac{8 M_u}{9\phi_b M_n} &< 1 \quad \text{for} \quad \frac{P_u}{\phi_c P_n} \geq 0.2 \end{aligned} \quad (6.22)$$

where  $P_u$  is the required strength (tension or compression);  $P_n$  is the nominal axial strength (tension or compression);  $\phi_c$  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  $\phi_b$  is the flexural resistance factor ( $\phi_b = 0.9$ ).

#### 6.5.1.5 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 Ref. [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 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 Eq. (6.1) considering  $PP_I = 0.15$  and  $Power = 1$ .  $R_e$  and  $\varepsilon$  are equal to 5 and 1, respectively.

Results of the present study and those of Refs. [7, 14], and [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).

**Fig. 6.22** Schematic of a 3-bay 15-story planar frame

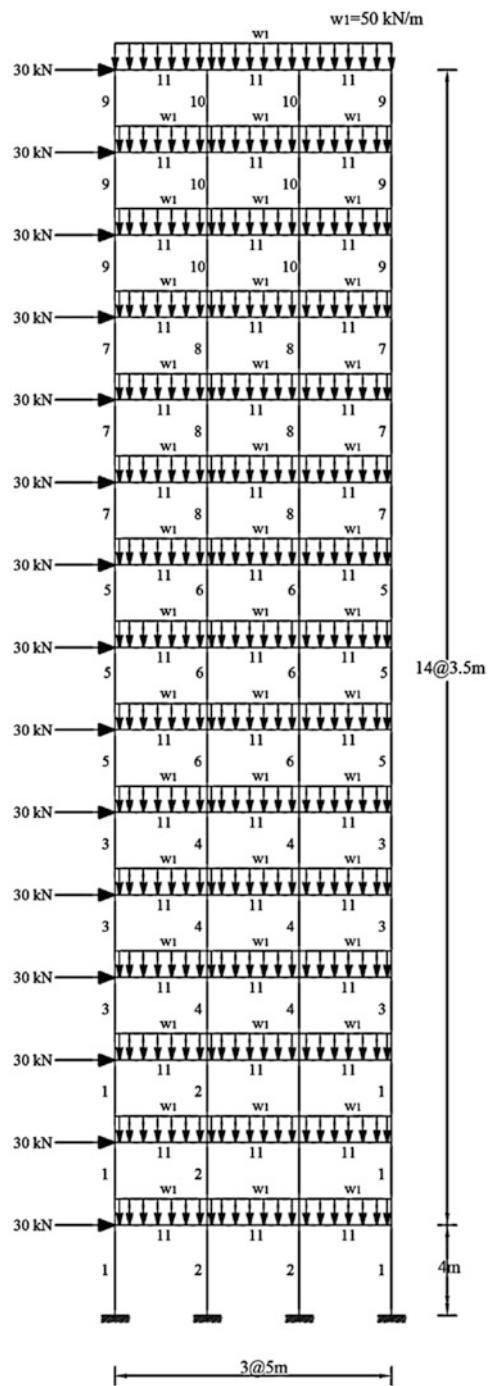
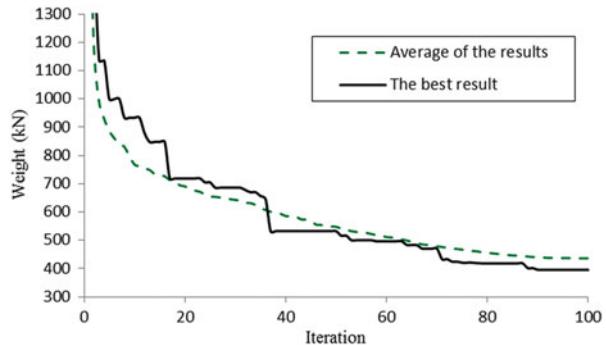


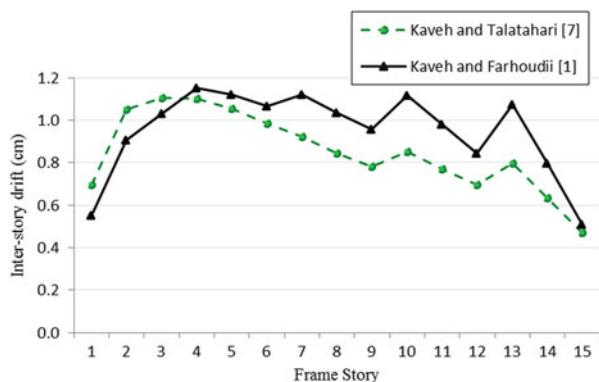
Table 6.10 Optimal design comparison for the 3-bay 15-story planar frame

Element group	Optimal W-shaped sections					Present work [1]
	Kaveh and Talatahari		HPSACO [14]		ICA [17]	
	PSO [14]	PSOPC [14]	HPSACO [14]	ICA [17]	CSS [7]	
1	W33 × 118	W27 × 129	W21 × 111	W24 × 117	W21 × 147	W12 × 87
2	W33 × 263	W24 × 131	W18 × 158	W21 × 147	W18 × 143	W36 × 182
3	W24 × 76	W24 × 103	W10 × 88	W27 × 84	W12 × 87	W21 × 93
4	W36 × 256	W33 × 141	W30 × 116	W27 × 114	W30 × 108	W18 × 106
5	W21 × 73	W24 × 104	W21 × 83	W14 × 74	W18 × 76	W18 × 65
6	W18 × 86	W10 × 88	W24 × 103	W18 × 86	W24 × 103	W14 × 90
7	W18 × 65	W14 × 74	W21 × 55	W12 × 96	W21 × 68	W10 × 45
8	W21 × 68	W27 × 94	W27 × 114	W24 × 68	W14 × 61	W12 × 65
9	W18 × 60	W21 × 57	W10 × 33	W10 × 39	W18 × 35	W6 × 25
10	W18 × 65	W18 × 71	W18 × 46	W12 × 40	W10 × 33	W10 × 45
11	W21 × 44	W21 × 44	W21 × 44	W21 × 44	W21 × 44	W21 × 44
Weight (kN)	496.68	452.34	426.36	417.466	412.62	395.35
Differences compared to DE	26 %	14 %	8 %	6 %	4 %	

**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]



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, and then the inter-story drifts are closer to the maximum allowable value.

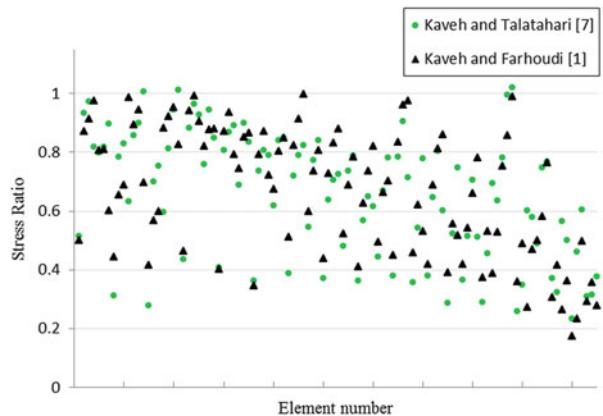
In Fig. 6.25 the stress ratios of the elements are shown. The maximum stress ratio is 99.69 %. One can see that similar to the inter-story limitation, stress ratios are closer to the limit line.

Figure 6.26 shows the *CF* changes during optimization. It is clear that the *CF* changes around predefined line.

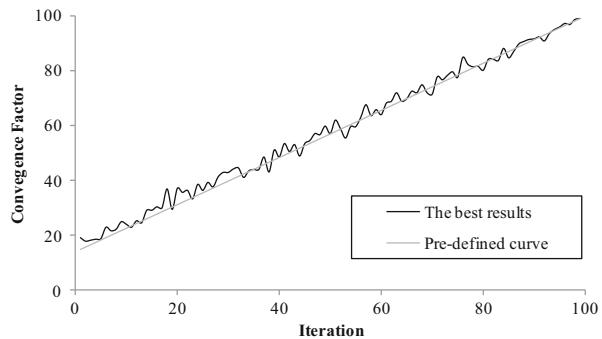
### 6.5.1.6 A 3-Bay 24-Story Planar Frame

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

**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]



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.

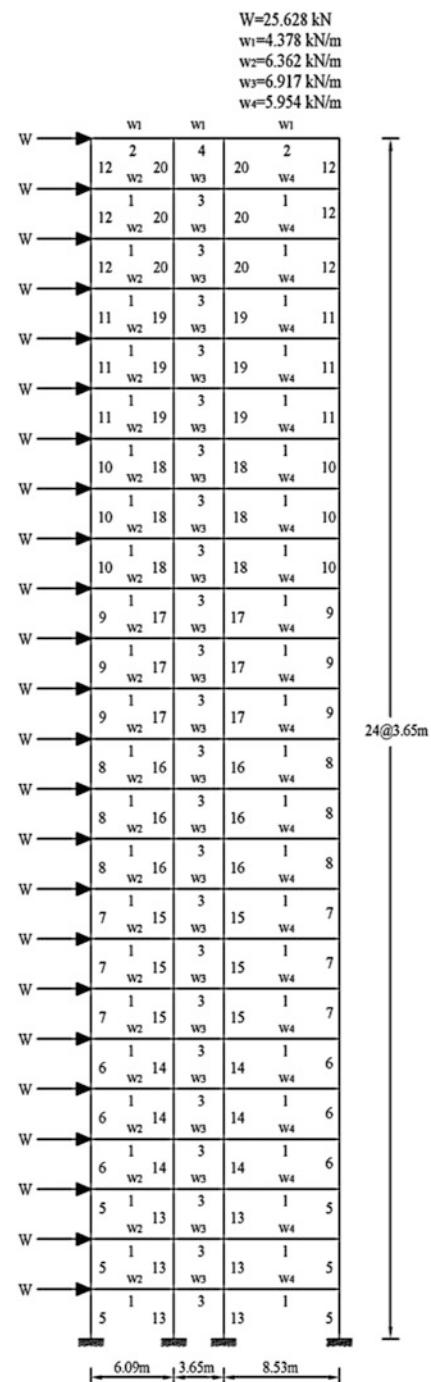
The effective length factors of the members are calculated as  $K_x \geq 0$  for the sway-permitted frame, and the out-of-plane effective length factor is specified as  $K_y = 1.0$ . All columns and beams are considered non-braced along their lengths. Fabrication conditions are imposed on the construction 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 four 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 Eq. (6.1) considering  $PP_I = 0.15$  and  $Power = 1$ .  $R_e$  and  $\epsilon$  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

**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

Element group	Optimal W-shaped sections				Present work [1]
	Camp et al. [18]	Degertekin [19]	Kaveh and Talatnahari [20]	ICA [7]	
1	ACO	HS	W30 × 90	W30 × 99	W30 × 90
2	W8 × 18	W10 × 22	W16 × 26	W21 × 50	W6 × 20
3	W24 × 55	W18 × 40	W18 × 35	W24 × 55	W21 × 44
4	W8 × 21	W12 × 16	W14 × 22	W8 × 28	W12 × 19
5	W14 × 145	W14 × 176	W14 × 145	W14 × 109	W14 × 159
6	W14 × 132	W14 × 176	W14 × 132	W14 × 159	W14 × 145
7	W14 × 132	W14 × 132	W14 × 120	W14 × 120	W14 × 132
8	W14 × 132	W14 × 109	W14 × 109	W14 × 90	W14 × 99
9	W14 × 68	W14 × 82	W14 × 48	W14 × 74	W14 × 68
10	W14 × 53	W14 × 74	W14 × 48	W14 × 68	W14 × 61
11	W14 × 43	W14 × 34	W14 × 34	W14 × 30	W14 × 43
12	W14 × 43	W14 × 22	W14 × 30	W14 × 38	W14 × 34
13	W14 × 145	W14 × 145	W14 × 159	W14 × 159	W14 × 145
14	W14 × 145	W14 × 132	W14 × 120	W14 × 132	W14 × 132
15	W14 × 120	W14 × 109	W14 × 109	W14 × 99	W14 × 109
16	W14 × 90	W14 × 82	W14 × 99	W14 × 82	W14 × 82
17	W14 × 90	W14 × 61	W14 × 82	W14 × 68	W14 × 74
18	W14 × 61	W14 × 48	W14 × 53	W14 × 48	W14 × 43
19	W14 × 30	W14 × 30	W14 × 38	W14 × 34	W14 × 30
20	W14 × 26	W14 × 22	W14 × 26	W14 × 22	W14 × 26
Weight (kN)	980.63	956.13	967.33	946.25	945.02
Difference compared to DE	7.5 %	4.8 %	6.0 %	3.7 %	3.6 %

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.

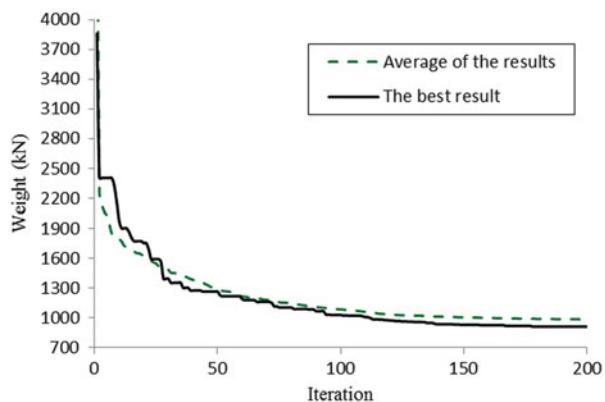
The maximum value of displacement is 26.11 cm which is less than the allowable limit (29.20 cm).

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 weight of structure, its stiffness is reduced, and the inter-story drifts are quite close to the maximum allowable value.

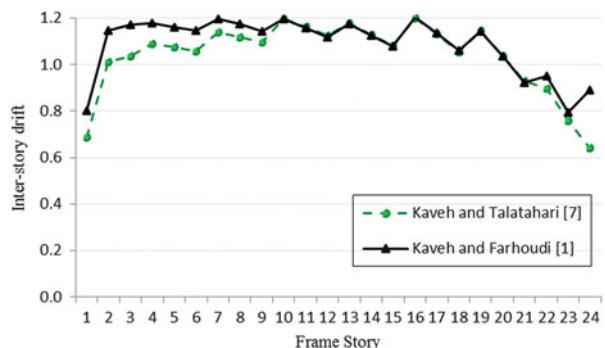
In Fig. 6.30 the stress ratios of the elements are shown. One can see that similar to the inter-story limitation, the stress ratios are closer to the limitation line. The maximum stress ratio is 98.33 %.

Figure 6.31 shows the *CF* changes during the optimization process. It is clear that the *CF* changes around the predefined line.

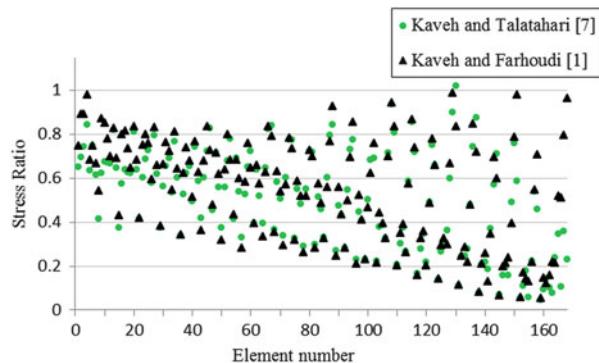
**Fig. 6.28** The optimum answer and the average answer, with the convergence history for the 3-bay 24-story frame using the DE [1]



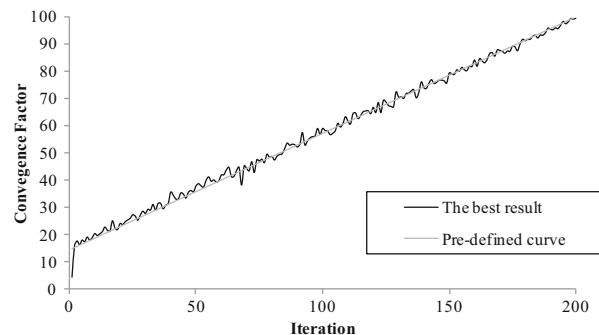
**Fig. 6.29** Comparison of the allowable and the existing inter-story drift for the 3-bay 24-story planar frame [1]



**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]



### 6.5.1.7 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
4. May J (1990) The Greenpeace book of dolphins. Greenpeace Communications Ltd., London
5. Thomas JA, Moss CF, Vater M (2002) Echolocation in bats and dolphins. University of Chicago Press, Chicago, IL
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. AISC, 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, 60601-1802, March 9
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**

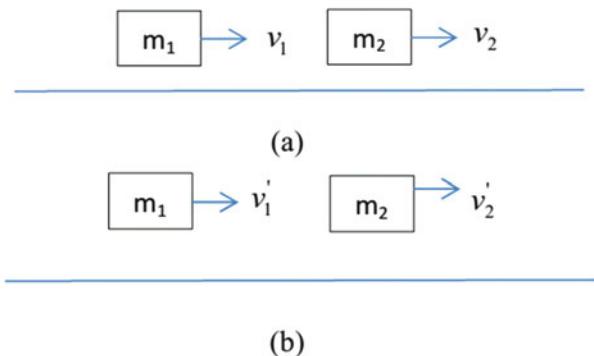
This chapter presents a novel efficient metaheuristic optimization algorithm called colliding bodies optimization (CBO) for optimization. This algorithm is based on one-dimensional collisions between bodies, with each agent solution being considered as the massed object or body. After a collision of 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 a simple formulation to find minimum or maximum of functions; also it is independent of parameters [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-studied 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).

**Fig. 7.1** The collision between two bodies. (a) Before the collision and (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:

$$m_1 v_1 + m_2 v_2 = m_1 v'_1 + m_2 v'_2 \quad (7.1)$$

Likewise, the conservation of the total kinetic energy is expressed as:

$$\frac{1}{2} m_1 v_1^2 + \frac{1}{2} m_2 v_2^2 = \frac{1}{2} m_1 v'_1^2 + \frac{1}{2} m_2 v'_2^2 + Q \quad (7.2)$$

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:

$$v'_1 = \frac{(m_1 - \epsilon m_2)v_1 + (m_2 + \epsilon m_1)v_2}{m_1 + m_2} \quad (7.3)$$

$$v'_2 = \frac{(m_2 - \epsilon m_1)v_2 + (m_1 + \epsilon m_2)v_1}{m_1 + m_2} \quad (7.4)$$

where  $\varepsilon$  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:

$$\varepsilon = \frac{|v_2' - v_1'|}{|v_2 - v_1|} = \frac{v'}{v} \quad (7.5)$$

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 ( $Q = 0$  and  $\varepsilon = 1$ ). 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 forms 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 ( $Q \neq 0$  &  $\varepsilon \leq 1$ ). In this case, after collision the velocity of separation is low.

For the most real objects, the value of  $\varepsilon$  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: (i) to improve the positions of moving objects and (ii) to push stationary objects toward better positions. After the collision, new positions of colliding bodies are updated based on new velocity by using the collision laws as discussed in the following:

The CBO procedure can briefly be outlined as

1. The initial positions of CBs are determined with random initialization of a population of individuals in the search space:

$$x_i^0 = x_{\min} + \text{rand}(x_{\max} - x_{\min}), \quad i = 1, 2, \dots, n, \quad (7.6)$$

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 value vectors of variables,  $\text{rand}$  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:

$$m_k = \frac{\frac{1}{\text{fit}(k)}}{\sum_{i=1}^n \frac{1}{\text{fit}(i)}}, \quad k = 1, 2, \dots, n \quad (7.7)$$

where  $\text{fit}(i)$  represents the objective function value of the agent  $i$  and  $n$  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  $\text{fit}(i)$  will be replaced by  $\frac{1}{\text{fit}(i)}$ .

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:

$$v_i = 0, \quad i = 1, \dots, \frac{n}{2} \quad (7.8)$$

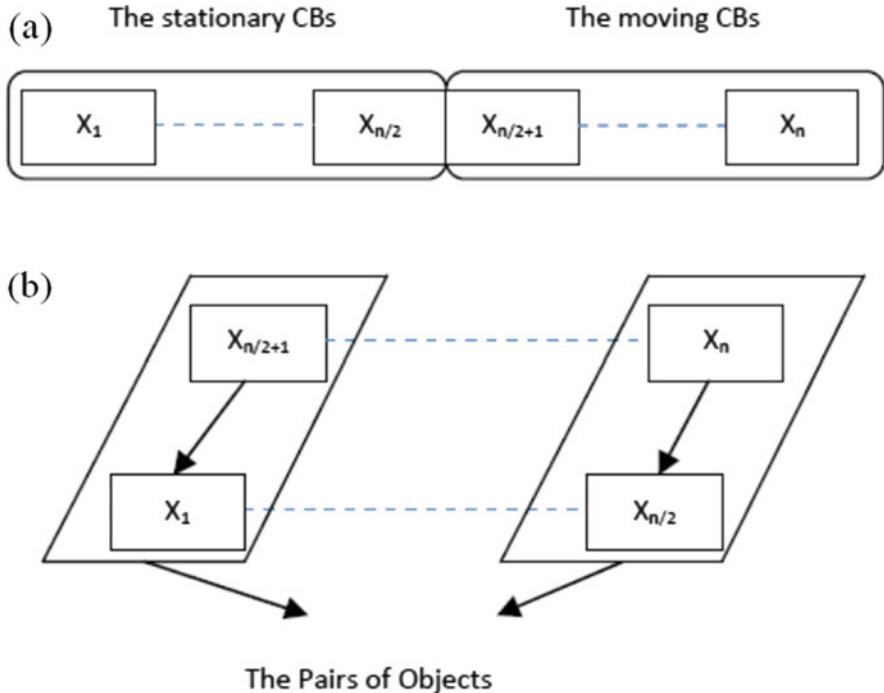
- The upper half of CBs (moving CBs): These CBs move toward the lower half. Then, according to Fig. 7.2b, the better and worse CBs, i.e., agents with upper fitness value, of each group will collide together. The change of the body position represents the velocity of these bodies before collision as:

$$v_i = x_i - x_{i-\frac{n}{2}}, \quad i = \frac{n}{2} + 1, \dots, n \quad (7.9)$$

where  $v_i$  and  $x_i$  are the velocity and position vector of the  $i$ th CB in this group, respectively, and  $x_{i-\frac{n}{2}}$  is the  $i$ th 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:

$$v'_i = \frac{\left( m_i - \epsilon m_{i-\frac{n}{2}} \right) v_i}{m_i + m_{i-\frac{n}{2}}}, \quad i = \frac{n}{2} + 1, \dots, n \quad (7.10)$$



**Fig. 7.2** (a) CBs sorted in increasing order; (b) colliding object pairs [1]

where  $v_i$  and  $v'_i$  are the velocity of the  $i$ th moving CB before and after the collision, respectively;  $m_i$  is the mass of the  $i$ th CB; and  $m_{i-\frac{n}{2}}$  is the mass of the  $i$ th CB pair. Also, the velocity of each stationary CB after the collision is:

$$v'_i = \frac{\left(m_{i+\frac{n}{2}} + \epsilon m_{i+\frac{n}{2}}\right)v_{i+\frac{n}{2}}}{m_i + m_{i+\frac{n}{2}}}, \quad i = 1, \dots, \frac{n}{2} \quad (7.11)$$

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 the mass of the  $i$ th CB;  $m_{i+\frac{n}{2}}$  is the mass of the  $i$ th moving CB pair; and  $\epsilon$  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 are:

$$x_i^{new} = x_{i-\frac{n}{2}} + rand \circ v'_i, \quad i = \frac{n}{2} + 1, \dots, n \quad (7.12)$$

where  $x_i^{new}$  and  $v_i'$  are the new position and the velocity after the collision of the  $i$ th moving CB, respectively, and  $x_{i-\frac{n}{2}}$  is the old position of the stationary CB pair. Also, the new positions of stationary CBs are obtained by:

$$x_i^{new} = x_i + \text{rand} \circ v_i', \quad i = 1, \dots, \frac{n}{2} \quad (7.13)$$

where  $x_i^{new}$ ,  $x_i$ , and  $v_i'$  are the new position, the old position, and the velocity after the collision of the  $i$ th stationary CB, respectively;  $\text{rand}$  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).

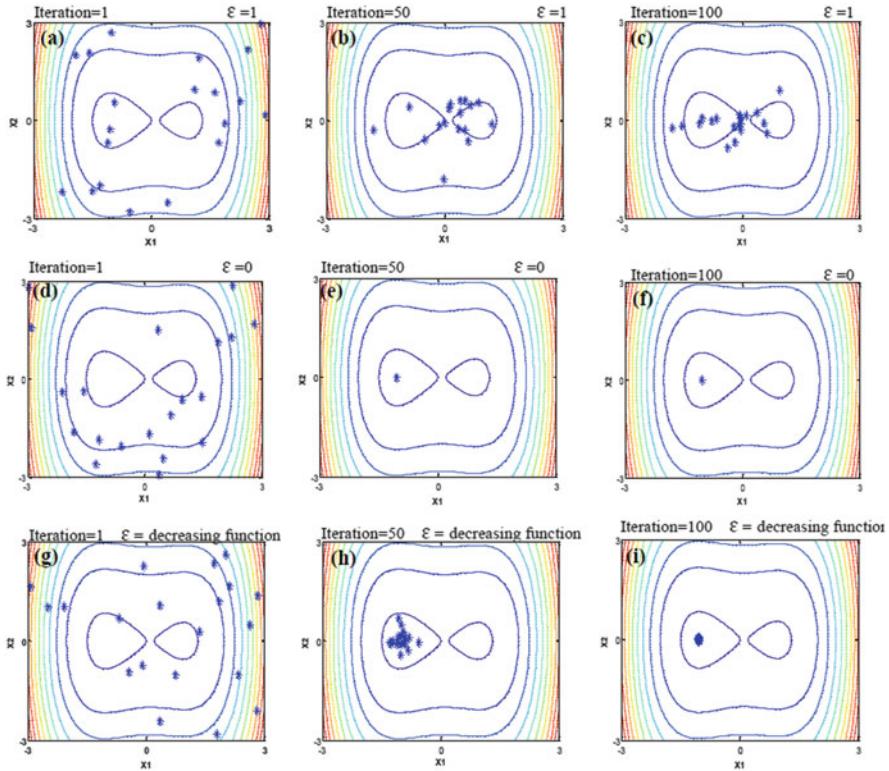
The penalty function approach was used for constraint handling. The fit (i) function corresponds to the effective cost. If optimization constraints are satisfied, there is no penalty; otherwise, the value of penalty is calculated as the ratio between the violation and the allowable limit.

### 7.2.2.2 The Coefficient of Restitution (COR)

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 chapter, 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-Pentini) chosen from Ref. [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 the 1st, 50th, and 100th iteration for these cases. These three typical cases result in the following:



**Fig. 7.3** Evolution of the positions of CBs during optimization history for different definitions of the coefficient of restitution (Aluffi-Pentini benchmark function) [1]

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).
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  $\epsilon$  is defined as:

$$\epsilon = 1 - \frac{iter}{iter_{max}} \quad (7.14)$$

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 4000. 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], and the evolution strategies developed by Montes and Coello [7] are 900,000, 200,000, and 25,000, respectively. Similar to CBO, the number of function evaluations for the charged system search algorithm developed by Kaveh and Talatahari [8] is 4000.

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 the 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 ( $\tau$ ), bending stress ( $\sigma$ ), buckling load ( $P_c$ ), and end deflection ( $\delta$ ) 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

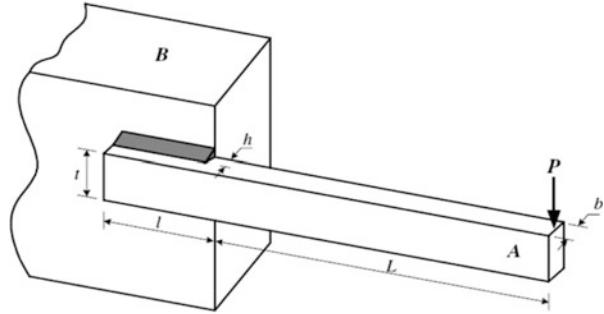
$$\{x_1, x_2, x_3, x_4\} \quad (7.15)$$

To minimize

$$\cos t(x) = 1.10471x_1^2x_2 + 0.04811x_3x_4(14 + x_2) \quad (7.16)$$

Subjected to

**Fig. 7.4** Schematic of the welded beam structure with indication of design variables



$$\begin{aligned}
 g_1(x) &= \tau(x) - \tau_{\max} \leq 0 \\
 g_2(x) &= \sigma(x) - \sigma_{\max} \leq 0 \\
 g_3(x) &= x_1 - x_4 \leq 0 \\
 g_4(x) &= 0.10471x_1^2 + 0.04811x_3x_4(14 + x_2) - 5 \leq 0 \\
 g_5(x) &= 0.125 - x_1 \leq 0 \\
 g_6(x) &= \delta(x) - \delta_{\max} \leq 0 \\
 g_7(x) &= p - p_c(x) \leq 0
 \end{aligned} \tag{7.17}$$

The bounds on the design variables are:

$$0.1 \leq x_1 \leq 2, \quad 0.1 \leq x_2 \leq 10, \quad 0.1 \leq x_3 \leq 10, \quad 0.1 \leq x_4 \leq 2 \tag{7.18}$$

where

$$\begin{aligned}
 \tau(x) &= \sqrt{(\tau')^2 + 2\tau'\tau''\frac{x_2}{2R} + (\tau'')^2} \\
 \tau' &= \frac{P}{\sqrt{2}x_1x_2}\tau'' = \frac{MR}{J}M = P\left(L + \frac{x_2}{2}\right)R = \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2} \\
 J &= 2\left\{\sqrt{2}x_1x_2\left[\frac{x_2^2}{12} + \left(\frac{x_1 + x_3}{2}\right)^2\right]\right\}\sigma(x) = \frac{6PL}{x_4x_3^2}\delta(x) = \frac{4PL^3}{Ex_3^3x_4} \\
 P_c(x) &= \frac{4.013\sqrt{E(x_3^2x_4^6/36)}}{L^2} \left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right)
 \end{aligned} \tag{7.19}$$

The constants in Eqs. (7.17) and (7.19) are chosen as follows:

$P = 6000$  lb,  $L = 14$  in.,  $E = 30 \times 10^6$  psi,  $G = 12 \times 10^6$  psi,  $\tau_{\max} = 13,600$  psi,  $\sigma_{\max} = 30,000$  psi, and  $\delta_{\max} = 0.25$  in.

Radgdsell 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 methods. Also, He and Wang [6] used effective coevolutionary particle swarm

**Table 7.1** Comparison of CBO optimized designs with literature for the welded beam problem

Design variables	Best solution found					Kaveh and Talatahari [8]	Present work
	Ragsdell and Phillips [9]	Deb [10]	Coello [11]	Coello and Montes [11]	He and Wang [6]		
$x_1$ (h)	0.245500	0.248900	0.208800	0.205986	0.202369	0.20582	0.205722
$x_2$ (l)	6.1960000	6.173000	3.420500	3.471328	3.544214	3.468109	3.47041
$x_3$ (t)	8.273000	8.178900	8.997500	9.020224	9.04821	9.038024	9.037276
$x_4$ (b)	0.245500	0.255300	0.210000	0.20648	0.205723	0.205723	0.205735
$f(x)$	1.728024	2.433116	1.748310	1.728226	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

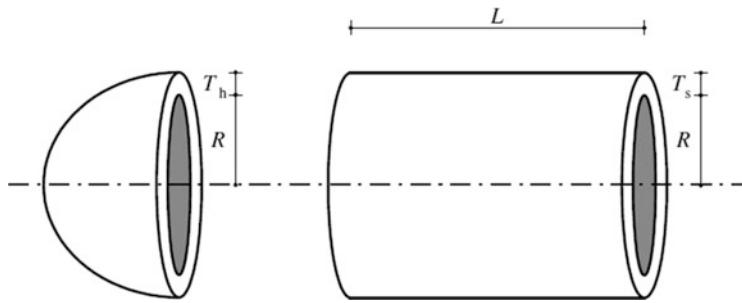
optimization, Montes and Coello [7] solved this problem utilizing evolution strategies, and Kaveh and Talatahari [8] employed charged system search.

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 (SD) 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 3000 psi and a minimum volume of 750 ft<sup>3</sup> 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 <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



**Fig. 7.5** Schematic of the spherical head and cylindrical wall of the pressure vessel with indication of design variables

$$\{x_1, x_2, x_3, x_4\} \quad (7.20)$$

To minimize

$$\cos t(x) = 0.6224x_3x_1x_4 + 1.7781x_3^2x_2 + 3.1611x_1^2x_4 + 19.8621x_3x_1^2 \quad (7.21)$$

Subject to

$$\begin{aligned} g_1(x) &= 0.0193x_3 - x_1 \leq 0 \\ g_2(x) &= 0.00954x_3 - x_2 \leq 0 \\ g_3(x) &= 750 \times 1728 - \pi x_3^2 x_4 - \frac{4}{3}\pi x_3^3 \leq 0 \\ g_4(x) &= x_4 - 240 \leq 0 \end{aligned} \quad (7.22)$$

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 in 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 (5889.911 versus 6059.088 of Ref. [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

**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]	8129.103	N/A	N/A	N/A
Kannan and Kramer [13]	7198.042	N/A	N/A	N/A
Deb and Gene [14]	6410.381	N/A	N/A	N/A
Coello [5]	6288.744	6293.843	6308.149	7.4133
Coello and Montes [11]	6059.946	6177.253	6469.322	130.9297
He and Wang [6]	6061.077	6147.133	6363.804	86.4545
Montes and Coello [7]	6059.745	6850.004	7332.879	426.0000
Kaveh and Talatahari [8]	6059.088	6067.906	6085.476	10.256
Present work [1]	5889.911	5934.201	6213.006	63.5417

**Fig. 7.6** Schematic of the tension/compression spring with indication of design variables

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

$$\cos t(x) = (x_3 + 2)x_2 x_1^2 \quad (7.25)$$

Subject to

$$\begin{aligned}
 g_1(x) &= 1 - \frac{x_2^3 x_3}{71785 x_1^4} \leq 0 \\
 g_2(x) &= \frac{4x_2^2 - x_1 x_2}{12566(x_2 x_1^3 - x_1^4)} + \frac{1}{5108 x_1^2} - 1 \leq 0 \\
 g_3(x) &= 1 - \frac{140.45 x_1}{x_2^2 x_3} \leq 0 \\
 g_4(x) &= \frac{x_1 + x_2}{1.5} - 1 \leq 0
 \end{aligned} \tag{7.26}$$

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] and 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 chapter 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 ASD-AISC (1989) [20] code, as follows:

$$\begin{cases} \sigma_i^+ = 0.6F_y & \text{for } \sigma_i \geq 0 \\ \sigma_i^- & \text{for } \sigma_i \leq 0 \end{cases} \tag{7.28}$$

where  $\sigma_i^-$  is calculated according to the slenderness ratio

**Table 7.5** Comparison of CBO optimized designs with literature for the tension/compression spring problem

Methods	Optimal design variables			Constraints				$f(x)$
	$x_1$ (d)	$x_2$ (D)	$x_3$ (N)	$g_1(x)$	$g_2(x)$	$g_3(x)$	$g_4(x)$	
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

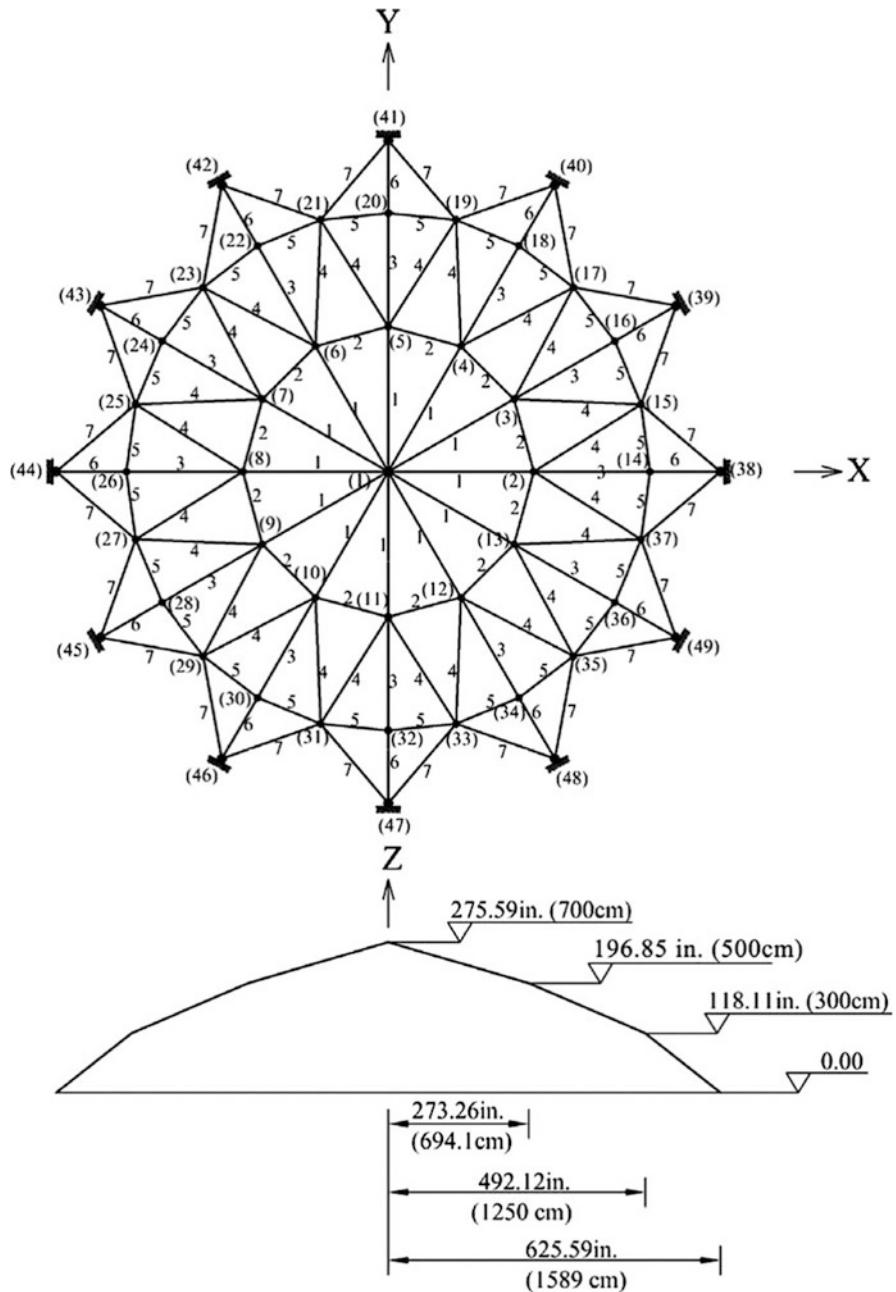
Methods	Best result	Average optimized cost	Worst result	Std dev
Belegundu [15]	0.0128334	N/A	N/A	N/A
Arora [16]	0.0127303	N/A	N/A	N/A
Coello [5]	0.0127048	0.012769	0.012822	3.9390e-5
Coello and Montes [11]	0.0126810	0.0127420	0.012973	5.9000e-5
He and Wang [6]	0.0126747	0.012730	0.012924	5.1985e-5
Montes 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

$$\sigma_i^- = \begin{cases} \left[ \left( 1 - \frac{\lambda_i^2}{2C_c^2} \right) F_y \right] / \left( \frac{5}{3} + \frac{3\lambda_i}{8C_c} - \frac{\lambda_i^3}{8C_c^3} \right) & \text{for } \lambda_i < C_c \\ \frac{12\pi^2 E}{23\lambda_i^2} & \text{for } \lambda_i \geq C_c \end{cases} \quad (7.29)$$

where  $E$  is the modulus of elasticity,  $F_y$  is the yield stress of steel,  $C_c$  is the slenderness ratio ( $\lambda_i$ ) dividing the elastic and inelastic buckling regions ( $C_c = \sqrt{2\pi^2 E/F_y}$ ),  $\lambda_i$  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<sup>3</sup> (7971.810 kg/m<sup>3</sup>). 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  $r_i = aA_i^{b_i}$  [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<sup>2</sup> (cm<sup>2</sup>). 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  $\pm 0.1969$  in (5 mm) imposed on all nodes in x and y directions (Case 2), no stress constraints but displacement limitations of  $\pm 0.1969$  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<sup>2</sup> (32.26 cm<sup>2</sup>) while for Case 3 and Case 4 is 20.0 in<sup>2</sup> (129.03 cm<sup>2</sup>).

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



**Fig. 7.7** Schematic of the spatial 120-bar dome truss with indication of design variables and main geometric dimensions

**Table 7.7** Comparison of CBO optimized designs with literature in the 120-bar dome problem

Optimal cross-sectional areas (in <sup>2</sup> )							
Case 1				Case 2			
Kaveh and Khayatazar [19]				Kaveh and Khayatazar [19]			
Element group	PSO	PSOPC	HPSACO	RO	Present work [1]	PSO	PSOPC
1	3.147	3.235	3.311	3.128	3.1229	15.97	3.083
2	6.376	3.370	3.428	3.357	3.3538	9.599	3.639
3	5.957	4.116	4.147	4.114	4.1120	7.467	4.095
4	4.806	2.784	2.831	2.783	2.7822	2.790	2.765
5	0.775	0.777	0.775	0.775	0.7750	4.324	1.776
6	13.798	3.343	3.474	3.302	3.3005	3.294	3.779
7	2.452	2.454	2.551	2.453	2.4458	2.479	2.438
Best weight (lb)	32,432.9	19,618.7	19,491.3	19,476.193	19,454.7	41,052.7	20,681.7
Average weight (lb)	—	—	—	—	19,466.0	—	—
Std (lb)	—	—	—	33,966	7.02	—	—
Case 3							
1	1.773	2.098	2.034	2.044	2.0660	12.802	3.040
2	17.635	16.444	15.151	15.665	15.9200	11.765	13.149
3	7.406	5.613	5.901	5.848	5.6785	5.654	5.646
4	2.153	2.312	2.254	2.290	2.2987	6.333	3.143
5	15.232	8.793	9.369	9.001	9.0581	6.963	8.759
6	19.544	3.629	3.744	3.673	3.6365	6.492	3.758
7	0.800	1.954	2.104	1.971	1.9320	4.988	2.502
Weight (lb)	46,893.5	31,776.2	31,670.0	31,733.2	31,724.1	51,986.2	33,481.2
Average weight (lb)	—	—	—	—	32,162.4	—	—
Std (lb)	—	—	—	274.991	240.22	—	—
Case 4							
1	1.773	2.098	2.034	2.044	2.0660	12.802	3.040
2	17.635	16.444	15.151	15.665	15.9200	11.765	13.149
3	7.406	5.613	5.901	5.848	5.6785	5.654	5.646
4	2.153	2.312	2.254	2.290	2.2987	6.333	3.143
5	15.232	8.793	9.369	9.001	9.0581	6.963	8.759
6	19.544	3.629	3.744	3.673	3.6365	6.492	3.758
7	0.800	1.954	2.104	1.971	1.9320	4.988	2.502
Weight (lb)	46,893.5	31,776.2	31,670.0	31,733.2	31,724.1	51,986.2	33,481.2
Average weight (lb)	—	—	—	—	32,162.4	—	—
Std (lb)	—	—	—	274.991	240.22	—	—

lower weight. CBO always completed the optimization process within 16,000 structural analyses ( $40 \text{ agents} \times 400 \text{ optimization iterations}$ ), while HPSACO required on average 10,000 analyses (400 optimization iterations) and PSOPC required 125,000 analyses (2500 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 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 ten layout variables. The cross-sectional areas (sizing variables) could vary between  $0.5$  and  $100 \text{ cm}^2$ .

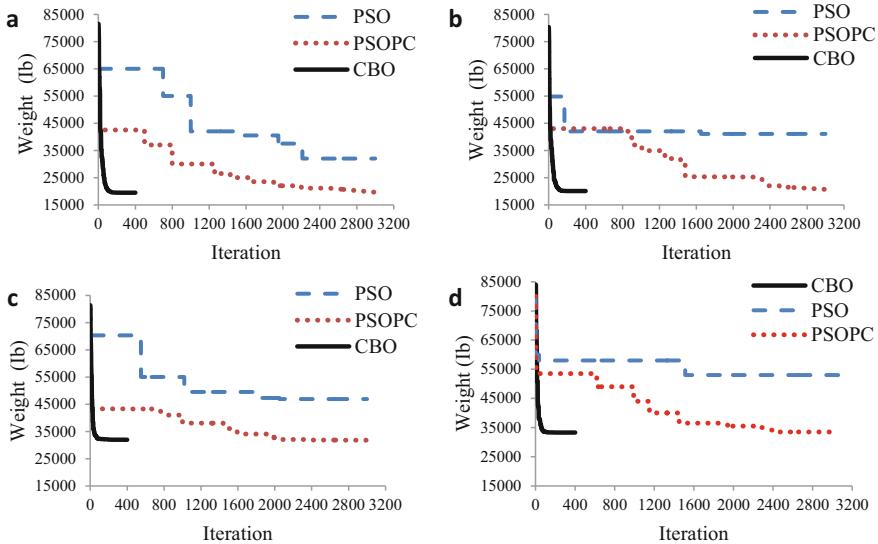
Material properties were set as follows: modulus of elasticity of  $210 \text{ GPa}$ , allowable stress of  $250 \text{ MPa}$ , and specific weight of  $7.8 \text{ t/m}^3$ . The structure is subject to self-weight and concentrated loads shown in Fig. 7.9a.

Table 7.8 compares CBO optimization results with literature. It appears that CBO found the best design overall saving about 1000 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.

## 7.3 CBO for Optimum Design of Truss Structures with Continuous Variables

This part considers the following: (i) The CBO algorithm is introduced for optimization of continuous problems. (ii) 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



**Fig. 7.8** Convergence curves obtained for the different variants of the 120-bar dome problem [2]

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

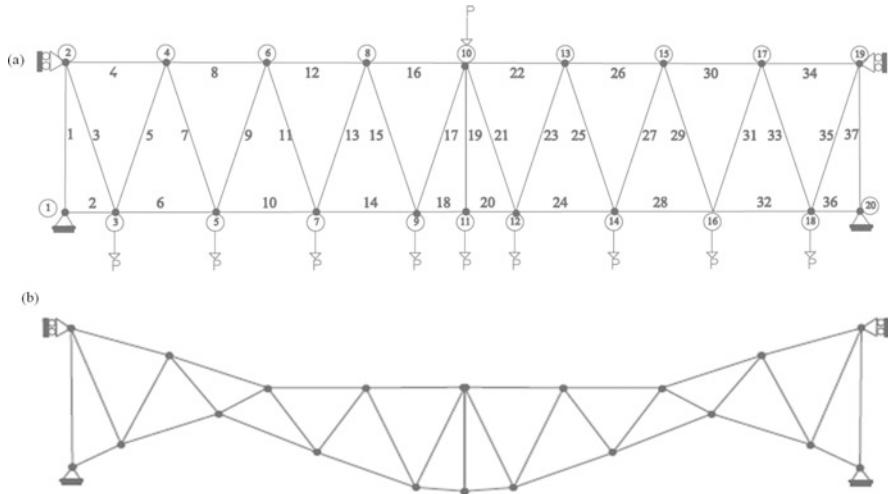
The flowchart of the CBO algorithm is shown in Fig. 7.11. The main steps of CBO algorithm are as follows:

#### Level 1: Initialization

- **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.



**Fig. 7.9** (a) Schematic of the Forth Truss Bridge, (b) optimized layout of the Forth Bridge [2]

- **Step 2: Group creation.** CBs are divided into two equal groups: (i) stationary group and (ii) 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 is evaluated before the collision [Eqs. (7.7)–(7.9)].
- **Step 4: Criteria after the collision.** The value velocity of each CB in each groups is 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 Eq. (7.14)].

### Level 3: Terminating Criterion Control

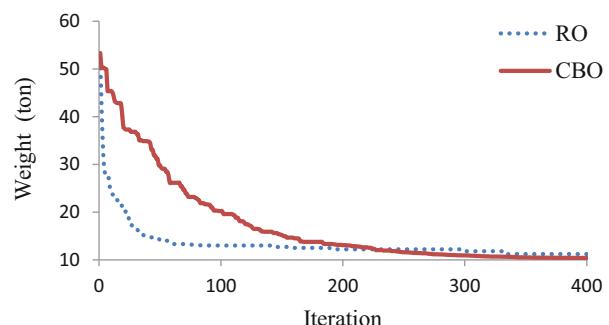
- **Step 1:** Repeat search level steps until a terminating criterion is satisfied.

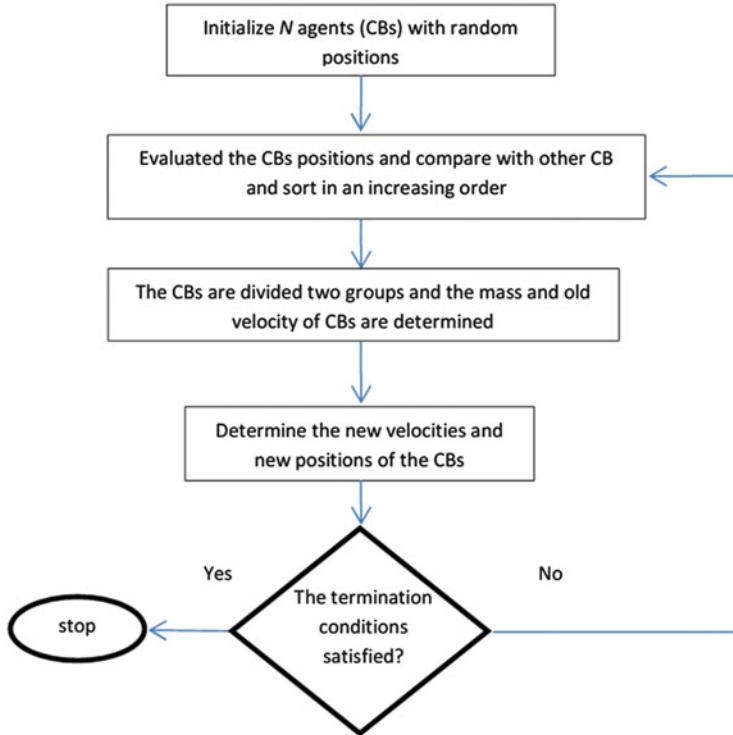
#### 7.3.2 Numerical Examples

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 numbers of design variables for the first to fifth examples are 8, 16, 32, and 26, respectively, and for the last example, 4, 8, and 12 variables are used. Similarly, the numbers of colliding bodies or agents for these examples are considered as

**Table 7.8** Comparison of CBO optimization results with literature for the Forth Bridge problem

No	Design variable	Kaveh and Khayatazad [19]			Present work [1]
		BB–BC	PSO	RO	
1	A <sub>1</sub>	56.41	25.20	20.54	23.314
2	A <sub>2</sub>	58.20	97.60	44.62	36.867
3	A <sub>3</sub> , A <sub>5</sub>	53.89	35.00	6.37	9.847
4	A <sub>4</sub>	60.21	64.30	50.10	49.679
5	A <sub>6</sub>	56.27	14.51	30.39	26.563
6	A <sub>7</sub>	57.08	37.91	17.61	12.737
7	A <sub>8</sub>	49.19	69.85	41.04	37.120
8	A <sub>10</sub>	48.67	8.76	8.55	1.545
9	A <sub>9</sub> , A <sub>11</sub>	45.43	47.54	33.93	28.35
10	A <sub>12</sub>	15.14	6.36	0.63	0.891
11	A <sub>14</sub>	45.31	27.13	26.92	24.110
12	A <sub>13</sub>	62.91	3.82	23.42	9.112
13	A <sub>18</sub>	56.77	50.82	42.06	29.071
14	A <sub>15</sub> , A <sub>17</sub>	46.66	2.70	2.01	8.222
15	A <sub>16</sub>	57.95	5.46	8.51	8.715
16	A <sub>19</sub>	54.99	17.62	1.27	2.107
17	Δy <sub>2</sub> , Δy <sub>19</sub>	6.89	140	70.88	11.093
18	Δy <sub>3</sub> , Δy <sub>18</sub>	17.74	139.65	64.88	50.352
19	Δy <sub>4</sub> , Δy <sub>17</sub>	1.81	117.59	-6.99	-50.529
20	Δy <sub>5</sub> , Δy <sub>16</sub>	23.57	139.70	128.31	119.315
21	Δy <sub>6</sub> , Δy <sub>15</sub>	3.22	-16.51	-64.24	-124.378
22	Δy <sub>7</sub> , Δy <sub>14</sub>	5.85	139.06	139.29	34.219
23	Δy <sub>8</sub> , Δy <sub>13</sub>	4.01	-127.74	-109.62	-120.867
24	Δy <sub>9</sub> , Δy <sub>12</sub>	10.52	-81.03	21.82	-41.323
25	Δy <sub>10</sub>	-25.99	60.16	-55.09	-115.609
26	Δy <sub>11</sub>	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)		1235.4	2323.7	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]

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:

$$Mer(X) = f(X) \times f_{penalty}(X) = f(X) \times \left( 1 + \varepsilon_1 \sum_{i=1}^{n_i} \max(0, g_i(x)) \right)^{\varepsilon_2} \quad (7.30)$$

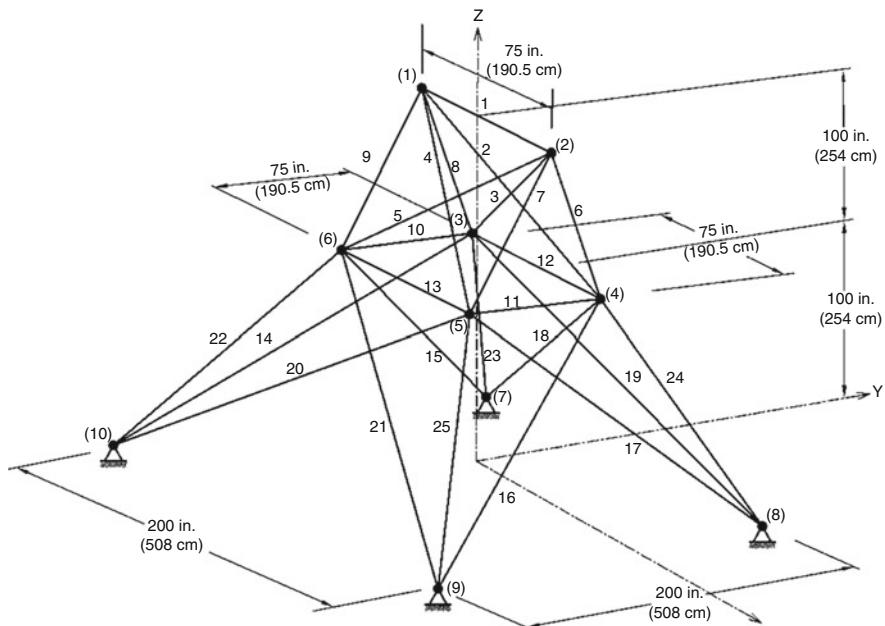
where  $X$  is the vector of design variables,  $g_i$  is the  $i$ th constraint from  $n_i$  inequality constraints ( $g_i(X) \leq 0$ ,  $i = 1, 2, \dots, n_i$ ),  $Mer(X)$  is the merit function,  $f(X)$  is the weight of structure, and  $f_{penalty}(X)$  is the penalty function which results from the violations of the constraints corresponding to the response of the structure. The parameters  $\varepsilon_1$  and  $\varepsilon_2$  are selected considering the exploration and the exploitation rate of the search space. In this study,  $\varepsilon_1$  is selected as unity and  $\varepsilon_2$  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  $0.1 \text{ lb/in}^3$  ( $2767.990 \text{ kg/m}^3$ ), and the modulus of elasticity is taken as 10,000 ksi ( $68,950 \text{ MPa}$ ). 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<sub>1</sub>, (2) A<sub>2</sub>–A<sub>5</sub>, (3) A<sub>6</sub>–A<sub>9</sub>, (4) A<sub>10</sub>–A<sub>11</sub>, (5) A<sub>12</sub>–A<sub>13</sub>, (6) A<sub>14</sub>–A<sub>17</sub>, (7) A<sub>18</sub>–A<sub>21</sub>, and (8) A<sub>22</sub>–A<sub>25</sub>.

Maximum displacement limitations of  $\pm 0.35$  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  $3.4 \text{ in}^2$  ( $0.6452\text{--}21.94 \text{ cm}^2$ ).

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 in Table 7.11 that the number of analyses and standard deviation of 20 independent runs for the CBO are 9090 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.



**Fig. 7.12** Schematic of a 25-bar spatial truss

**Table 7.9** Loading conditions for the 25-bar spatial truss

Node	Case 1			Case 2		
	P <sub>X</sub> kips (kN)	P <sub>Y</sub> kips (kN)	P <sub>Z</sub> kips (kN)	P <sub>X</sub> kips (kN)	P <sub>Y</sub> kips (kN)	P <sub>Z</sub> 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** Comparison of CBO optimized designs with literature in the 25-bar spatial truss

Element group	Optimal cross-sectional areas (in <sup>2</sup> )				
	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 <sub>1</sub>	0.10	0.010	0.047	0.0157
2	A <sub>2</sub> –A <sub>5</sub>	1.80	2.121	2.022	2.0217
3	A <sub>6</sub> –A <sub>9</sub>	2.30	2.893	2.95	2.9319
4	A <sub>10</sub> –A <sub>11</sub>	0.20	0.010	0.010	0.0102
5	A <sub>12</sub> –A <sub>13</sub>	0.10	0.010	0.014	0.0109
6	A <sub>14</sub> –A <sub>17</sub>	0.80	0.671	0.688	0.6563
7	A <sub>18</sub> –A <sub>21</sub>	1.80	1.611	1.657	1.6793
8	A <sub>22</sub> –A <sub>25</sub>	3.0	2.717	2.663	2.7163
Best weight (lb)	546	545.21	544.38	544.656	544.310
Average weight (lb)	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	9596	15,000	13,880	9090

### 7.3.2.2 A 72-Bar Spatial Truss Structure

Schematic topology and element numbering of a 72-bar space truss are shown in Fig. 7.14. The elements are classified in 16 design groups according to Table 7.12.

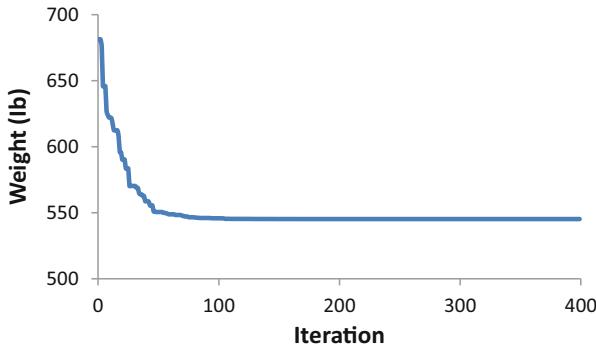


Fig. 7.13 The convergence diagram for the 25-bar spatial truss [2]

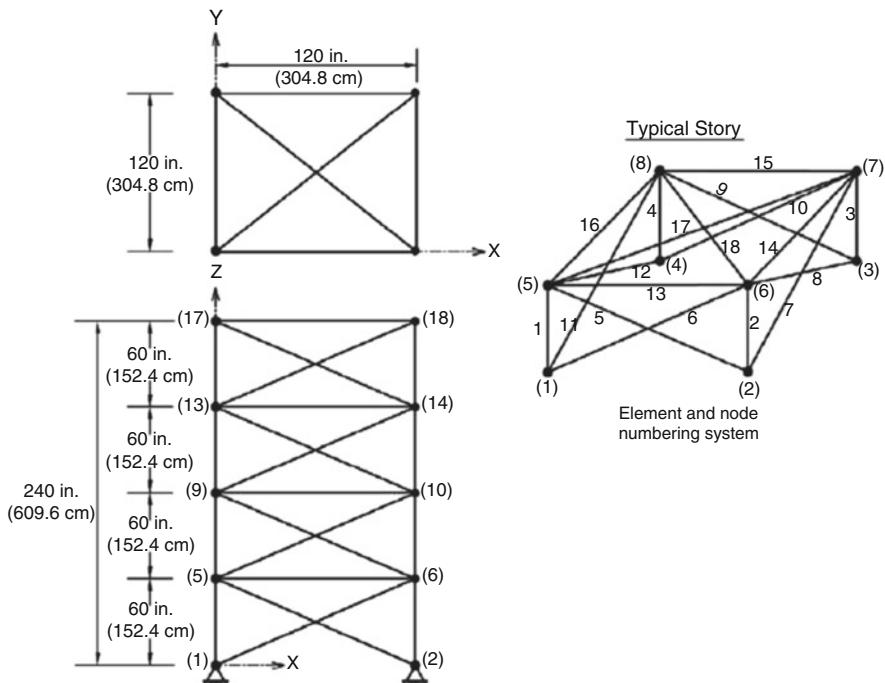
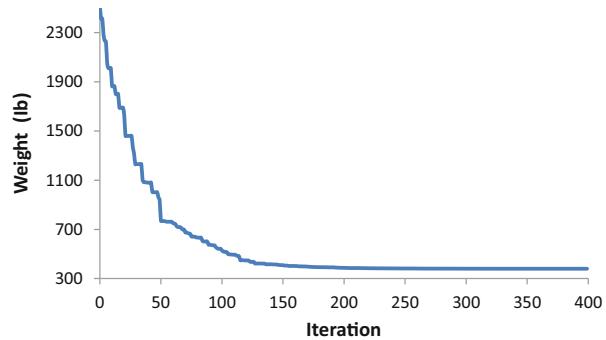


Fig. 7.14 Schematic of a 72-bar spatial truss

The material density is  $0.1 \text{ lb/in}^3$  ( $2767.990 \text{ kg/m}^3$ ) and the modulus of elasticity is taken as 10,000 ksi ( $68,950 \text{ MPa}$ ). The members are subjected to the stress limits of  $\pm 25 \text{ ksi}$  ( $\pm 172.375 \text{ MPa}$ ). The uppermost nodes are subjected to the displacement limits of  $\pm 0.25 \text{ in}$  ( $\pm 0.635 \text{ cm}$ ) in both  $x$  and  $y$  directions. The minimum permitted cross-sectional area of each member is taken as  $0.10 \text{ in}^2$  ( $0.6452 \text{ cm}^2$ ), and the

**Table 7.12** Comparison of CBO optimized designs with literature in the 72-bar spatial truss ( $\text{in}^2$ )

Element group	Optimal cross-sectional areas ( $\text{in}^2$ )					Present work [2]
	Erbatur et al. GA [23]	Camp et al. ACO [24]	Perez et al. PSO [25]	Camp BB–BC [26]	Kaveh et al. RO [19]	
1–4	1.755	1.948	1.7427	1.8577	1.8365	1.9028
5–12	0.505	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 (lb)	385.76	380.24	381.91	379.85	380.458	379.6943
Average weight (lb)	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]

maximum cross-sectional area of each member is  $4.00 \text{ in}^2$  ( $25.81 \text{ cm}^2$ ). 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

**Fig. 7.16** Comparison of the allowable and existing stresses in the elements of the 72-bar truss structure [2]

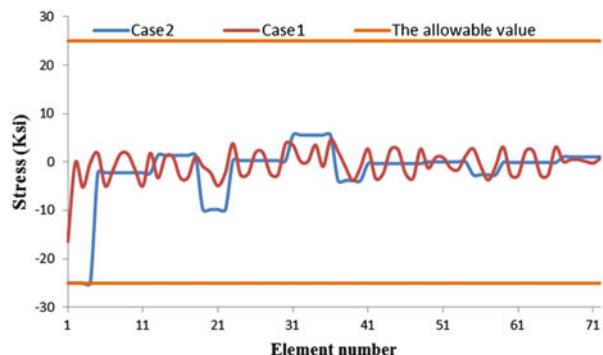


Table 7.12 summarizes the results obtained by the present work and those of the previously reported researches. The best result of the CBO approach is 379.694, while it is 385.76, 380.24, 381.91, 379.85, and 380.458 *Ib* for the GA [23], ACO [24], PSO [25], BB–BC [26], and RO [19] algorithm, respectively. Also, the number of analyses of the CBO is 15,600, while it is 18,500, 19,621, and 19,084 for the ACO, BB–BC, and RO algorithm, respectively. Also, it is evident in 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.

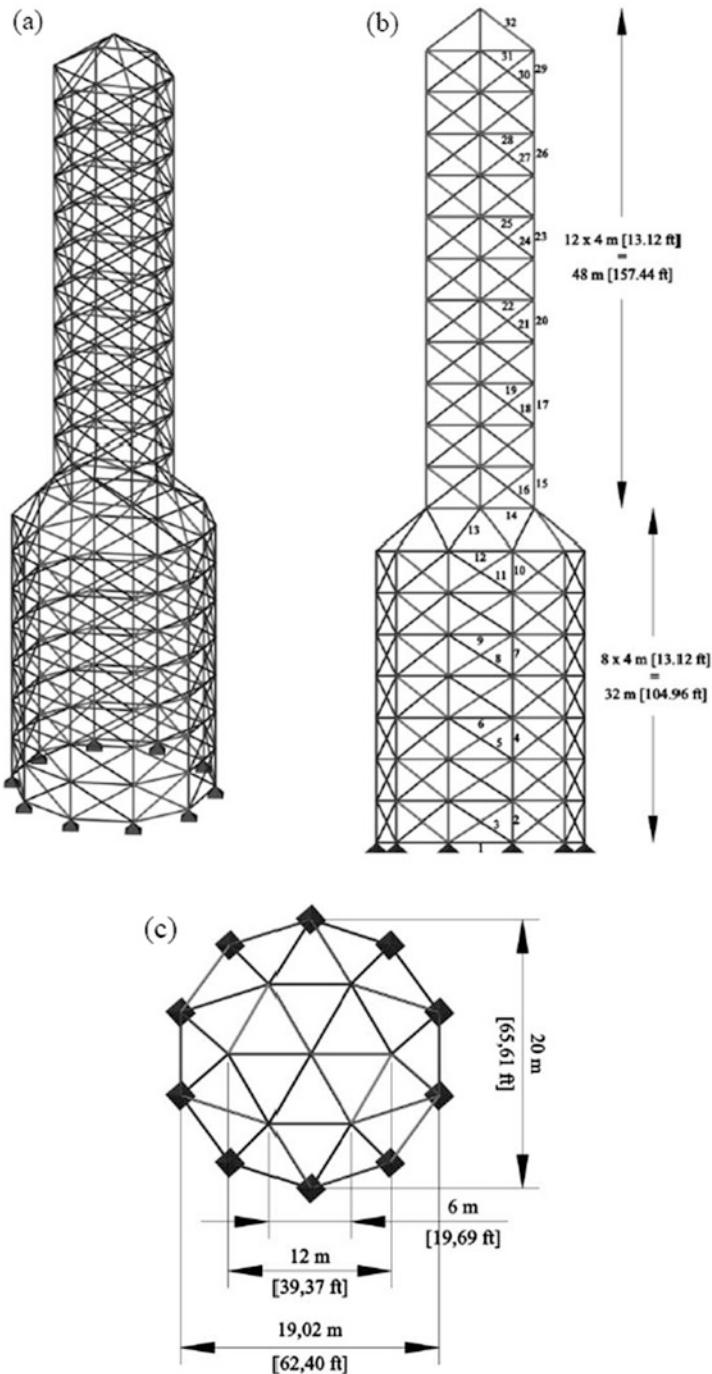
### 7.3.2.3 A 582-Bar Tower Truss

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 $^2$  ( $1000$  cm $^2$ ), respectively.

The allowable tensile and compressive stresses are used as specified by the ASD-AISC [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 limitations of nodal displacements which should be no more than 8.0 cm (3.15 in.) in all directions.

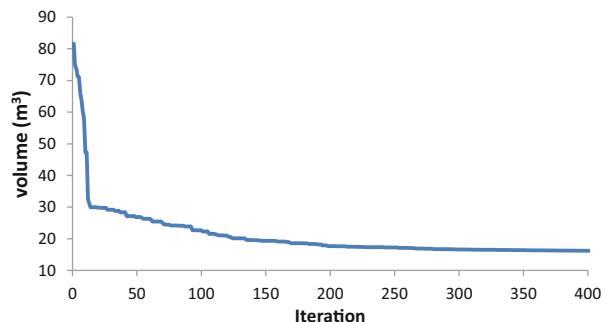
Table 7.13 lists the optimal values of the 32 size variables obtained by the present algorithm. Figure 7.18 shows the convergence diagrams for the utilized



**Fig. 7.17** Schematic of a 582-bar tower truss. (a) Three-dimensional view, (b) side view, (c) top view

**Table 7.13** Optimum design cross sections for the 582-bar tower truss

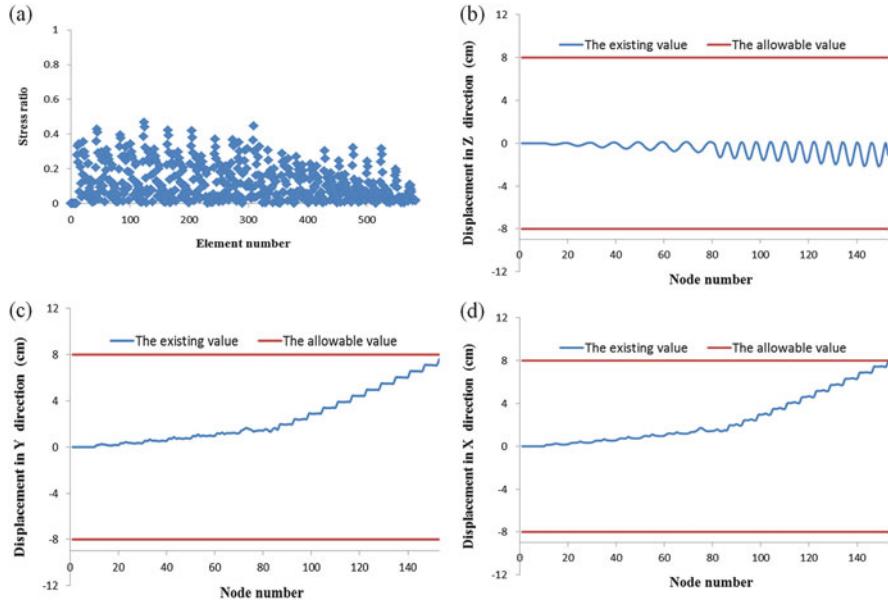
Element groups	Present work [2]	Element groups	Present work [2]
	Area, cm <sup>2</sup>		Area, cm <sup>2</sup>
1	20.5526	17	155.6601
2	162.7709	18	21.4951
3	24.8562	19	25.1163
4	122.7462	20	94.0228
5	21.6756	21	20.8041
6	21.4751	22	21.223
7	110.8568	23	53.5946
8	20.9355	24	20.628
9	23.1792	25	21.5057
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	171.9115	31	21.1888
16	31.5471	32	29.6669
	Volume (m <sup>3</sup> )		16.1520

**Fig. 7.18** The convergence diagram of the CBO for 582-bar tower truss [2]

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

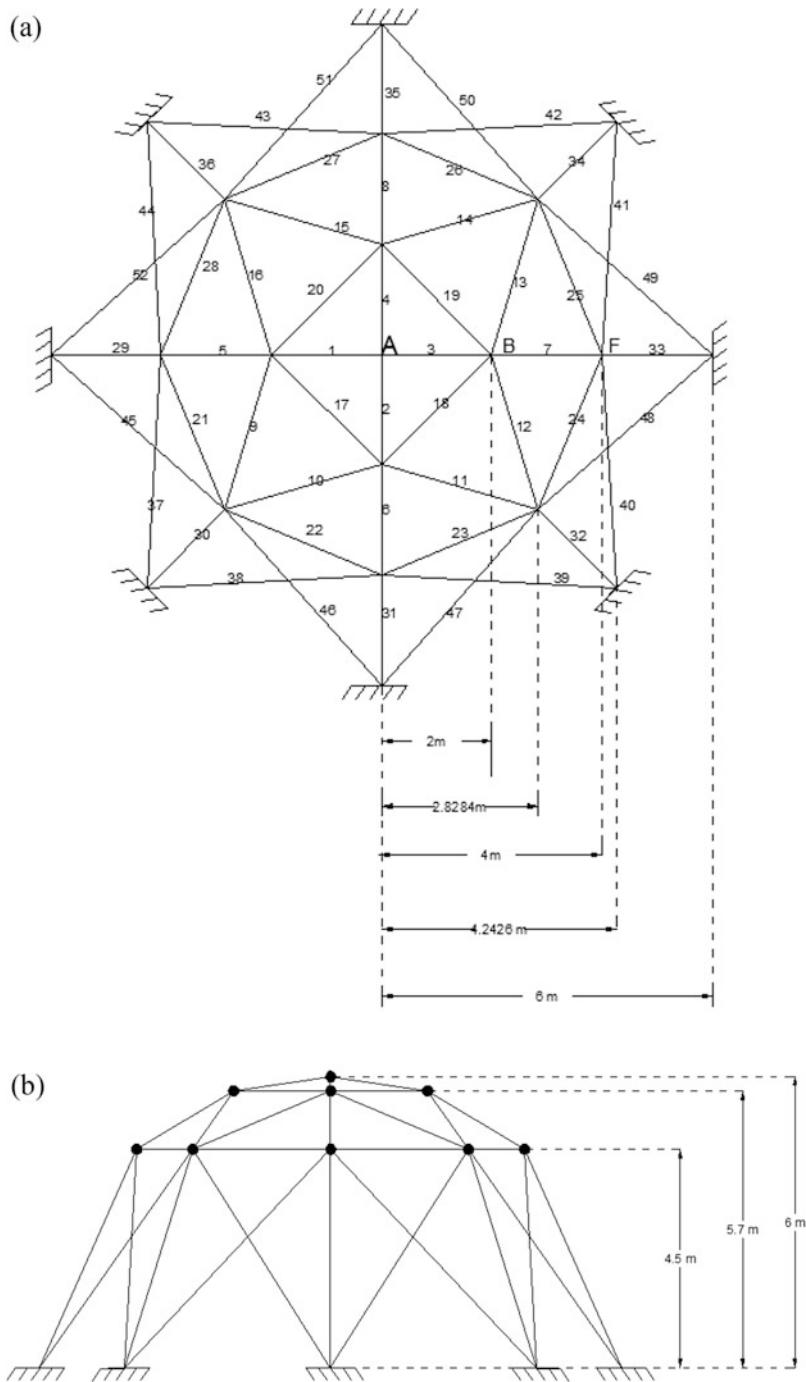
Figure 7.20 shows the initial topology and the element numbering of a 52-bar dome-like space truss. This example has been investigated by Lingyun et al. [29]. Gomes [30] utilized the NHGA and PSO algorithms. This has also been



**Fig. 7.19** Comparison of the allowable and existing constraints for the 582-bar truss using the DHPSCAO, [2], (a) stress ratio, (b) displacement in the z direction. (c) Displacement in the y direction. (d). Displacement in the x direction

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 nonstructural masses of  $m = 50 \text{ kg}$  are added to the free nodes. The material density is  $7800 \text{ kg/m}^3$  and the modulus of elasticity is  $210,000 \text{ MPa}$ . 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  $10 \text{ cm}^2$ . 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  $\pm 2 \text{ m}$ . There are two constraints in the first two natural frequencies so that  $\omega_1 \leq 15.916 \text{ Hz}$  and  $\omega_2 \geq 28.648 \text{ Hz}$ . This example is considered to be a 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 terms 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.



**Fig. 7.20** Schematic of the 52-bar space truss. (a) Top view, (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	4000	4000

**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

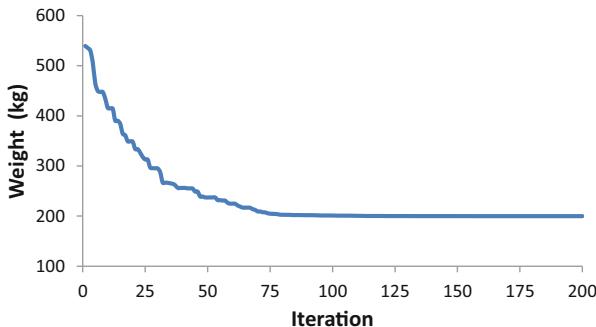


Fig. 7.21 Convergence history for the 52-bar truss [2]

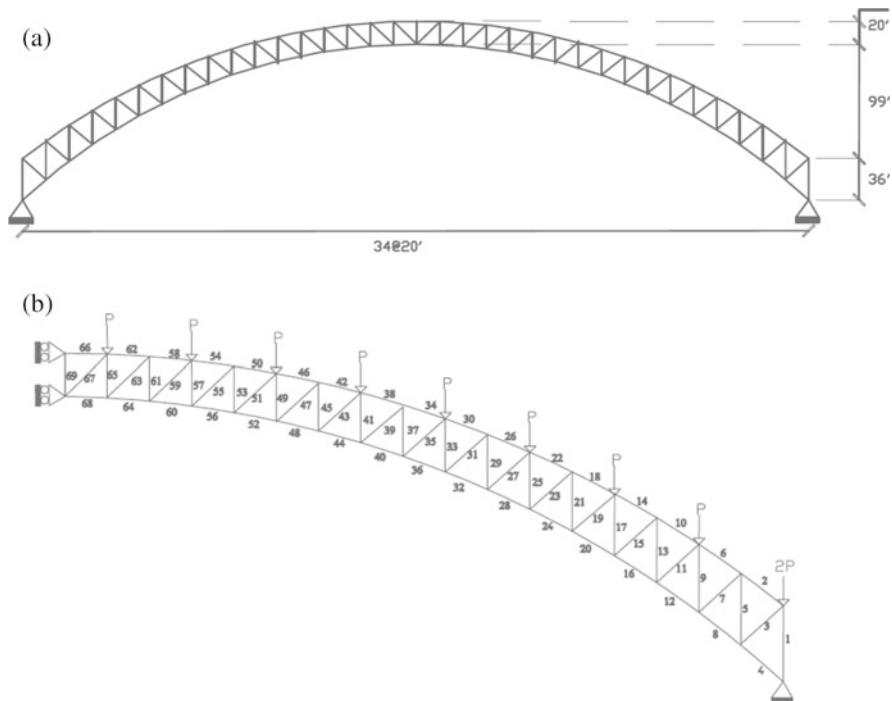


Fig. 7.22 (a) Schematic of the Burro Creek Bridge. (b) Finite element nodal and element numbering of Burro Creek Bridge

### 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

**Table 7.17** Three different design variables for the Burro Creek Bridge

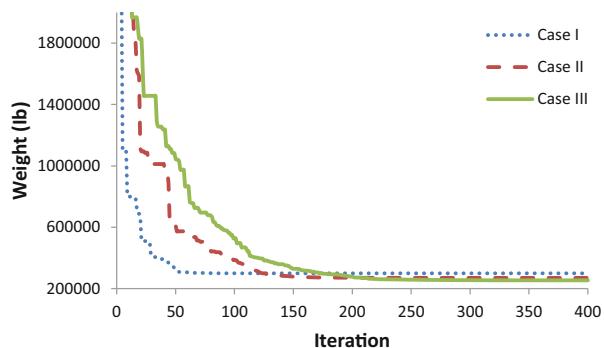
Design variables	Member number		
	Case 1 (4 variables)	Case 2 (8 variables)	Case 3 (12 variables)
1	67,63,59,55,51,47,43,39,35, 31,27,23,19,15,11,7,3	67,63,59,55,51,47,43,39,35, 31,27	67,63,59,55,51,47,43
2	66,62,58,54,50,46,42,38,34, 30,26,22,18,14,10,6,2	66,62,58,54,50,46,42,38,34, 30,26	66,62,58,54,50,46,42
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 ( $\text{in}^2$ ) with those of TLS for the Burro Creek Bridge

Design variables	Maktobi et al. TLS [32]			Present work [2]		
	Case 1	Case 2	Case 3	Case 1	Case 2	Case 3
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 (lb)	368,598.1	315,885.7	298,699.9	299,756.7	269,839.5	253,871.3
No. of analyses	15,000	35,000	50,000	8000	8000	8000

quadratic parabola. Because of symmetry of this truss, one can analyze 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  $4.2 \times 10^9 \text{ lb}/\text{ft}^2$ ,  $F_y$  is taken as  $72.0 \times 10^5 \text{ lb}/\text{ft}^2$ , and the density of material is  $495 \text{ lb}/\text{ft}^3$ . For this

**Fig. 7.23** Comparison of the convergence rates between three different cases for the Burro Creek Bridge [2]



example, allowable tensile and compressive stresses are considered according to ASD-AISC (1989) [20]. According to the 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 to 5.0 in<sup>2</sup>. 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 weights of the CBO are 299,756.7, 269,839.5, and 253,871.3 lb, while these are 368,598.1, 315,885.7, and 298,699.9 for Cases 1, 2, and 3, 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 a 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 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.

## References

1. Kaveh A, Mahdavi VR (2014) Colliding bodies optimization: a novel meta-heuristic method. *Comput Struct* 139:18–27
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, Kissimmee, 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: Dasgupta D, Michalewicz Z (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, USA
16. Arora JS (1989) Introduction to optimum design. McGraw-Hill, New York
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. AISC, Chicago, IL
21. Rajeev S, Krishnamoorthy CS (1992) Discrete optimization of structures using genetic algorithms. *Struct Eng ASCE* 118:1233–1250
22. Schutte JJ, Groenewold AA (2003) Sizing design of truss structures using particle swarms. *Struct Multidiscip Optim* 25:261–269
23. Erbatur F, Hasançebi O, Tütüncü I, Kılıç 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 Constr 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, 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 addressed. 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 is 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].

## 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 light.

### 8.2.1 Definitions and Concepts from Ray Theory

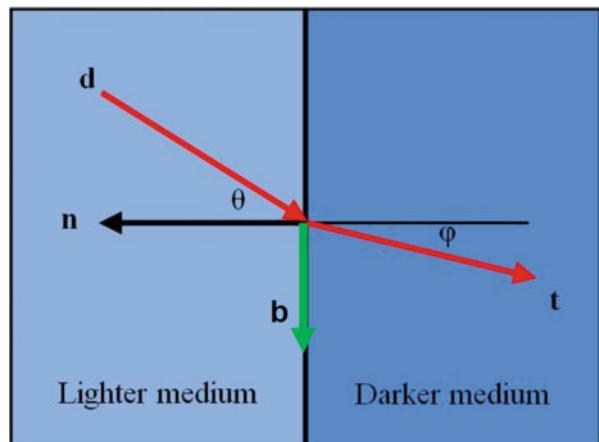
The contents of this section mainly follow the definitions and concepts presented in Ref. [4]. Certain transparent materials, so-called dielectric, refract the light. As the light travels through these materials, its path is changed according to the Snell's refraction law. Each transparent material has 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:

$$n_d \cdot \sin(\theta) = n_t \cdot \sin(\varphi). \quad (8.1)$$

where  $\theta$  and  $\varphi$  are the angles between the normal of two surfaces,  $n$ , 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  $t$ .

The computation for finding  $t$  is rather lengthy, but not difficult. To help us along the way, we use Fig. 8.1, where the vectors  $d$ ,  $n$ , and  $b$  are unit vectors. This makes the formula slightly less complicated. The steps for finding  $t$  in a two-dimensional space are outlined in the following:

**Fig. 8.1** Incident and refracted rays and their specifications [1]



1. We express  $\mathbf{t}$  in terms of  $\mathbf{n}$  and  $\mathbf{b}$ .

Let  $\mathbf{t}_n$  be the component of  $\mathbf{t}$  along  $\mathbf{n}$  and  $\mathbf{t}_b$  be the component of  $\mathbf{t}$  along  $\mathbf{b}$ . Then, since  $\mathbf{t}$  is a unit vector, we have:

$$\cos(\varphi) = \frac{\|\mathbf{t}_n\|}{\|\mathbf{t}\|} = \|\mathbf{t}_n\|. \quad (8.2)$$

Similarly,

$$\sin(\varphi) = \|\mathbf{t}_b\|. \quad (8.3)$$

Now

$$\mathbf{t}_n = -\|\mathbf{t}_n\| \cdot \mathbf{n}, \quad (8.4)$$

and

$$\mathbf{t}_b = \|\mathbf{t}_b\| \cdot \mathbf{b}. \quad (8.5)$$

Therefore,

$$\mathbf{t} = -\cos(\varphi) \cdot \mathbf{n} + \sin(\varphi) \cdot \mathbf{b}. \quad (8.6)$$

2. Express  $\mathbf{b}$  in terms of the known quantities using the fact that  $\mathbf{b}$  is the unit length vector parallel to the projection of  $\mathbf{d}$  onto the perpendicular to  $\mathbf{n}$ . Let  $\mathbf{d}_n$  be the component of  $\mathbf{d}$  along  $\mathbf{n}$  and  $\mathbf{d}_b$  be the component of  $\mathbf{d}$  along  $\mathbf{b}$ , the direction perpendicular to  $\mathbf{n}$ . Then,

$$\mathbf{d} = \mathbf{d}_n + \mathbf{d}_b. \quad (8.7)$$

Thus,

$$\mathbf{d}_b = \mathbf{d} - \mathbf{d}_n = \mathbf{d} - (\mathbf{d} \cdot \mathbf{n}) \cdot \mathbf{n}. \quad (8.8)$$

Also, since  $\mathbf{d}$  is a unit vector, we have:

$$\sin(\theta) = \frac{\|\mathbf{d}_b\|}{\|\mathbf{d}\|} = \|\mathbf{d}_b\|. \quad (8.9)$$

Thus,

$$\mathbf{b} = \frac{\mathbf{d}_b}{\|\mathbf{d}_b\|} = \frac{\mathbf{d} - (\mathbf{d} \cdot \mathbf{n}) \cdot \mathbf{n}}{\sin(\theta)}. \quad (8.10)$$

3. Using Eq. (8.1), one can express everything in terms of  $\mathbf{n}$ ,  $\mathbf{d}$ ,  $n_d$ , and  $n_r$ .

Therefore,

$$\begin{aligned} \mathbf{t} = -\cos(\varphi) \cdot \mathbf{n} + \sin(\varphi) \cdot \mathbf{b} &= -\cos(\varphi) \cdot \mathbf{n} + \sin(\varphi) \cdot \frac{\mathbf{d} - (\mathbf{d} \cdot \mathbf{n}) \cdot \mathbf{n}}{\sin(\theta)} = \\ &\quad -\cos(\varphi) \cdot \mathbf{n} + \frac{n_d}{n_t} \cdot (\mathbf{d} - (\mathbf{d} \cdot \mathbf{n}) \cdot \mathbf{n}) \end{aligned} \quad (8.11)$$

Now, we express  $\cos(\varphi)$  in terms of known quantities. Using the identity, we have:

$$\cos(\varphi) = \sqrt{1 - \sin^2(\varphi)}, \quad (8.12)$$

and employing Eq. (8.1), we obtain:

$$\cos(\varphi) = \sqrt{1 - \frac{n_d^2}{n_t^2} \cdot \sin^2(\theta)}. \quad (8.13)$$

Finally, we have:

$$\mathbf{t} = -\mathbf{n} \cdot \sqrt{1 - \frac{n_d^2}{n_t^2} \cdot \sin^2(\theta)} + \frac{n_d}{n_t} \cdot (\mathbf{d} - (\mathbf{d} \cdot \mathbf{n}) \cdot \mathbf{n}) \quad (8.14)$$

Here,  $\mathbf{t}$  is a normalized vector.

In tracing a ray in a two-dimensional space,  $\mathbf{d}$ ,  $\mathbf{t}$ , and  $\mathbf{n}$  were placed in  $z=0$  plane. In a three-dimensional space, it is clear that one can pass a plane through two vectors like  $\mathbf{n}$  and  $\mathbf{d}$ , which intersect each other at one point. Thus, the ray tracing in three-dimensional spaces is the extension of ray tracing in two-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  $i^*$  and  $j^*$ , then  $\mathbf{n}$  and  $\mathbf{d}$  can be rewritten in terms of these unit vectors. Finally, after finding  $\mathbf{t}$  in the new coordinate system, it can be rearranged based on the primary coordinate system. One of these new unit components can be  $\mathbf{n}$  as  $i^*$ . The other one can be found by the following relationship:

$$\mathbf{n} \cdot \mathbf{d} = \|\mathbf{n}\| \cdot \|\mathbf{d}\| \cdot \cos(\omega) = \cos(\omega) \quad (8.15)$$

where  $\omega$  is the angle between  $\mathbf{n}$  and  $\mathbf{d}$ . Now if

$$\mathbf{n} \cdot \mathbf{d} = 0 \quad (8.16)$$

then  $\omega = \pi/2$  and  $\mathbf{d}$  will be  $j^*$ , and if

$$0 < \mathbf{n} \cdot \mathbf{d} \leq 1 \quad (8.17)$$

then the direction of  $\mathbf{j}^*$  will be obtained by:  $(\mathbf{n} - \frac{\mathbf{d}}{\mathbf{n} \cdot \mathbf{d}})$ , since

$$\mathbf{n} \cdot \left( \mathbf{n} - \frac{\mathbf{d}}{\mathbf{n} \cdot \mathbf{d}} \right) = \mathbf{n} \cdot \mathbf{n} - \frac{\mathbf{n} \cdot \mathbf{d}}{\mathbf{n} \cdot \mathbf{d}} = 1 - 1 = 0 \quad (8.18)$$

And finally

$$\mathbf{j}^* = \frac{(\mathbf{n} - \frac{\mathbf{d}}{\mathbf{n} \cdot \mathbf{d}})}{\text{norm}(\mathbf{n} - \frac{\mathbf{d}}{\mathbf{n} \cdot \mathbf{d}})} \quad (8.19)$$

where norm is the function in MATLAB that provides the magnitude of a vector. Similarly, if

$$-1 \leq \mathbf{n} \cdot \mathbf{d} < 0 \quad (8.20)$$

$\mathbf{j}^*$  can be obtained by:

$$\mathbf{j}^* = \frac{(\mathbf{n} + \frac{\mathbf{d}}{-\mathbf{n} \cdot \mathbf{d}})}{\text{norm}(\mathbf{n} + \frac{\mathbf{d}}{-\mathbf{n} \cdot \mathbf{d}})}. \quad (8.21)$$

Now, in the new form,  $\mathbf{d}$  and  $\mathbf{n}$  are stated as:

$$\mathbf{n}^* = (1, 0), \quad (8.22)$$

And

$$\mathbf{d}^* = (\mathbf{d} \cdot \mathbf{i}^*, \mathbf{d} \cdot \mathbf{j}^*) \quad (8.23)$$

Therefore, after calculating  $\mathbf{t}^* = (t_1^*, t_2^*)$  in a two-dimensional space,  $\mathbf{t}$  in a three-dimensional space is obtained as:

$$\mathbf{t} = t_1^* \mathbf{i}^* + t_2^* \mathbf{j}^* \quad (8.24)$$

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

	$-0.05 \leq \mathbf{n} \cdot \mathbf{d} \leq 0.05$	$0.05 \leq \mathbf{n} \cdot \mathbf{d} \leq 1$	$-1 \leq \mathbf{n} \cdot \mathbf{d} \leq -0.05$
$\mathbf{i}^*$	$\mathbf{n}$	$\mathbf{n}$	$\mathbf{n}$
$\mathbf{j}^*$	$\mathbf{d}$	$\mathbf{j}^* = \frac{(\mathbf{n} - \frac{\mathbf{d}}{\mathbf{n} \cdot \mathbf{d}})}{\text{norm}(\mathbf{n} - \frac{\mathbf{d}}{\mathbf{n} \cdot \mathbf{d}})}$	$\mathbf{j}^* = \frac{(\mathbf{n} + \frac{\mathbf{d}}{-\mathbf{n} \cdot \mathbf{d}})}{\text{norm}(\mathbf{n} + \frac{\mathbf{d}}{-\mathbf{n} \cdot \mathbf{d}})}$

## 8.2.2 Ray Optimization Method

### 8.2.2.1 Scattering and Evaluation Step

Like every other population-based 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 with each group having two members, and then the corresponding agents are moved to their new positions. In other words, we move the first group to the new position based on a two-dimensional space, and also the second group is moved to the new position in another two-dimensional 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 two-dimensional spaces and one three-dimensional space for the movement. After the movements, we join them together to have a 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:

$$X_{ij} = X_{j,\min} + \text{rand.} (X_{j,\max} - X_{j,\min}). \quad (8.25)$$

where  $X_{ij}$  is the  $j$ th variable of the  $i$ th agent.  $X_{j,\min}$  and  $X_{j,\max}$  are the minimum and maximum limits of the  $j$ th variable, and  $\text{rand}$  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 abovementioned agents, a group of movement vectors should be assigned according to their division, and if the agent has a 3-variable group and two 2-variable groups, it must have a 3-variable movement vector group and two 2-variable movement vector groups, respectively. For the first movement, these vectors are obtained by:

$$v_{ij} = -1 + 2.rand. \quad (8.26)$$

where  $v_{ij}$  is the  $j$ th component of the  $i$ th 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  $<1$ . 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 as (1,0) and (0,1) for two-dimensional spaces and (1,0,0), (0,1,0), and (0,0,1) for three-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 components 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 best-so-far agent at this stage is selected as the global best, and for each agent, the best-so-far position by this stage is selected as its local best.

### 8.2.2.3 Origin Making and Convergence 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 the origin and it is specified by:

$$\mathbf{O}_i^k = \frac{(ite + k).\mathbf{GB} + (ite - k).\mathbf{LB}_i}{2.ite} \quad (8.27)$$

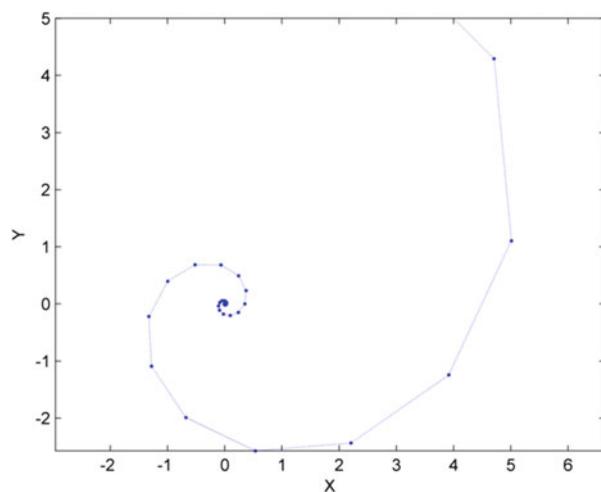
where  $\mathbf{O}_i^k$  is the origin of the  $i$ th agent for the  $k$ th iteration,  $ite$  is the total number of iterations for the optimization process, and  $\mathbf{GB}$  and  $\mathbf{LB}_i$  are the global best and local best of the  $i$ th agent, respectively. As Eq. (8.25) implies, at the 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 becomes 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  $\mathbf{O}$  and its end is the current position of agent, it should be anticipated that the agent will converge to  $\mathbf{O}$ . With refinement of  $\mathbf{O}$  during the optimization, the agents approach to the best result.

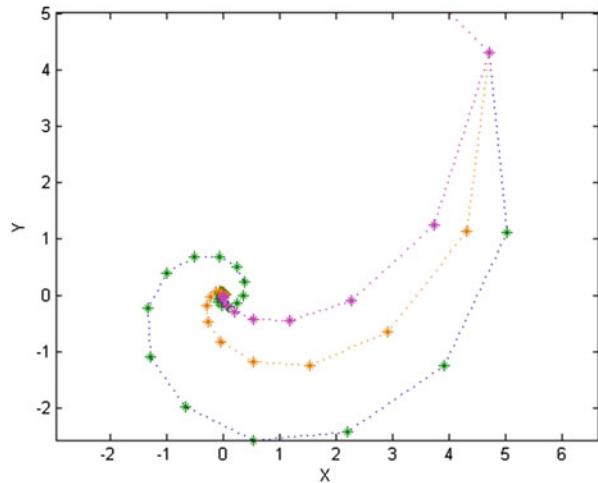
In order to show how the agents converge to a point, consider an agent with arbitrary position and movement vector in the two-dimensional space, like  $(4, 5)$  and  $(0.707, -0.707)$ , respectively. If this particle wants to move to a predefined point like the origin, it can be moved toward this point as illustrated in Fig. 8.2. It should be mentioned that some restrictions are imposed for this convergence. These consist of a fixed value of 0.6 as the ratio of index of 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 of 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 Eq. (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:

**Fig. 8.2** The movement of an agent to the origin [1]



**Fig. 8.3** Ratio of the index of refraction ratio of the magnet line, brown line, and blue line are 0.5, 0.65, and 0.85, respectively [1]



$$\mathbf{V}_{i,l} = \mathbf{V}_{i,l}' . norm(\mathbf{X}_{i,l} - \mathbf{O}_{i,l}). \quad (8.28)$$

where  $\mathbf{V}_{i,l}'$ ,  $\mathbf{X}_{i,l}$ ,  $\mathbf{O}_{i,l}$ , and  $\mathbf{V}_{i,l}$  are the normalized movement vector, current position of the agent, the origin, and refined movement vector of the  $i$ th agent, respectively, that belong to  $l$ th group.

In some cases it is possible that an agent,  $\mathbf{O}_{i,l}$  and its current position are the same, so the direction of the normal cannot be obtained. This problem occurs when the agent is the best-so-far 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:

$$\mathbf{V}_{i,l}^{k+1} = \frac{\mathbf{V}_{i,l}^k}{norm(\mathbf{V}_{i,l}^k)} . rand.0.001 \quad (8.29)$$

In this equation,  $\mathbf{V}_{i,l}^k$  is the movement vector of the  $k$ th iteration that belongs to  $l$ th group of the  $i$ th agent, and  $\mathbf{V}_{i,l}^{k+1}$  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 words, 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:

$$\mathbf{V}_{ijl}^{(k+1)} = -1 + 2.rand, \quad (8.30)$$

where  $\mathbf{V}_{ijl}^{(k+1)}$  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:

$$\mathbf{V}_{il}^{k+1} = \frac{\mathbf{V}_{il}^{(k+1)'}'}{norm(\mathbf{V}_{il}^{(k+1)'})} \cdot \frac{a}{d} \cdot rand \quad (8.31)$$

Here,  $a$  is calculated by the following relationship:

$$a = \sqrt{\sum_{i=1}^n (X_{i,\max} - X_{i,\min})^2} \quad n = \begin{cases} 2 & \text{for two variable groups} \\ 3 & \text{for three variable groups} \end{cases} \quad (8.32)$$

where  $X_{i,\max}$  and  $X_{i,\min}$  are the maximum and minimum limits of variables that belong to  $i$ th component of the movement vector and  $d$  is a number that divides  $a$  into smaller segments 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.

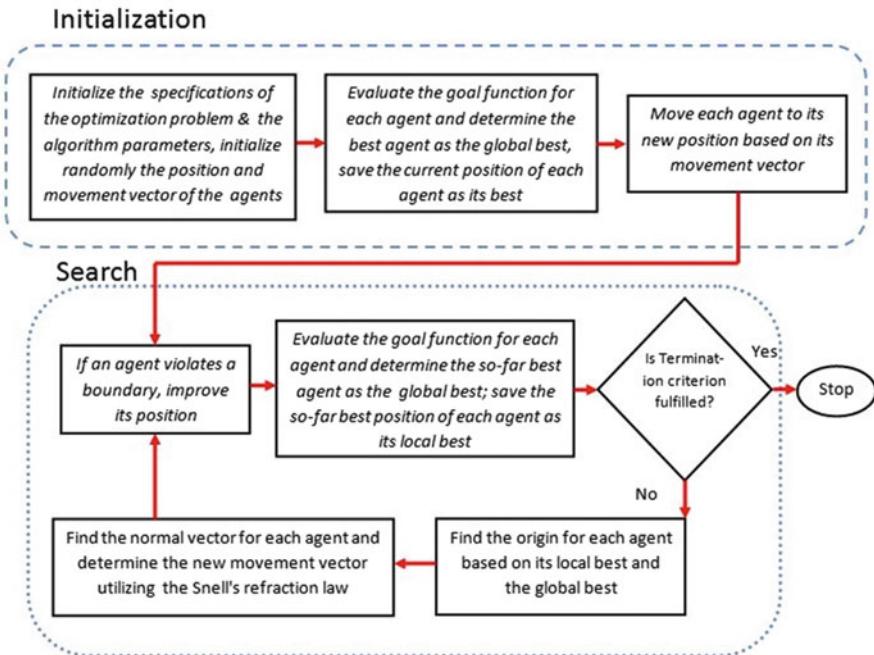
#### 8.2.2.4 Termination or Repetition Step

By approaching to a pre-specific criterion, the process of optimization ceases. Some criteria are considered as:

- Maximum number of iterations: by approaching to a predefined number of iteration, the process of optimization will be terminated.
- Number of ineffective iterations: 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 the 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 vectors, the agents will move to their new positions. This cycle is continued until a predefined criterion is fulfilled.

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 convergence 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



**Fig. 8.4** The flowchart of the RO [1]

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 Ref. [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 metaheuristics, considering a large number of agent causes vigorous search, but it

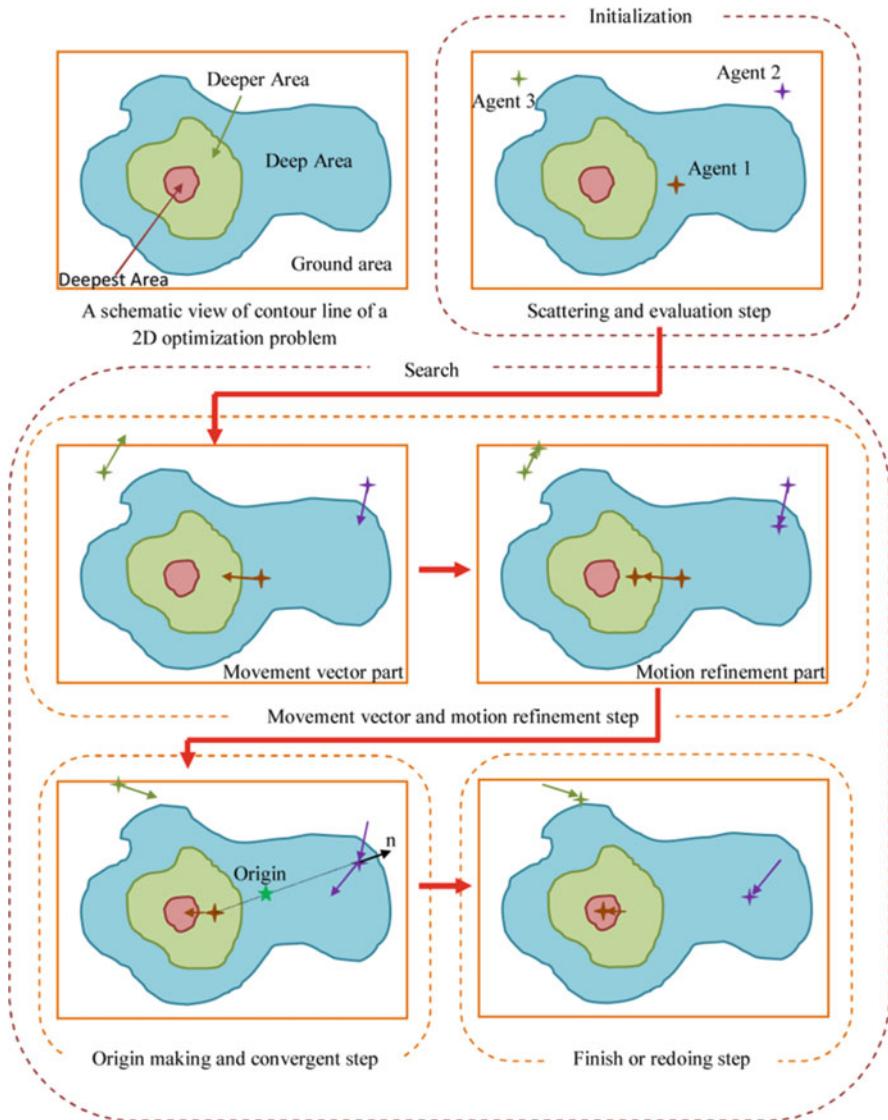
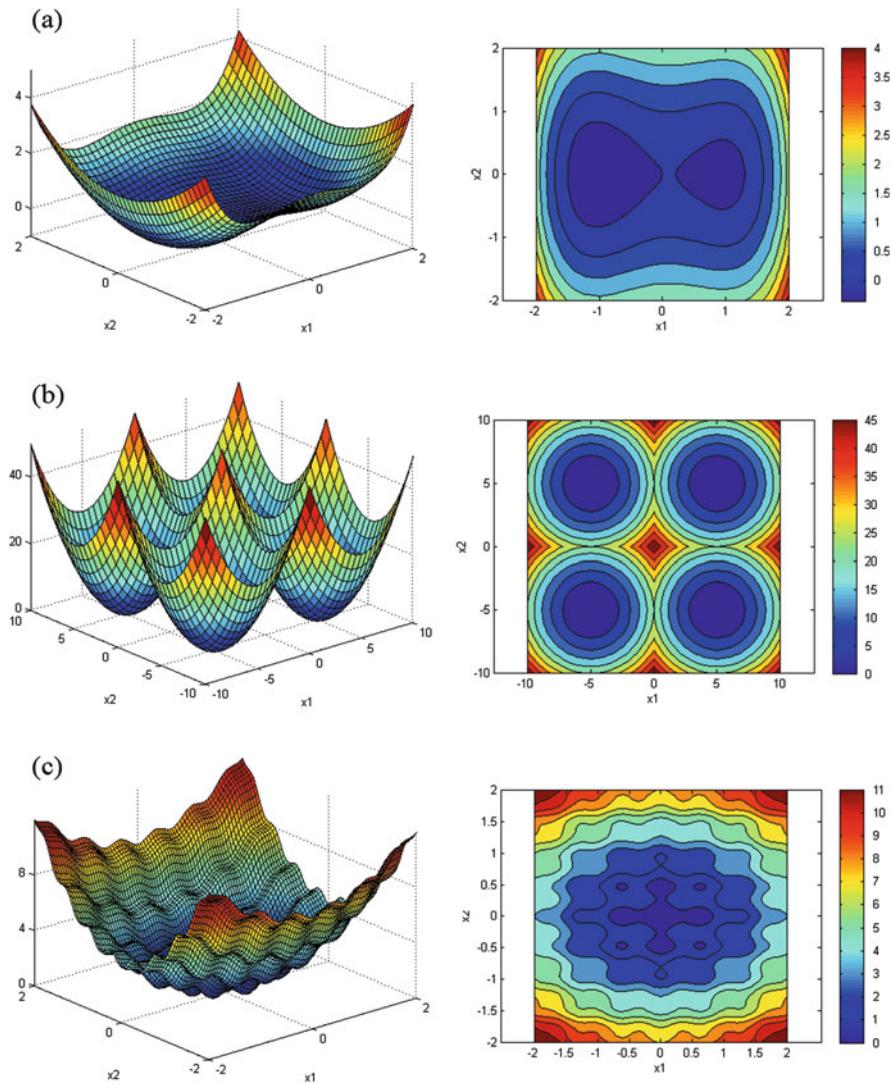


Fig. 8.5 Schematic view of the agent movement [1]

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 be mentioned that we used 100 agents for the EXP16, SHEKEL5, SHEKEL7, and 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

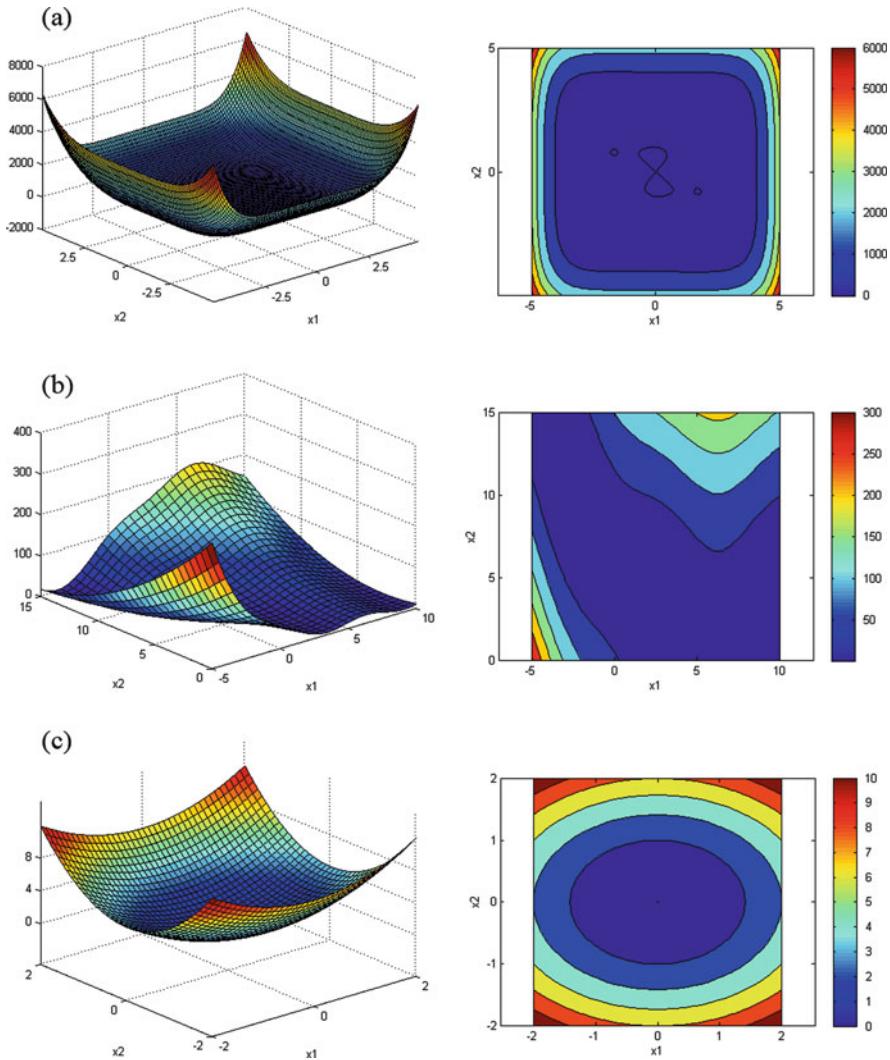
**Table 8.2** Specifications of the benchmark problems

Function name	Interval	Function	Global minimum
Aluffi-Pentini	$X \in [-10, 10]^2$	$f(X) = \frac{1}{4}x_1^4 - \frac{1}{2}x_1^2 + \frac{1}{10}x_1 + \frac{1}{4}x_2^2$	-0.352386
Bohachevsky 1	$X \in [-100, 100]^2$	$f(X) = x_1^2 + 2x_2^2 - \frac{3}{10} \cos(3\pi x_1) - \frac{4}{10} \cos(4\pi x_2) + \frac{7}{10}$	0.0
Bohachevsky 2	$X \in [-50, 50]^2$	$f(X) = x_1^2 + 2x_2^2 - \frac{3}{10} \cos(3\pi x_1) \cos(4\pi x_2) + \frac{3}{10}$	0.0
Becker and Lago	$X \in [-10, 10]^2$	$f(X) = ( x_1  - 5)^2 + ( x_2  - 5)^2$	0.0
Branin	$0 \leq x_2 \leq 15, -5 \leq x_1 \leq 10$	$f(X) = (x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1)^2 + 10(1 - \frac{1}{8\pi}) \cos(x_1) + 10$	0.397887
Camel	$X \in [-5, 5]^2$	$f(X) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	-1.0316
Cb3	$X \in [-5, 5]^2$	$f(X) = 2x_1^2 - 1.05x_1^4 + \frac{1}{6}x_1^6 + x_1x_2 + x_2^2$	0.0
Cosine mixture	$n=4, X \in [-1, 1]^n$	$f(X) = \sum_{i=1}^n x_i^2 - \frac{1}{10} \sum_{i=1}^n \cos(5\pi x_i)$	-0.4
De Jong	$X \in [-5.12, 5.12]^2$	$f(X) = x_1^2 + x_2^2 + x_3^2$	0.0
Exponential	$n=2, 4, 8, X \in [-1, 1]^n$	$f(X) = -\exp\left(-0.5 \sum_{i=1}^n x_i^2\right)$	-1.0
Goldstein and Price	$X \in [-2, 2]^2$	$f(X) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)[30 + (2x_1 - 3x_2)^2(18 - 32x_1 - 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2]$	3.0
Griewank	$X \in [-10, 100]^2$	$f(X) = 1 + \frac{1}{200} \sum_{i=1}^n x_i^2 - \prod_{i=1}^2 \cos\left(\frac{x_i}{\sqrt{i}}\right)$	0.0
Rastrigin	$X \in [-1, 1]^2$	$f(X) = \sum_{i=1}^n (x_i^2 - \cos(18x_i))$	-2.0



**Fig. 8.6** A perspective view and the related contour lines for some of the function when  $n = 2$  [1]: (a) Aluffi-Pentini, (b) Becker and Lago, (c) Bohachevsky 1

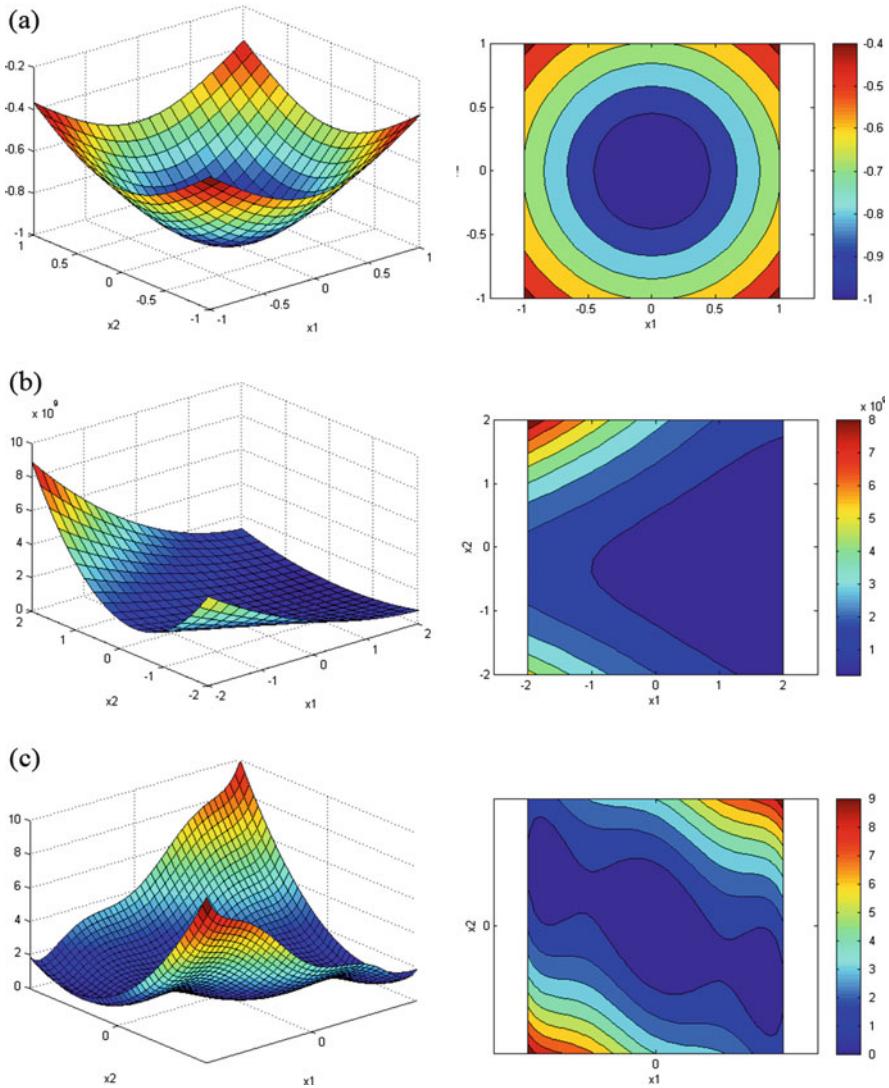
independent runs is chosen as 50. The numbers in the parentheses represent the ratio of successful runs in which the algorithm has found the global minimum with a predefined accuracy, which is taken as  $\epsilon = f_{\min} - f_{\text{final}} = 10^{-4}$ . The absence of the parentheses shows that the algorithm has been successful in all independent runs. 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 the function when  $n = 2$  [1]: (a) Camel, (b) Branin, (c) Bohachevsky 2

### 8.2.3.2 Engineering Design Problems

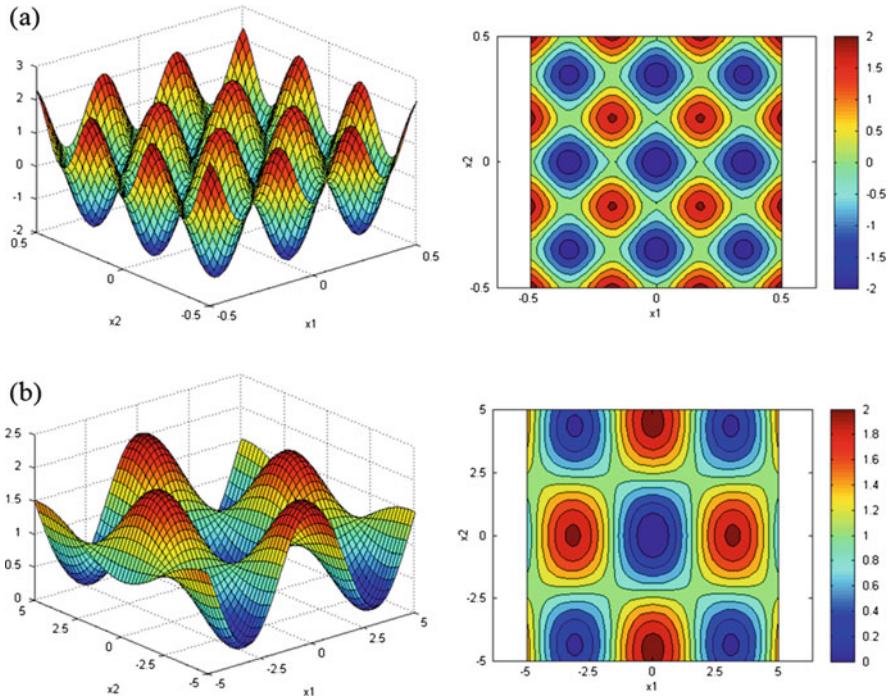
In this section, two engineering design problems are presented to show the efficiency of the RO. For the sake of simplicity, the penalty approach is used for constraint handling. In using the penalty function, if the constraints are not violated, the penalty will be zero; otherwise, the value of the penalty is calculated by dividing the violation of the allowable limit to the limit itself. It should be mentioned that in the RO, the use of this type of constraint handling is not a necessity, and any other types of constraint handling approach can be employed.



**Fig. 8.8** A perspective view and the related contour lines for some of the function when  $n = 2$  [1]: (a) Exponential, (b) Goldstein and Price, (c) Cb3

### A Tension/Compression Spring Design Problem

Belegundu [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



**Fig. 8.9** A perspective view and the related contour lines for some of the function when  $n = 2$  [1]: (a) Griewank, (b) Rastrigin

$d$ , and the number of active coils. Table 8.4 shows the cost function, the constraints, and the bounds of the design space.

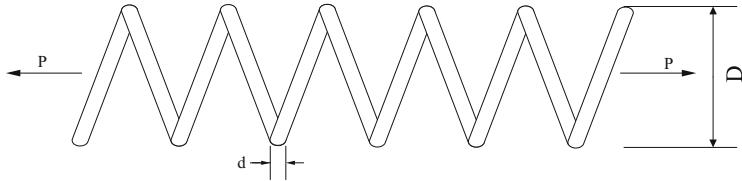
Belegundu [7] has solved this problem using eight different mathematical optimization techniques (only the best results are shown). Also, it has been solved by Arora [8] using a numerical optimization technique called constraint at the constant cost. Finally, this problem is solved by RO using 40 agents, 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 abovementioned methods. The average value and the standard deviation for 50 independents runs are 0.13547 and 0.001159, respectively.

### 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 ( $\tau$ ), bending stress ( $\sigma$ ), bulking load ( $P_c$ ), end deflection ( $\delta$ ), and side constraint. The design variables are  $h (=x_1)$ ,  $L (=x_2)$ ,  $t (=x_3)$ , and  $b (=x_4)$

**Table 8.3** Performance comparison for the benchmark problems

Function	GEN	GEN_S	GEN_S_M_LS	Kaveh and Khayatazad [1]
AP	1360 (0.99)	1360	1253	331
Bf1	3992	3356	1615	677
Bf2	20,234	3373	1636	582
BL	19,596	2412	1436	303
Branin	1442	1418	1257	463
Camel	1358	1358	1300	332
Cb3	9771	2045	1118	262
CM	2105	2105	1539	802
Dejoung	9900	3040	1281	452
EXP2	938	936	807	136
EXP4	3237	3237	1496	382
EXP8	3237	3237	1496	1287
EXP16	8061	8061	1945	17,236 (0.46)
Griewank	18,838 (0.91)	3111 (0.91)	1652 (0.99)	1091 (0.98)
Rastrigin	1533 (0.97)	1523 (0.97)	1381	1013 (0.98)
Goldstein and Price	1478	1478	1325	451
SHEKEL5	2527 (0.61)	2507 (0.61)	2049 (0.67)	3401 (0.52)
SHEKEL7	2567 (0.72)	2500 (0.72)	2032 (0.75)	3459 (0.76)
SHEKEL10	2641 (0.71)	2598 (0.71)	2141 (0.76)	4152 (0.66)
Total	114,815 (94.26)	49,655 (94.31)	28,759 (95.63)	36,812 (91.36)

**Fig. 8.10** Schematic of a tension–compression spring

(Fig. 8.11). Table 8.6 shows the cost function, the constraints, and the bounds of the design space.

This problem has been solved by Ragsdell and Phillips [9] and Deb [10]. Ragsdell and Phillips 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 (Davidon–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

**Table 8.4** Specifications of the tension/compression spring problem

Cost function	$f(X) = (x_3 + 2)x_2x_1^2$
Constraint functions	$g_1(X) = 1 - \frac{x_2^3x_3}{71785x_1^4} \leq 0$
	$g_2(X) = \frac{4x_2^4 - x_1x_2}{12566(x_2x_1^3 - x_1^4)} + \frac{1}{5108x_1^2} - 1 \leq 0$
	$g_3(X) = 1 - \frac{140.45x_1}{x_2^2x_3} \leq 0$
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

Methods	Optimal design variable			$f_{\text{cost}}$
	X <sub>1</sub> (d)	X <sub>2</sub> (D)	X <sub>3</sub> (N)	
Belegundu [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

50 independents runs. The results are collected in Table 8.7 indicating that the RO has a better solution than the abovementioned methods. The mean of the results and the standard deviation of 50 independent runs are 1.9083 and 0.173744, respectively.

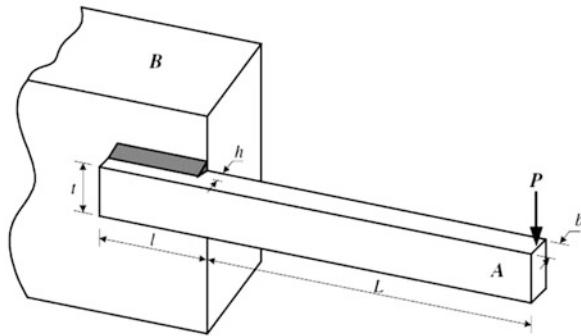
## 8.3 Ray Optimization for Size and Shape Optimization of Truss Structures

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** Schematic of a welded beam system



**Table 8.6** Specifications of a welded beam design problem

Cost function	$f(X) = 1.1047x_1^2x_2 + 0.04811x_3x_4(14.0 + x_2)$
Constraint functions	$g_1(X) = \tau(X) - \tau_{\max} \leq 0$ $g_2(X) = \sigma(X) - \sigma_{\max} \leq 0$ $g_3(X) = x_1 - x_4 \leq 0$ $g_4(X) = 0.1047x_1^2 + 0.04811x_3x_4(14.0 + x_2) - 5.0 \leq 0$ $g_5(X) = 0.125 - x_1 \leq 0$ $g_6(X) = \delta(X) - \delta_{\max} \leq 0$ $g_7(X) = P - P_c(X)^c \leq 0$
	$\tau(X) = \sqrt{(\tau')^2 + 2\tau'\tau''\frac{x_2}{2R} + (\tau'')^2}$
	$\tau' = \frac{P}{\sqrt{2x_1x_2}} \quad \tau'' = \frac{MR}{J}$
	$M = P\left(\frac{L+x_4}{2}\right), \quad R = \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1+x_3}{2}\right)^2}$
	$J = 2\left\{\sqrt{2}x_1x_2\left[\frac{x_2^2}{12} + \left(\frac{x_1+x_3}{2}\right)^2\right]\right\}$
	$\sigma(X) = \frac{6PL}{x_4x_3^2}, \quad \delta(X) = \frac{4PL^3}{Ex_4x_3^3}$
	$P_c(X)^c = \frac{4.013E\sqrt{\frac{x_2^2x_4}{36}}}{L^2}\left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right)$
	$P = 6000\text{lb}, \quad L = 14\text{in}$
	$E = 30 \times 10^6\text{psi}, \quad G = 12 \times 10^6\text{psi}$
	$\tau_{\max} = 13.6 \times 10^3\text{psi}, \quad \sigma_{\max} = 30 \times 10^3\text{psi}, \quad \delta_{\max} = 0.25\text{in}$
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				$f_{\text{cost}}$
	$X_1(h)$	$X_2(l)$	$X_3(t)$	$X_4(b)$	
Ragsdell 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

$$\begin{aligned}
 & \text{minimize} \quad W(\{x\}) = \sum_{i=1}^n \gamma_i A_i L_i(x) \\
 & \text{subject to : } \delta_{\min} \leq \delta_i \leq \delta_{\max}, \quad i = 1, 2, \dots, m \\
 & \sigma_{\min} \leq \sigma_i \leq \sigma_{\max}, \quad i = 1, 2, \dots, n \\
 & \sigma_i^b \leq \sigma_i^i \leq 0, \quad i = 1, 2, \dots, ns \\
 & A_{\min} \leq A_i \leq A_{\max}, \quad i = 1, 2, \dots, ng
 \end{aligned} \tag{8.33}$$

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),  $\gamma$  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 the cross-sectional area of the member  $i$  chosen between  $A_{\min}$  and  $A_{\max}$ ,  $\min$  shows the lower bound and  $\max$  indicates the upper bound,  $\sigma_i$  and  $\delta_i$  are the stress and nodal deflection, respectively, and  $\sigma_i^b$  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 the 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 iterations is considered as 400. In order to assess 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 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 types 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  $0.283 \text{ lb/in}^3$  ( $7933.410 \text{ kg/m}^3$ ) 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 twenty-nine groups, as shown in Table 8.11. The minimum cross-sectional area of all members is  $0.1 \text{ in}^2$  ( $0.6452 \text{ cm}^2$ ) and the maximum cross-sectional area is  $20 \text{ in}^2$  ( $129.03 \text{ cm}^2$ ).

The RO algorithm found the best weight as 25,193.228 lb after 326 iterations (32,600 analyses), with the standard deviation being 1221.146 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 is infinite span, the structure can be modeled as shown in Fig. 8.14a. Because of symmetry of this problem, one can analyze 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  $-140$  and  $140$  cm. Thus, there are ten variables for shape optimization. In the field of size optimization, the cross sections of each bar, with

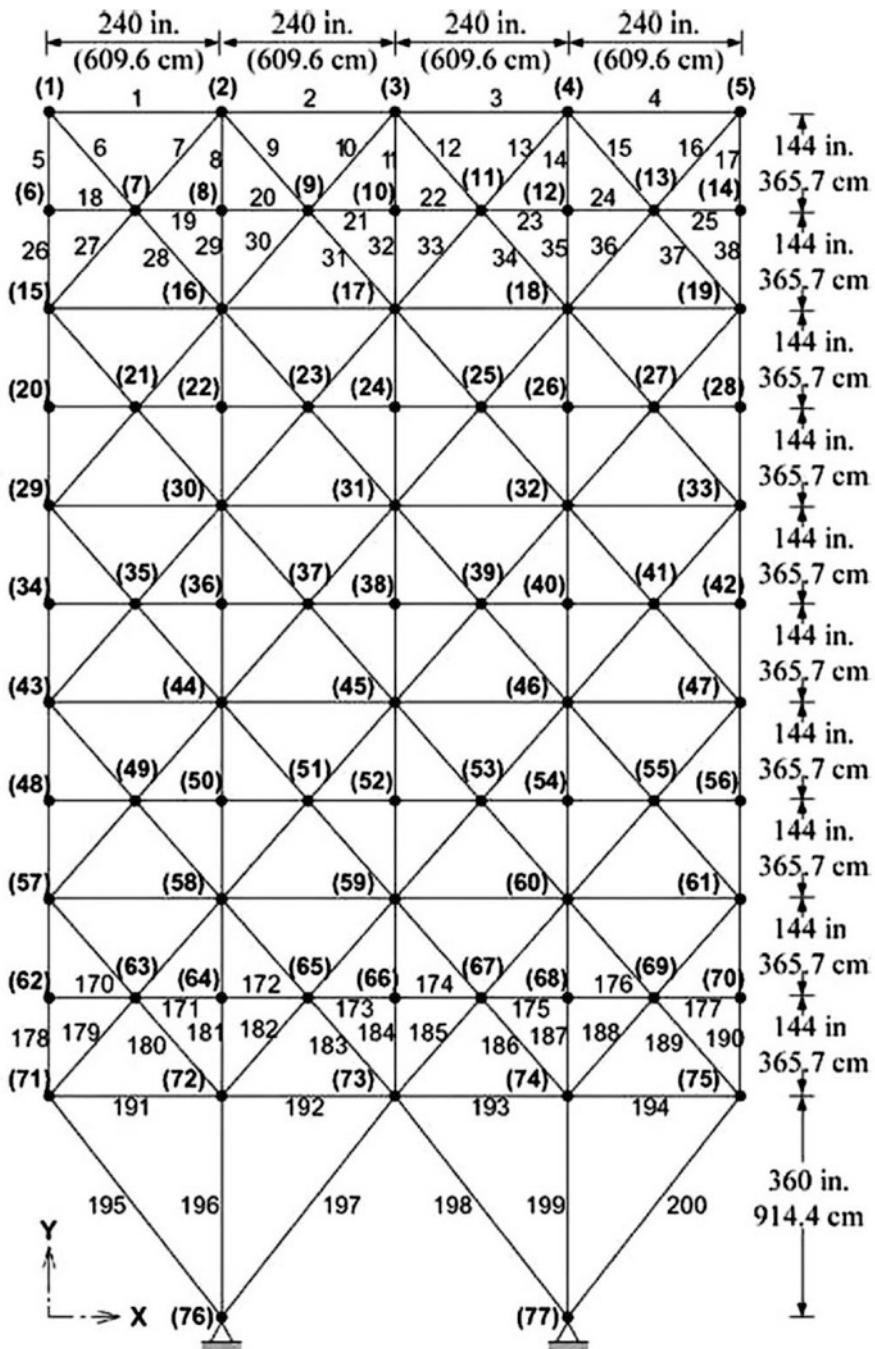
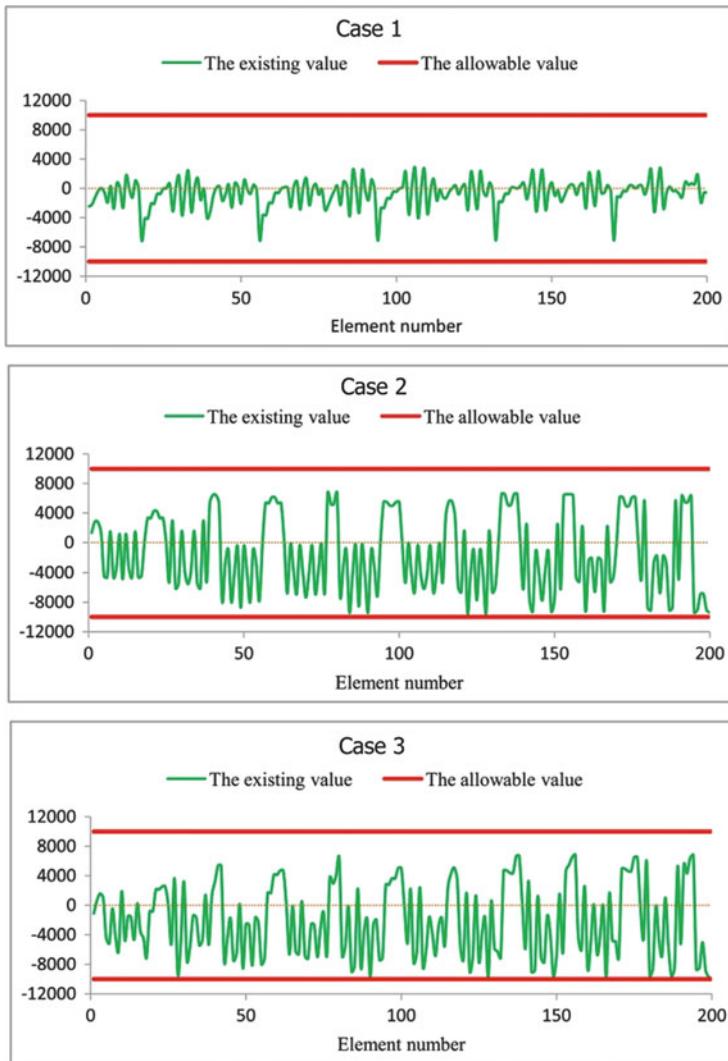


Fig. 8.12 Schematic of a 200-bar planar truss structure

**Table 8.8** Optimal design comparison for the 200-bar planar truss

Group	Variables members ( $A_i, i = 1, \dots, 200$ )	Optimal cross-sectional areas ( $\text{in}^2$ )			
		Kaveh and Talatahari [5]	HPSACO	( $\text{in}^2$ )	Kaveh and Khayatazzad [2] ( $\text{cm}^2$ )
1	1,2,3,4	0.8016	0.7590	0.1033	0.382
2	5,8,11,14,17	2.4028	0.9032	0.9184	2.116
3	19,20,21,22,23,24	4.3407	1.1000	0.1202	0.102
4	18,25,56,63,94,101,132,139,170,177	5.6972	0.9952	0.1009	0.141
5	26,29,32,35,38	3.9538	2.1350	1.8664	3.662
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
7	39,40,41,42	5.6080	1.0041	0.1000	0.121
8	43,46,49,52,55	9.1953	2.8052	2.9683	3.544
9	57,58,59,60,61,62	4.5128	1.0344	0.1000	0.108
10	64,67,70,73,76	4.6012	3.7842	3.9456	5.565
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
12	77,78,79,80	18.7510	0.4302	0.4501	0.138
13	81,84,87,90,93	5.9937	5.2683	4.96029	5.139
14	95,96,97,98,99,100	0.1000	0.9685	1.0738	0.101
15	102,105,108,111,114	8.1561	6.0473	5.9785	8.742
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
17	115,116,117,118	11.1520	0.5920	0.73743	0.998
18	119,122,125,128,131	7.1263	8.1858	7.3809	7.212
19	133,134,135,136,137,138	4.4650	1.0362	0.66740	0.152
20	140,143,146,149,152	9.1643	9.2062	8.3000	8.452
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
22	153,154,155,156	0.5541	1.8336	1.0000	0.413
23	157,160,163,166,169	16.1640	10.6110	10.8262	10.146
24	171,172,173,174,175,176	0.4974	0.9851	0.1000	0.874
					5.639

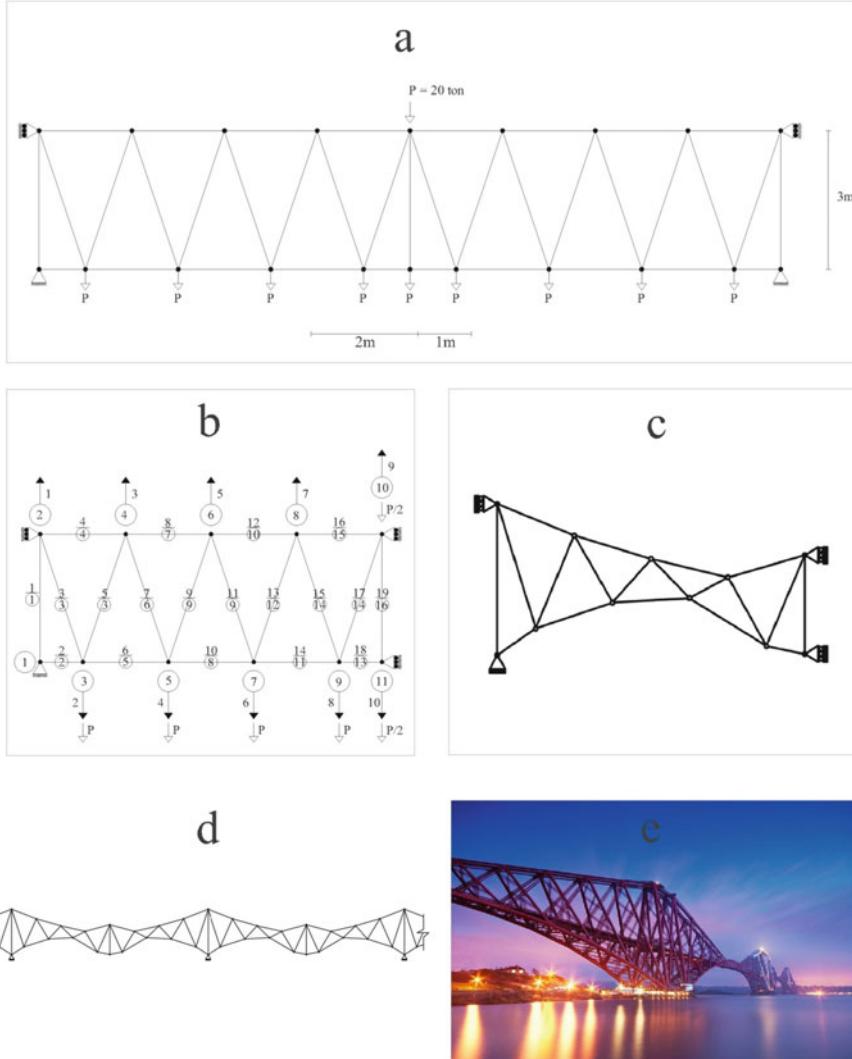
	Weight (lb)	178,181,184,187,190	16,2250	12,5090	11,6976	11,384	73,428
25		158,159,161,162,164,165,167,168,179,180,182,183,185,186,188,189	1.0042	1.9755	1.3880	1.197	7.721
26		191,192,193,194	3.6098	4.5149	4.9523	5.747	37.069
27		195,197,198,200	8.3684	9.8000	8.8000	7.823	50.461
28		196,199	15.5620	14.5310	14.6645	13.655	88.074
29			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]

their areas being  $0.5 \text{ cm}^2$  through  $100 \text{ cm}^2$ , are other design variables. In Fig. 8.14b, the encircled numbers show the element grouping, and there are 16 groups of elements for size optimization. The modulus of elasticity of material is  $2.1 \times 10^8 \text{ kN/m}^2$ , the allowable stress value is  $25 \text{ kN/cm}^2$ , and the specific weight of material is  $7.8 \text{ t/m}^3$ . 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 provided in Table 8.9. The standard deviation and average weight for 30 independent runs are

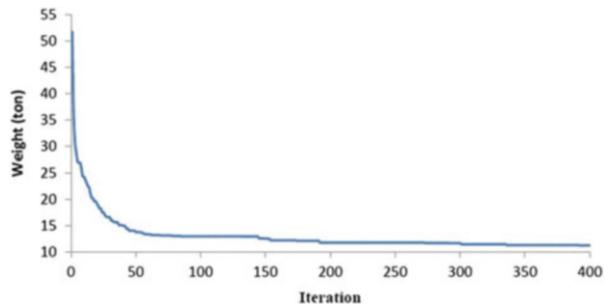


**Fig. 8.14** (a) Problem diagram. (b) Analytical model. (c) Optimal shape. (d) Optimized configuration formed by concatenating basic modules. (e) Schematic of the First-of-Forth Bridge, built during 1883–1890 [11]

545.5 kg and 11,969.2 kg, respectively. The prior work obtained 7102 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 2323.7 kg and 1235.4 kg, respectively. Finally, 25,269.3 kg and

**Table 8.9** Optimal cross sections and coordinates for the model of First-of-Forth Bridge

Element group	Optimal cross section ( $\text{cm}^2$ )			Coordinate variable	Optimal coordinate (cm)		
	BB–BC	PSO	Kaveh and Khayatazad [2]		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)	1235.4	2323.7	545.5

**Fig. 8.15** Convergence curve for the model of the First-of-Forth Bridge [2]

40,154.1 kg are obtained as the average weights for 30 independent runs for the PSO and BB–BC.

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 end and middle is greater than the other sections. Therefore, for resisting 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 the difference is the construction requirements instead of saving material.

### 8.3.2.3 A 37-Bar Simply Supported Truss

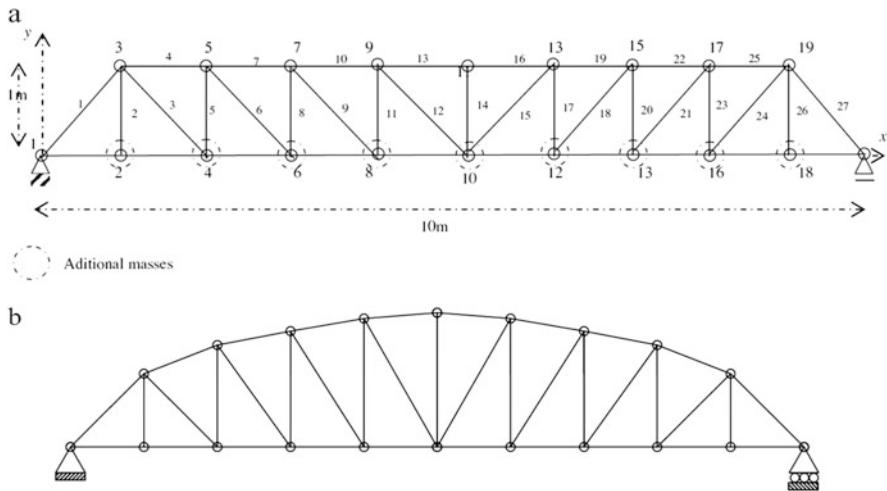
This example has been investigated by Wang et al. [12] using the evolutionary node shift method and by Lingyun et al. [13] utilizing the NHGA algorithm. It is a simple supported Pratt-type 37-bar truss as shown in Fig. 8.16a. There are nonstructural masses of  $m = 10 \text{ kg}$  attached to each of the bottom nodes of the lower chord, which are modeled as bar elements with rectangular cross-sectional areas of  $4 \times 10^{-3} \text{ m}^2$ . The other bars are modeled as simple bar elements in the field of size optimization with the range of  $1 \times 10^{-4} \text{ m}^2$  through  $3.5 \times 10^{-4} \text{ m}^2$ . The material property for the bar elements is selected as  $E = 2.1 \times 10^{11} \text{ N/m}^2$  and  $\rho = 7800 \text{ kg/m}^3$ . Also, this example is considered as a truss shape optimization since all nodes of the upper chord are allowed to vary in the y-axis in a symmetrical manner in the range 1 m through 2.5 m. There are three constraints in the first three natural frequencies as  $\omega_1 = 20 \text{ Hz}$ ,  $\omega_2 = 40 \text{ Hz}$ , and  $\omega_3 = 60 \text{ Hz}$ . 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 among the optimal design cross sections of several methods including the present work (RO). As it can be seen, the best results for the RO, Wang et al. [12], Lingyun et al. [13], and 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 curve 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

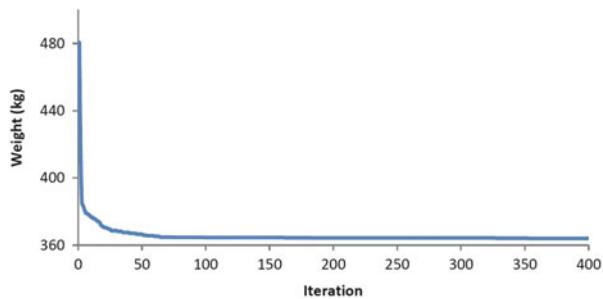


**Fig. 8.16** (a) Schematic of a 37-bar truss structure with added masses. (b) Schematic of the 37-bar structure optimized by the present work [12]

**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
$A_1, A_{27}$ ( $\text{cm}^2$ )	3.2508	2.8932	2.6797	3.0124
$A_2, A_{26}$ ( $\text{cm}^2$ )	1.2364	1.1201	1.1568	1.0623
$A_3, A_{24}$ ( $\text{cm}^2$ )	1.0000	1.0000	2.3476	1.0005
$A_4, A_{25}$ ( $\text{cm}^2$ )	2.5386	1.8655	1.7182	2.2647
$A_5, A_{23}$ ( $\text{cm}^2$ )	1.3714	1.5962	1.2751	1.6339
$A_6, A_{21}$ ( $\text{cm}^2$ )	1.3681	1.2642	1.4819	1.6717
$A_7, A_{22}$ ( $\text{cm}^2$ )	2.4290	1.8254	4.6850	2.0591
$A_8, A_{20}$ ( $\text{cm}^2$ )	1.6522	2.0009	1.1246	1.6607
$A_9, A_{18}$ ( $\text{cm}^2$ )	1.8257	1.9526	2.1214	1.4941
$A_{10}, A_{19}$ ( $\text{cm}^2$ )	2.3022	1.9705	3.8600	2.4737
$A_{11}, A_{17}$ ( $\text{cm}^2$ )	1.3103	1.8294	2.9817	1.5260
$A_{12}, A_{15}$ ( $\text{cm}^2$ )	1.4067	1.2358	1.2021	1.4823
$A_{13}, A_{16}$ ( $\text{cm}^2$ )	2.1896	1.4049	1.2563	2.4148
$A_{14}$ ( $\text{cm}^2$ )	1.0000	1.0000	3.3276	1.0034
Weight (kg)	366.50	368.84	377.20	364.04

**Fig. 8.17** Convergence curve of the RO for the simply supported 37-bar truss with added masses [2]



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

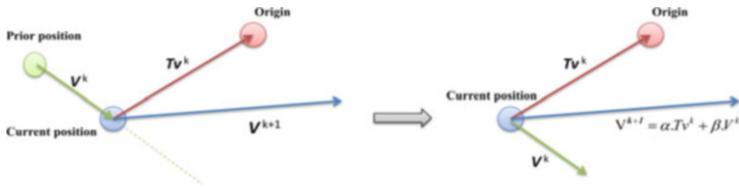
In the improved ray optimization algorithm, a memory which saves some or all the historically best positions of agents is considered as local best memory (*LBM*). If the best-so-far positions of all agents are saved (especially when the number of agents is large), the computational cost grows. Therefore, in this technique, when the number of agents is 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 refined. This violated component must be regenerated by the following formula:

$$X_{ij}^{k+1} = X_{ij}^k + 0.9 \left( Int_{ij} - X_{ij}^k \right) \quad (8.34)$$

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 3, the proposed formula cannot be applied directly, and first the main problem must be divided into some subproblems and after the calculation merge the results of the subproblems 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.



**Fig. 8.18** Generation of the new movement vector [3]

$$T\mathbf{v}_i = O_i - X_i \quad (8.35)$$

$$\mathbf{V}_i^{k+1} = \alpha.T\mathbf{v}_i^k + \beta.\mathbf{V}_i^k \quad (8.36)$$

where  $T\mathbf{v}_i$  is the target vector and  $\mathbf{V}_i^{k+1}$  and  $\mathbf{V}_i^k$  are movement vectors in  $(k+1)$ th and  $k$ th iteration, respectively.  $O_i^k$  is the origin according to Eq. (8.25), but in this method,  $LB$  is considered randomly from the local best memory.  $\alpha$  and  $\beta$  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,  $\alpha$  and  $\beta$  are increasing and decreasing functions respectively and are defined as

$$\alpha = 1 + \left( \frac{k}{ite} \right) \quad (8.37)$$

$$\beta = 1 - 0.5 \left( \frac{k}{ite} \right) \quad (8.38)$$

Finally, all the  $\mathbf{V}_i^{k+1}$  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:

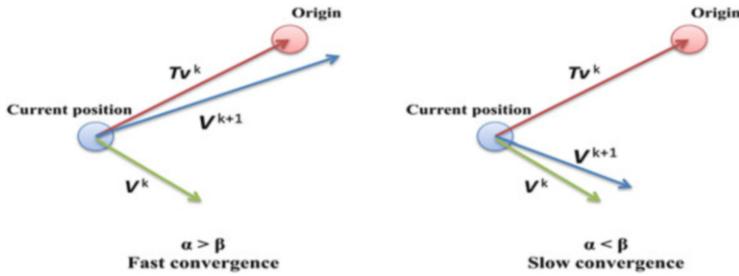
a. With probability *stoch*

$$\mathbf{V}_{ij}^{k+1} = -1 + 2.rand \quad (8.39)$$

$$\mathbf{V}_i^{K+1} = \frac{\mathbf{V}_i^{k+1}}{norm(\mathbf{V}_i^{k+1})} \cdot \frac{a}{d} \cdot rand \quad (8.40)$$

b. With probability (*1-stoch*),

If  $norm(O_i^k - X_i^k) = 0$ ,



**Fig. 8.19** Influence of the  $\alpha$  and  $\beta$  parameters [3]

$$V_i^{k+1} = \frac{V_i^k}{\text{norm}(V_i^k)} \cdot \text{rand}.0.001 \quad (8.41)$$

Otherwise,

$$V_i^{k+1} = V_i^{k+1} \cdot \text{norm}(X_i^k - O_i^k) \quad (8.42)$$

In the case of problems that have function constraints (behavior constraints), the following formulas are utilized instead of Eq. (8.40):

$$V_i^{K+1} = V_i^{k+1} \cdot \frac{a}{d} \quad (8.43)$$

$$d = d + r.d. \left( \frac{k}{ite} \right) \quad (8.44)$$

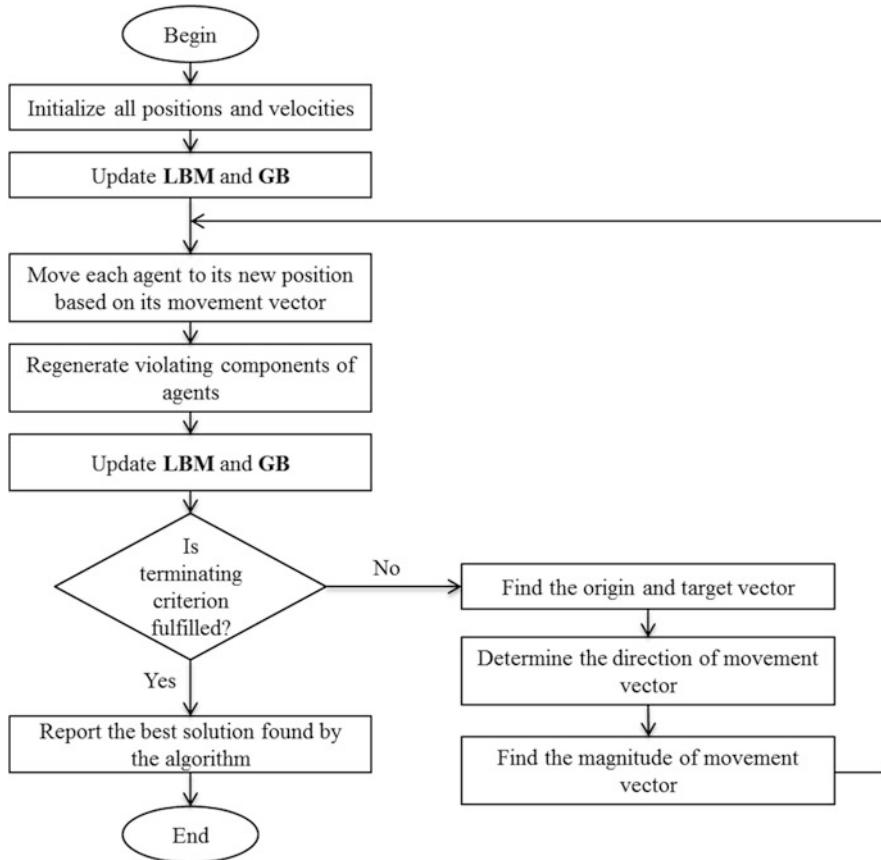
where  $r$  is a constant factor, and when the number of iterations rises, the value of  $d$  increases and it helps the algorithm to handle the constraint well.  $a$  and  $d$  are the factors that were defined in Sect. 8.2.2.3.

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 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.



**Fig. 8.20** The flowchart of the IRO [3]

The IRO method with different numbers of agents has been tested for some of these functions. Assigning the number of agents as 50 in the CM, Griewank, and Rastrigin functions and as 10 for the 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 modification of GA [6], the standard RO [1], and the 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.

**Table 8.11** Performance comparison for the benchmark problems

Function	GEN_S_M_LS	Ray optimization	Kaveh et al. [3]
AP	1253	331	$253 \pm 38.7985$ (100)
Bf1	1615	677	$438 \pm 48.4636$ (100)
Bf2	1636	582	$395 \pm 45.9674$ (100)
BL	1436	303	$194 \pm 31.2733$ (100)
Branin	1257	463	$312 \pm 81.0515$ (100)
Camel	1300	332	$184 \pm 21.0855$ (100)
Cb3	1118	262	$247 \pm 36.4549$ (100)
CM	1539	802	$1290 \pm 65.3543$ (100)
Dejoung	1281	452	$213 \pm 26.3344$ (100)
EXP2	807	136	$90 \pm 20.5115$ (100)
EXP4	1496	382	$220 \pm 50.5624$ (100)
EXP8	1496	1287	$512 \pm 97.7743$ (100)
EXP16	1945	17,236 (0.46)	$1141 \pm 142.76$ (100)
Grienwank	1652 (0.99)	1091 (0.98)	$1383 \pm 100.3458$ (100)
Rastrigin	1381	1013 (0.98)	$1662 \pm 202.3105$ (100)
Goldstein and Price	1325	451	$361 \pm 59.0105$ (100)
Total	22,537 (99.94)	25,800 (96.38)	8895 (100)

#### 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. Penalty function formula and constraint conditions for truss structures are briefly overviewed at the first subsection, and 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 is 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

$$f_{\text{cost}}(\{X\}) = (1 + \varepsilon_1 \cdot v)^{\varepsilon_2} \times W(\{X\}), \quad v = \sum_{j=1}^n \max[0, g_j(\{X\})] \quad (8.45)$$

$$g_j(\{X\}) \leq 0, \quad j = 1, 2, \dots, n \quad (8.46)$$

**Table 8.12** Algorithm parameters for truss examples

Parameter	25-bar	72-bar	120-bar
d	15	15	10
r	7	7	20

where  $W(\{x\})$  is the weight of the structure (Sect. 8.3.1),  $v$  denotes the sum of the violations of the design,  $g_j(\{X\})$  denotes design constraints, and  $n$  represents the number of constraints. The constants  $\varepsilon_1$  and  $\varepsilon_2$  are selected considering the exploration and the exploitation rate of the search space. Here,  $\varepsilon_1$  is set to unity;  $\varepsilon_2$  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,  $\varepsilon_2$  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:

$$\begin{cases} \sigma_i^+ = 0.6 F_y & \text{for } \sigma_i \geq 0 \\ \sigma_i^- & \text{for } \sigma_i < 0 \end{cases} \quad (8.47)$$

$$\sigma_i^- = \begin{cases} \left[ \left( 1 - \frac{\lambda_i^2}{2 C_c^2} \right) F_y \right] / \left( \frac{5}{3} + \frac{3\lambda_i}{8 C_c} + \frac{\lambda_i^3}{8 C_c^3} \right) & \text{for } \lambda_i \geq C_c \\ \frac{12 \pi^2 E}{23 \lambda_i^2} & \text{for } \lambda_i \leq C_c \end{cases} \quad (8.48)$$

where  $E$  is the modulus of elasticity,  $F_y$  is the yield stress of steel,  $C_c$  denotes the slenderness ratio ( $\lambda_i$ ) dividing the elastic and inelastic buckling regions ( $C_c = \sqrt{2\pi^2 E/F_y}$ ),  $\lambda_i$  = 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].

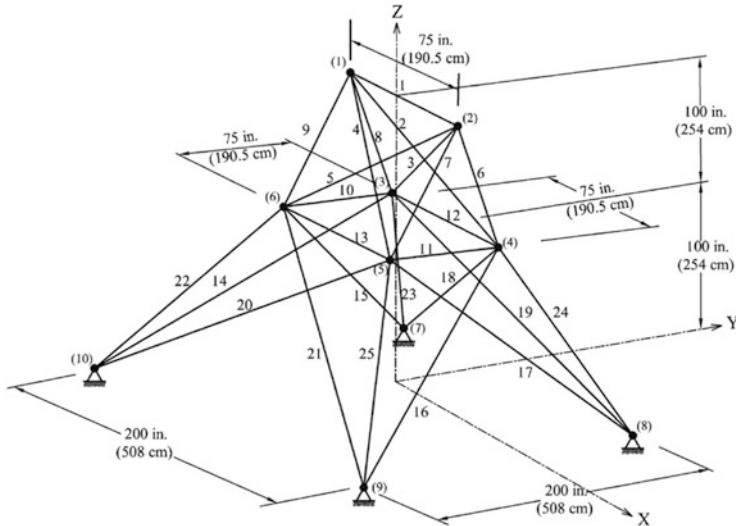
The other constraint corresponds to the limitation of the nodal displacements:

$$\delta_i - \delta_i^u \leq 0 \quad i = 1, 2, \dots, nn \quad (8.49)$$

where  $\delta_i$  is the nodal deflection,  $\delta_i^u$  is the allowable deflection of node  $i$ , and  $nn$  is the number of nodes.

### The 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 are shown in Fig. 8.21, [17]. The material density is considered as 0.1 lb/in<sup>3</sup>



**Fig. 8.21** Schematic of a 25-bar space truss

( $2767.990 \text{ kg/m}^3$ ), and the modulus of elasticity is taken as  $10^7 \text{ psi}$  ( $68,950 \text{ MPa}$ ). 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,  $\pm 40 \text{ ksi}$  ( $275.80 \text{ MPa}$ ) for each member and  $\pm 0.35 \text{ in}$  ( $\pm 8.89 \text{ mm}$ ) for each node in the  $x$ ,  $y$ , and  $z$  directions. The range of discrete cross-sectional areas is from 0.1 to  $3.4 \text{ in}^2$  ( $0.6452-21.94 \text{ cm}^2$ ) with  $0.1 \text{ in}^2$  ( $0.6452 \text{ cm}^2$ ) increment (resulting in 34 discrete cross sections) for each of the eight element groups [18].

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 performs better than others when the number of analyses and average weight for 50 runs are compared.

### The 72-Bar Space Truss with Discrete Variables

For the 72-bar spatial truss structure shown in Fig. 8.22 [20], the material density is  $0.1 \text{ lb/in}^3$  ( $2767.990 \text{ kg/m}^3$ ), and the modulus of elasticity is  $10^7 \text{ psi}$  ( $68,950 \text{ MPa}$ ). 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)  $A_{49}-A_{52}$ ,

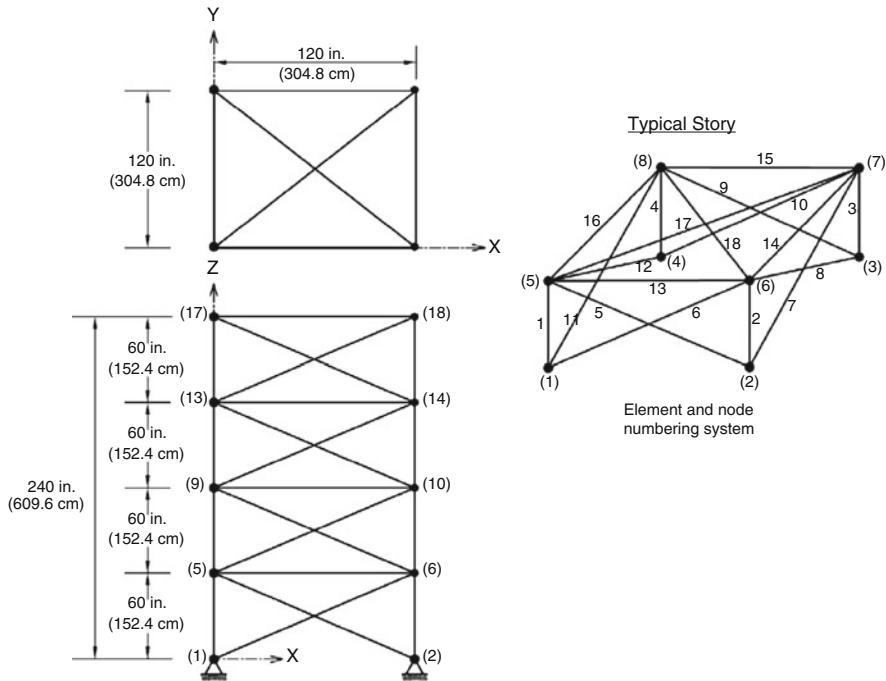
**Table 8.13** Performance comparison for the 25-bar spatial truss

Element group		Optimal cross-sectional areas (in <sup>2</sup> )					
		GA [18]	GA [18]	ACO [19]	BB–BC phases 1, 2 [18]	Kaveh et al. [3]	
		in <sup>2</sup>	cm <sup>2</sup>				
1	A <sub>1</sub>	0.10	0.10	0.10	0.10	0.10	0.645
2	A <sub>2</sub> –A <sub>5</sub>	1.80	0.50	0.30	0.30	0.30	1.935
3	A <sub>6</sub> –A <sub>9</sub>	230	3.40	3.40	3.40	3.40	21.935
4	A <sub>10</sub> –A <sub>11</sub>	020	0.10	0.10	0.10	0.10	0.645
5	A <sub>12</sub> –A <sub>13</sub>	010	1.90	2.10	2.10	2.10	13.548
6	A <sub>14</sub> –A <sub>17</sub>	0.80	0.90	1.00	1.00	1.00	6.452
7	A <sub>18</sub> –A <sub>21</sub>	.80	0.50	0.50	0.50	0.50	3.226
8	A <sub>22</sub> –A <sub>25</sub>	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)
No. of analyses		800	15,000	7700	9000	925	

(12) A<sub>53</sub>–A<sub>54</sub>, (13) A<sub>55</sub>–A<sub>58</sub>, (14) A<sub>59</sub>–A<sub>66</sub> (15), A<sub>67</sub>–A<sub>70</sub>, and (16) A<sub>71</sub>–A<sub>72</sub>. In this example, designs for 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  $\pm 25$  ksi ( $\pm 172.375$  MPa). Maximum displacement limitations of  $\pm 0.25$  in ( $\pm 6.35$  mm) are imposed on every node in every direction and on the uppermost nodes in both x and y directions, respectively.

In this case, the discrete variables are selected from 64 discrete values from 0.111 to 33.5 in<sup>2</sup> (71.613–21,612.860 mm<sup>2</sup>). 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 the 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, 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 algorithm [21]. The curves for the best result and the average penalized weight of 50 runs are shown in Fig. 8.23. Convergence speed in IRO is acceptable, and steplike movements in the 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  $389.87 \pm 1.1643$  to  $408.17 \pm 71.2108$  considering 47 and all 50 independent runs, respectively. In 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.



**Fig. 8.22** Schematic of a 72-bar space truss

**Table 8.14** Multiple loading conditions for the 72-bar truss

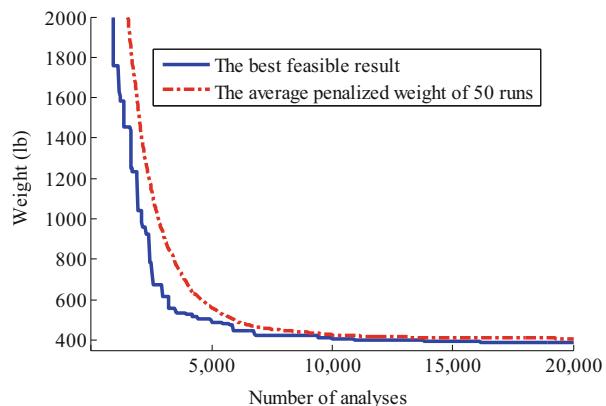
Case	Node	$F_x$ kips (kN)	$F_y$ kips (kN)	$F_z$ kips (kN)
1	17	0.0	0.0	-5.0 (-22.25)
	18	0.0	0.0	-5.0 (-22.25)
	19	0.0	0.0	-5.0 (-22.25)
	20	0.0	0.0	-5.0 (-22.25)
2	17	5.0 (22.25)	5.0 (22.25)	-5.0 (-22.25)

### Design of the 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 of the element groups are numbered in this figure. The 120 members are categorized into seven groups, because of symmetry. Other conditions of the 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<sup>3</sup> (7971.810 kg/m<sup>3</sup>). 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 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

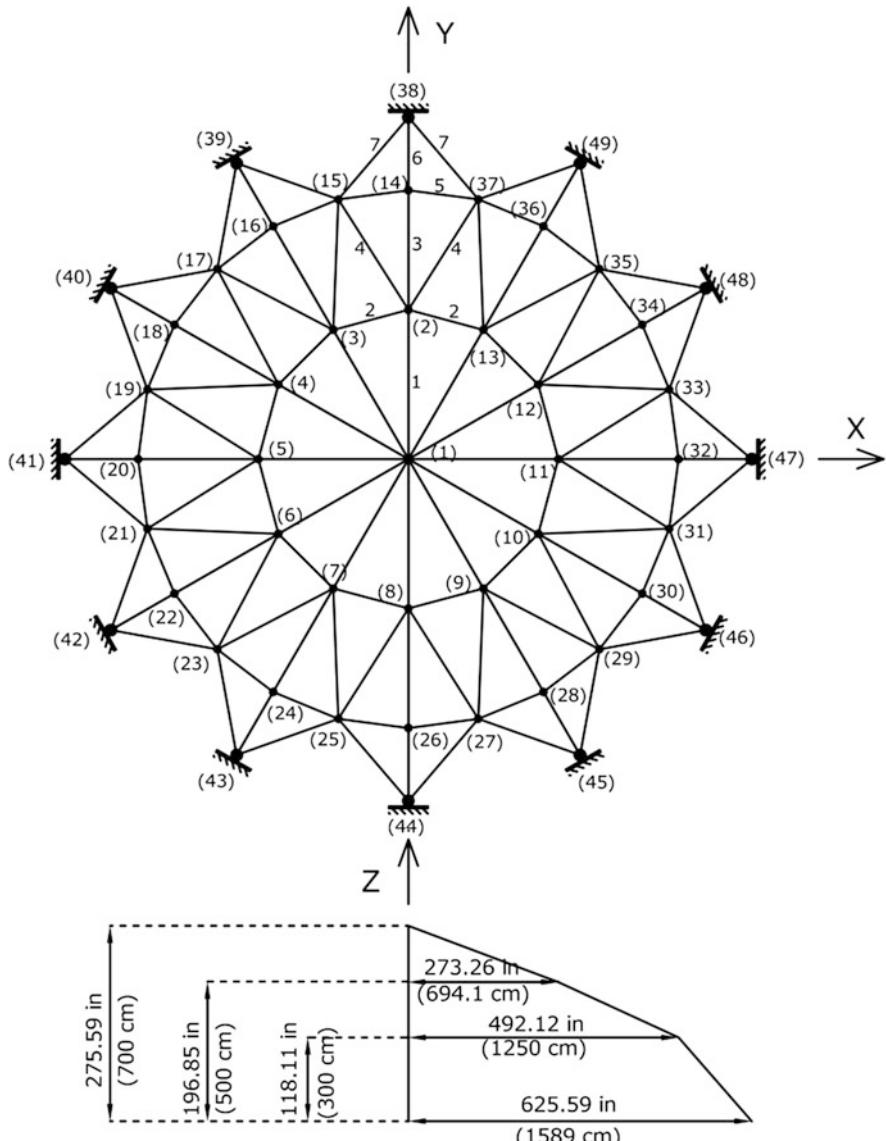
**Table 8.15** Performance comparison for the 72-bar spatial truss with discrete variable

Element group	Optimal cross-sectional areas ( $\text{in}^2$ )							
	GA [21]	PSOPC [21]	HPSO [21]	HPSACO [22]	ICA [21]	Kaveh et al. [3] $\text{in}^2$		
1	A <sub>1</sub> –A <sub>4</sub>	0.196	4.490	4.970	1.800	1.99	1.99	12.839
2	A <sub>5</sub> –A <sub>12</sub>	0.602	1.457	1.228	0.442	0.442	0.563	3.632
3	A <sub>13</sub> –A <sub>16</sub>	0.307	0.111	0.111	0.141	0.111	0.111	0.716
4	A <sub>17</sub> –A <sub>18</sub>	0.766	0.111	0.111	0.111	0.141	0.111	0.716
5	A <sub>19</sub> –A <sub>22</sub>	0.391	2.620	2.880	1.228	1.228	1.228	7.923
6	A <sub>23</sub> –A <sub>30</sub>	0.391	1.130	1.457	0.563	0.602	0.563	3.632
7	A <sub>31</sub> –A <sub>34</sub>	0.141	0.196	0.141	0.111	0.111	0.111	0.716
8	A <sub>35</sub> –A <sub>36</sub>	0.111	0.111	0.111	0.111	0.141	0.111	0.716
9	A <sub>37</sub> –A <sub>40</sub>	1.800	1.266	1.563	0.563	0.563	0.563	3.632
0	A <sub>41</sub> –A <sub>48</sub>	0.602	1.457	1.228	0.563	0.563	0.442	2.852
1	A <sub>49</sub> –A <sub>52</sub>	0.141	0.111	0.111	0.111	0.111	0.111	0.716
2	A <sub>53</sub> –A <sub>54</sub>	0.307	0.111	0.196	0.250	0.111	0.111	0.716
3	A <sub>55</sub> –A <sub>58</sub>	1.563	0.442	0.391	0.196	0.196	0.196	1.265
4	A <sub>59</sub> –A <sub>66</sub>	0.766	1.457	1.457	0.563	0.563	0.563	3.632
5	A <sub>67</sub> –A <sub>70</sub>	0.141	1.228	0.766	0.442	0.307	0.391	2.523
6	A <sub>71</sub> –A <sub>72</sub>	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)
	No. of analyses	N/A	150,000	50,000	5330	4500	17,925	

**Fig. 8.23** Convergence curves for the 72-bar space truss [3]

minimum cross-sectional area of all members is  $0.775 \text{ in}^2$  ( $5 \text{ cm}^2$ ), and the maximum cross-sectional area is taken as  $20.0 \text{ in}^2$  ( $129.032 \text{ cm}^2$ ). The constraints are stress constraints [as defined in Eqs. (8.45) and (8.46)] and displacement limitations of  $\pm 0.1969$  in ( $\pm 5 \text{ mm}$ ), 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

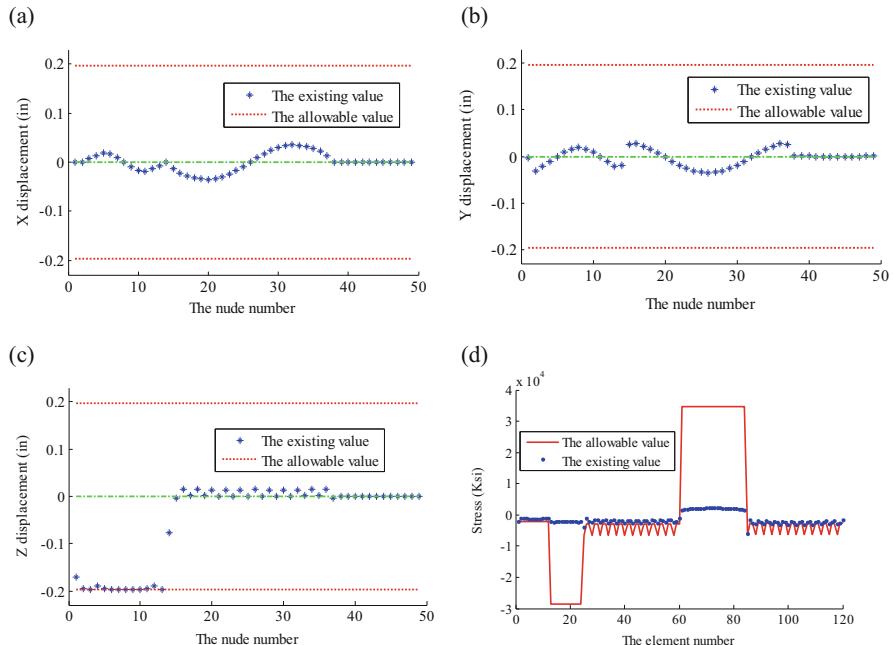


**Fig. 8.24** Schematic of a 120-bar dome-shaped truss

find the best solution while this number is equal to 150,000, 32,600, 10,000, 10,000, 7000, and 6000 analyses for a PSO-based algorithm [5]; a PSO and ACO hybrid 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.

**Table 8.16** Performance comparison for the 120-bar dome-shaped truss with continuous variables

Optimal cross-sectional areas ( $\text{in}^2$ )						
Element group	PSOPC [5]	PSACO [5]	HPSACO [20]	HBB-BC [77]	CSS [21]	ICA Kaveh et al. [3]
1 A <sub>1</sub>	3,040	3,026	3,095	3,037	3,027	3,0252
2 A <sub>2</sub>	13,149	15,222	14,405	14,431	14,606	14,8354
3 A <sub>3</sub>	5,646	4,904	5,020	5,130	5,044	5,2446
4 A <sub>4</sub>	3,143	3,123	3,352	3,134	3,139	3,1413
5 A <sub>5</sub>	8,759	8,341	8,631	8,591	8,543	8,4541
6 A <sub>6</sub>	3,758	3,418	3,432	3,377	3,367	3,3567
7 A <sub>7</sub>	2,502	2,498	2,499	2,500	2,497	2,4947
Best weight (lb)	33,481.2	33,263.9	33,248.9	33,287.9	33,251.9	33,256.2
Average weight (lb)	N/A	N/A	N/A	N/A	N/A	33,280.85
No. of analyses	150,000	32,600	10,000	10,000	7000	6000
					18,300	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

Comparing the final results of the IRO and those of the other metaheuristics shows that IRO finds a design close 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 %.

## 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 Civil Eng 57(2):97–112
4. Laval Philippe B (2003) Mathematics for computer graphics-ray tracing III. Kennesaw State University, Georgia, Nov 12

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, USA
8. Arora JS (1989) Introduction to optimum design. McGraw-Hill, New York
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. AISC, 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 Constr 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 MBB–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 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:

$$A_i^{c(k)} = \frac{\sum_{j=1}^N \frac{1}{Mer^j} \cdot A_i^{(k,j)}}{\sum_{j=1}^N \frac{1}{Mer^j}} \quad i = 1, 2, \dots, ng \quad (9.1)$$

where  $A_i^{(k,j)}$  is the  $i$ th component of the  $j$ th solution generated in the  $k$ th iteration and  $N$  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 offsprings 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:

$$A_i^{(k+1,j)} = A_i^{c(k)} + \frac{r_j \alpha_1 (A_{\max} - A_{\min})}{k + 1} \quad i = 1, 2, \dots, ng \quad (9.2)$$

where  $r_j$  is a random number from a standard normal distribution which changes for each candidate and  $\alpha_1$  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 are 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:

$$\begin{cases} \sigma_i^{\min} < \sigma_i < \sigma_i^{\max} & \Rightarrow \Phi_{\sigma}^{(i)} = 0 \\ \sigma_i^{\min} > \sigma_i \text{ or } \sigma_i^{\max} < \sigma_i & \Rightarrow \Phi_{\sigma}^{(i)} = \frac{\sigma_i - \sigma_i^{\min/\max}}{\sigma_i^{\min/\max}} \end{cases} \quad i = 1, 2, \dots, n \quad (9.3)$$

$$\begin{cases} \sigma_b < \sigma_i < 0 & \Rightarrow \Phi_{\sigma b}^{(i)} = 0 \\ \sigma_i < 0 \wedge \sigma_i < \sigma_b & \Rightarrow \Phi_{\sigma b}^{(i)} = \frac{\sigma_i - \sigma_b}{\sigma_b} \end{cases} \quad i = 1, 2, \dots, ns \quad (9.4)$$

$$\begin{cases} \delta_i^{\min} < \delta_i < \delta_i^{\max} & \Rightarrow \Phi_{\delta}^{(i)} = 0 \\ \delta_i^{\min} > \delta_i \text{ or } \delta_i^{\max} < \delta_i & \Rightarrow \Phi_{\delta}^{(i)} = \frac{\delta_i - \delta_i^{\min/\max}}{\delta_i^{\min/\max}} \end{cases} \quad i = 1, 2, \dots, m \quad (9.5)$$

In optimizing structures, the main objective is to find the minimum amount of the merit function. This function is defined as [4]:

$$Mer^k = \varepsilon_1 \cdot W^k + \varepsilon_2 \cdot (\Phi_{\sigma}^k + \Phi_{\delta}^k + \Phi_{\sigma b}^k)^{\varepsilon_3} \quad (9.6)$$

$Mer^k$  is the merit function for the  $k$ th candidate;  $\varepsilon_1$ ,  $\varepsilon_2$ , and  $\varepsilon_3$  are coefficients of merit function; and  $\Phi_{\sigma}^k$ ,  $\Phi_{\delta}^k$ , and  $\Phi_{\sigma b}^k$  are 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:

$$Mer^k = \varepsilon_1 \cdot W^k + \varepsilon_2 \cdot \sum_{i=1}^{np} (\Phi_{\sigma(i)}^k + \Phi_{\delta(i)}^k + \Phi_{\sigma b(i)}^k)^{\varepsilon_3} \quad (9.7)$$

where  $np$  is the number of multiple loadings. In this part, for a better control on other parameters,  $\varepsilon_1$  is set to 1. The coefficient  $\varepsilon_2$  is taken as the weight of the structure, and the coefficient  $\varepsilon_3$  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,  $\varepsilon_3$  is set to 1.5 but gradually it is increased to 3 (Ref. [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 toward 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:

$$A_i^{(k+1,j)} = \alpha_2 A_i^{c(k)} + (1 - \alpha_2) \left( \alpha_3 A_i^{gbest(k)} + (1 - \alpha_3) A_i^{lbest(k,j)} \right) + \frac{r_j \alpha_1 (A_{\max} - A_{\min})}{k + 1} \quad \begin{cases} i = 1, 2, \dots, ng \\ j = 1, 2, \dots, N \end{cases} \quad (9.8)$$

where  $A_i^{lbest(k,j)}$  is the best position of the  $j$ th particle up to the iteration  $k$ ,  $A_i^{gbest(k)}$  is the best position among all candidates up to the iteration  $k$ , and  $\alpha_2$  and  $\alpha_3$  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 suboptimization 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 subdomains 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:

$$\begin{cases} A_{\min,i}^{(k_{SOM})} = A_i^{gbest(k_{SOM}-1)} - \beta_1 \cdot (A_{\max,i}^{(k_{SOM}-1)} - A_{\min,i}^{(k_{SOM}-1)}) \geq A_{\min,i}^{(k_{SOM}-1)} \\ A_{\max,i}^{(k_{SOM})} = A_i^{gbest(k_{SOM}-1)} + \beta_1 \cdot (A_{\max,i}^{(k_{SOM}-1)} - A_{\min,i}^{(k_{SOM}-1)}) \leq A_{\max,i}^{(k_{SOM}-1)} \end{cases} \quad \begin{cases} i = 1, 2, \dots, ng \\ k_{SOM} = 2, \dots, nc \end{cases} \quad (9.9)$$

where  $\beta_1$  is an adjustable factor which determines the amount of the remaining search space and in this research it is taken as 0.3 (Ref. [6]) and  $A_{\min,i}^{(k_{SOM})}$  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:

$$A_{\min,i}^{(1)} = A_{\min}, \quad A_{\max,i}^{(1)} = A_{\max} \quad i = 1, 2, \dots, ng \quad (9.10)$$

**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  $\beta_2$ , and therefore the amount of the accuracy rate of each variable is equal to:

$$A_i^{*(k_{SOM})} = \frac{(A_{\max,i}^{(k_{SOM})} - A_{\min,i}^{(k_{SOM})})}{\beta_2 - 1} \quad i = 1, 2, \dots, ng \quad (9.11)$$

where  $A_i^{*(k_{SOM})}$  is the amount of increase in allowable cross-sectional area; Unlike ACO,  $\beta_2$  (the number of subdomains) does not affect the optimization time, and in the BB–BC optimization,  $\beta_2$  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:

$$A_{\min,i}^{(k_{SOM})}, \quad A_{\min,i}^{(k_{SOM})} + A_i^{*(k_{SOM})}, \quad \dots, \quad A_{\min,i}^{(k_{SOM})} + (\beta_2 - 1) \cdot A_i^{*(k_{SOM})} = A_{\max,i}^{(k_{SOM})} \quad i = 1, 2, \dots, ng \quad (9.12)$$

**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 criterion for SOM can be described as:

$$A_i^{*(nc)} \leq A^* \quad i = 1, 2, \dots, ng \quad (9.13)$$

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.

Suboptimization 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.

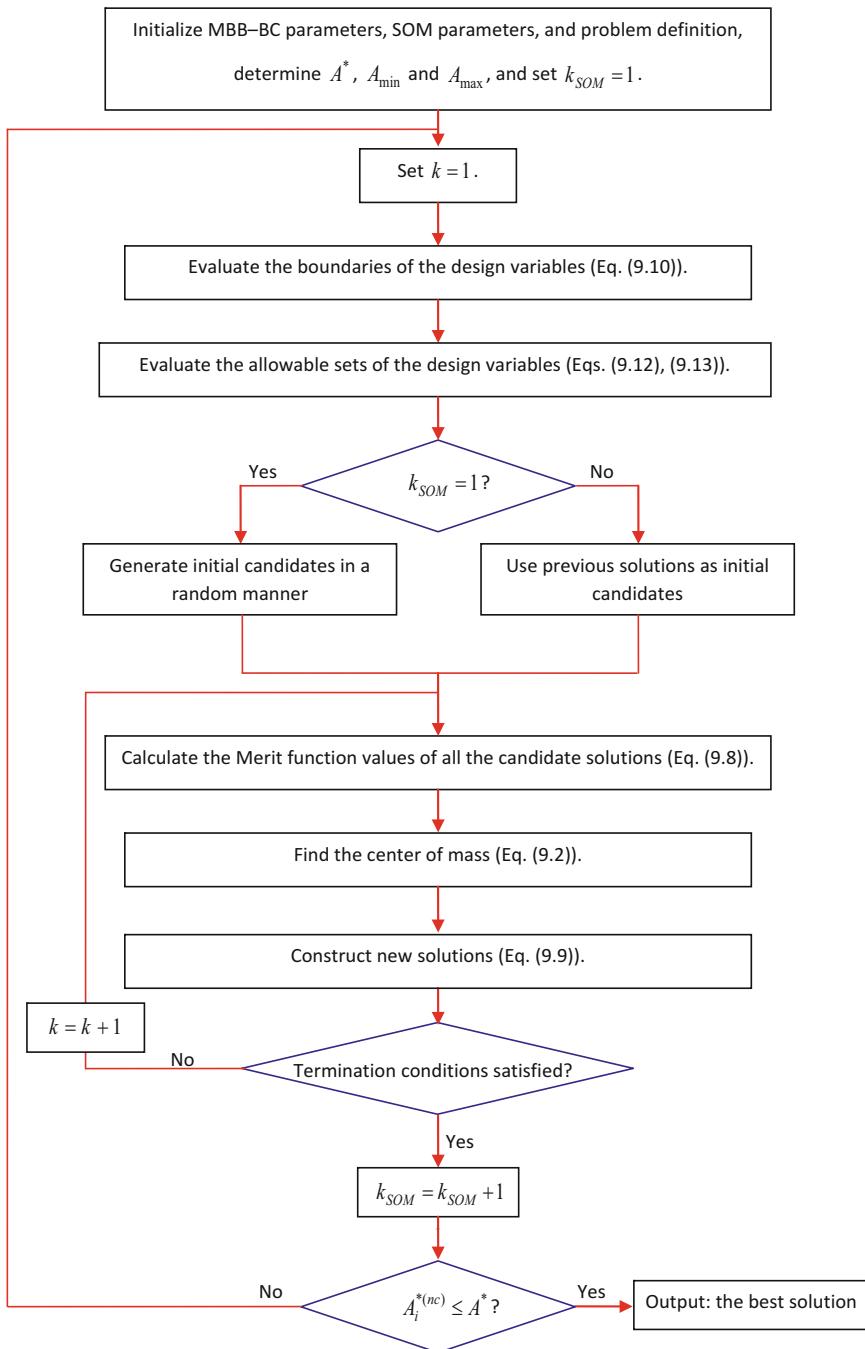
## 9.3 Size Optimization of Space Trusses Using a MBB–BC Algorithm

### 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:

$$\begin{aligned} \text{minimize} \quad & W(\{x\}) = \sum_{i=1}^n \gamma_i \cdot A_i \cdot L_i \\ \text{subject to :} \quad & \delta_{\min} \leq \delta_i \leq \delta_{\max} \quad i = 1, 2, \dots, m \\ & \sigma_{\min} \leq \sigma_i \leq \sigma_{\max} \quad i = 1, 2, \dots, n \\ & \sigma_i^b \leq \sigma_i \leq 0 \quad i = 1, 2, \dots, ns \\ & A_{\min} \leq A_i \leq A_{\max} \quad i = 1, 2, \dots, ng \end{aligned} \quad (9.14)$$

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);  $\gamma_i$  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  $\max = \text{upper bound}$ ;  $\sigma_i$  and  $\delta_i$  are the stress and nodal deflection, respectively; and  $\sigma_i^b$  is the allowable buckling stress in member  $i$  when it is in compression.

**Fig. 9.1** The flowchart for the MBB–BC algorithm

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 suboptimization 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.

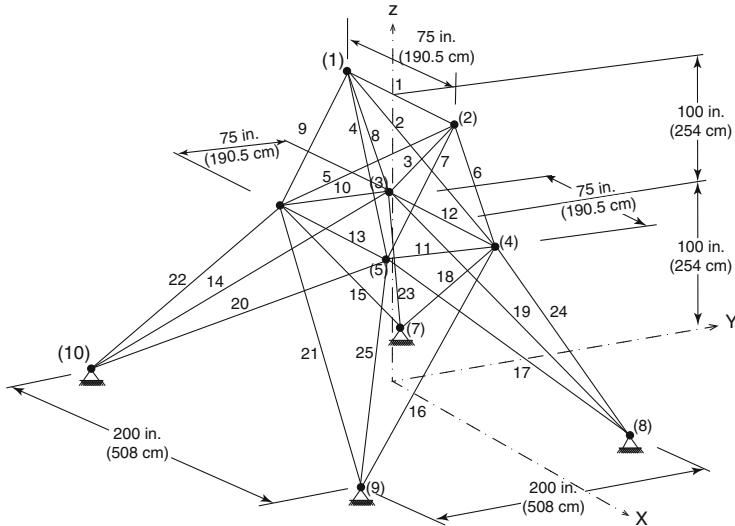
### **9.3.2 Design Examples**

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 the 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. The last example is optimized by the MBB–BC optimization for 20 times, while performance comparisons of the MBB–BC method in other examples are based on 50 evaluations. The performance of the present algorithm in the first example is compared to the simple and improved heuristic approaches, in the second example it is compared to the simple heuristic algorithms, and in the third example, some improved approaches are considered from the literature. Example



**Fig. 9.2** Schematic of a 25-bar spatial truss

4 is optimized using GA, PSO, a hybrid PSO (PSOPC [7]), BB-BC, and the MBB-BC method. The 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.

### 9.3.2.1 A 25-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 in Refs. [1, 24, 25, 8–13]. In these studies, the material density is considered as 0.1 lb/in<sup>3</sup> (2767.990 kg/m<sup>3</sup>), 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<sub>1</sub>, (2) A<sub>2</sub>–A<sub>5</sub>, (3) A<sub>6</sub>–A<sub>9</sub>, (4) A<sub>10</sub>–A<sub>11</sub>, (5) A<sub>12</sub>–A<sub>13</sub>, (6) A<sub>14</sub>–A<sub>17</sub>, (7) A<sub>18</sub>–A<sub>21</sub>, and (8) A<sub>22</sub>–A<sub>25</sub>

This spatial truss is subjected to two loading conditions shown in Table 9.1. Maximum displacement limitations of  $\pm 0.35$  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<sup>2</sup> (0.6452–21.94 cm<sup>2</sup>).

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  $\alpha_2$  and  $\alpha_3$  on the average weight of designs for the 25-bar truss. This figure shows that  $\alpha_2 = 0.40$  and  $\alpha_3$

**Table 9.1** Loading conditions for the 25-bar spatial truss

Node	Case 1			Case 2		
	P <sub>X</sub> kips (kN)	P <sub>Y</sub> kips (kN)	P <sub>Z</sub> kips (kN)	P <sub>X</sub> kips (kN)	P <sub>Y</sub> kips (kN)	P <sub>Z</sub> 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 <sub>1</sub>	35.092 (241.96)	40.0 (275.80)
2	A <sub>2</sub> ~A <sub>5</sub>	11.590 (79.913)	40.0 (275.80)
3	A <sub>6</sub> ~A <sub>9</sub>	17.305 (119.31)	40.0 (275.80)
4	A <sub>10</sub> ~A <sub>11</sub>	35.092 (241.96)	40.0 (275.80)
5	A <sub>12</sub> ~A <sub>13</sub>	35.092 (241.96)	40.0 (275.80)
6	A <sub>14</sub> ~A <sub>17</sub>	6.759 (46.603)	40.0 (275.80)
7	A <sub>18</sub> ~A <sub>21</sub>	6.959 (47.982)	40.0 (275.80)
8	A <sub>22</sub> ~A <sub>25</sub>	11.082 (76.410)	40.0 (275.80)

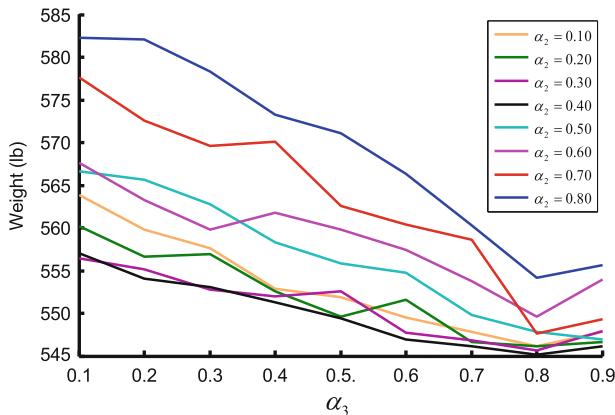
= 0.80 are suitable values for MBB–BC algorithm. These parameter values are used for all other examples presented.

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 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, and the standard deviation are compared. Also, the MBB–BC approach has smaller required number of iterations for convergence than 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 A 72-Bar Spatial Truss

For the 72-bar spatial truss structure shown in Fig. 9.4, the material density is 0.1 lb/in<sup>3</sup> (2767.990 kg/m<sup>3</sup>), and the modulus of elasticity is 10,000 ksi (68,950 MPa).

The members are subjected to the stress limits of  $\pm 25$  ksi ( $\pm 172.375$  MPa). The uppermost nodes are subjected to the displacement limits of  $\pm 0.25$  in (0.635) in



**Fig. 9.3** Effect of MBB–BC parameters on average weight of the 25-bar truss

both the x and y directions. The 72 structural members of this spatial truss are sorted into 16 groups using symmetry: (1) A<sub>1</sub>–A<sub>4</sub>, (2) A<sub>5</sub>–A<sub>12</sub>, (3) A<sub>13</sub>–A<sub>16</sub>, (4) A<sub>17</sub>–A<sub>18</sub>, (5) A<sub>19</sub>–A<sub>22</sub>, (6) A<sub>23</sub>–A<sub>30</sub>, (7) A<sub>31</sub>–A<sub>34</sub>, (8) A<sub>35</sub>–A<sub>36</sub>, (9) A<sub>37</sub>–A<sub>40</sub>, (10) A<sub>41</sub>–A<sub>48</sub>, (11) A<sub>49</sub>–A<sub>52</sub>, (12) A<sub>53</sub>–A<sub>54</sub>, (13) A<sub>55</sub>–A<sub>58</sub>, (14) A<sub>59</sub>–A<sub>66</sub> (15), A<sub>67</sub>–A<sub>70</sub>, and (16) A<sub>71</sub>–A<sub>72</sub>. The minimum permitted cross-sectional area of each member is 0.10 in<sup>2</sup> (0.6452 cm<sup>2</sup>), and the maximum cross-sectional area of each member is 4.00 in<sup>2</sup> (25.81 cm<sup>2</sup>). 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 as 1.912 lb [14]. In addition, the required analyses for reaching a convergence are 13,200 analyses, which are 48 % and 40 % less than the BB–BC method and ACO, respectively. Table 9.5 compares the performance of the improved BB–BC algorithm with those previously reported in the literature.

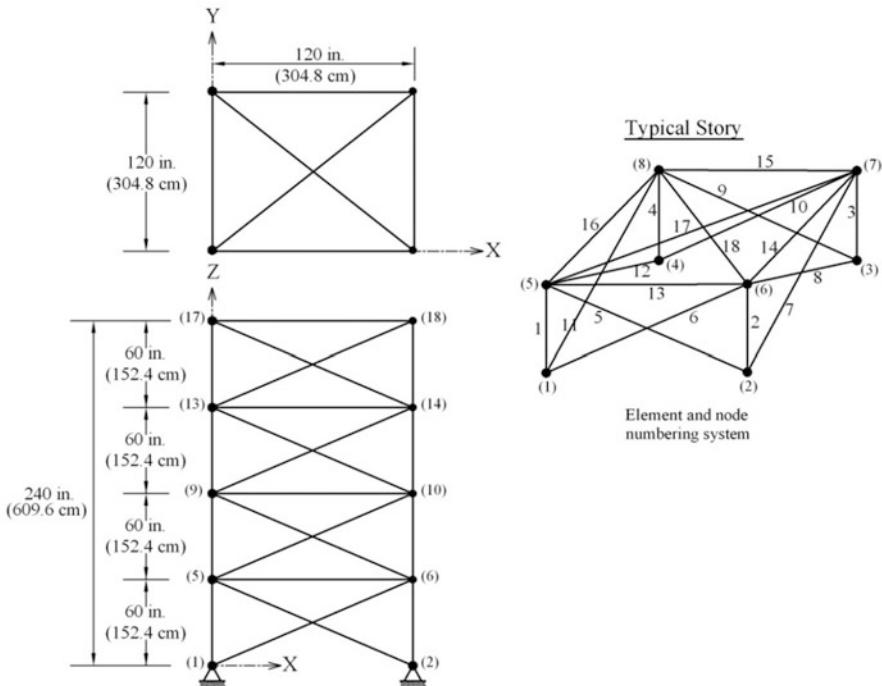
### 9.3.2.3 A 120-Bar Dome Truss

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<sup>3</sup> (7971.810 kg/m<sup>3</sup>). 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<sup>2</sup> (2 cm<sup>2</sup>), and the

**Table 9.3** Performance comparison for the 25-bar spatial truss

Optimal cross-sectional areas (in <sup>2</sup> )								
	Rajeev and Krishnamoorthy GA [8]	Schutte and Groenwold PSO [17]	Lee and Geem HSI [18]	Kaveh et al. IACS [4]	Kaveh and Talatahari PSACO [15]	HPSACO [16]	Camp BB-BC [14]	Present work [2] in <sup>2</sup> cm <sup>2</sup>
1 A <sub>1</sub>	0.10	0.010	0.047	0.010	0.010	0.010	0.010	0.010 0.065
2 A <sub>2</sub> ~A <sub>5</sub>	1.80	2.121	2.022	2.042	2.054	2.092	1.993	12.856
3 A <sub>6</sub> ~A <sub>9</sub>	2.30	2.893	2.950	3.001	3.008	2.964	3.056	19.717
4 A <sub>10</sub> ~A <sub>11</sub>	0.20	0.010	0.010	0.010	0.010	0.010	0.010	0.010 0.065
5 A <sub>12</sub> ~A <sub>13</sub>	0.10	0.010	0.014	0.010	0.010	0.010	0.010	0.010 0.065
6 A <sub>14</sub> ~A <sub>17</sub>	0.80	0.671	0.688	0.684	0.684	0.679	0.689	0.6665 4.293
7 A <sub>18</sub> ~A <sub>21</sub>	1.80	1.611	1.657	1.625	1.616	1.611	1.642	10.594
8 A <sub>22</sub> ~A <sub>25</sub>	3.0	2.717	2.663	2.672	2.673	2.678	2.686	2.679 17.281
Best weight (lb)	546	545.21	544.38	545.03	545.04	544.99	545.38	545.16 2425 N
Average weight (lb)	N/A	546.84	N/A	545.74	N/A	545.52	545.78	545.66
Std dev (lb)	N/A	1.478	N/A	0.620	N/A	0.315	0.491	0.367
No. of analyses	N/A	9596	15,000	3520	28,850	9875	20,566	12,500



**Fig. 9.4** Schematic of a 72-bar spatial truss

**Table 9.4** Loading conditions for the 72-bar spatial truss

Node	Case 1			Case 2		
	P <sub>X</sub> kips (kN)	P <sub>Y</sub> kips (kN)	P <sub>Z</sub> kips (kN)	P <sub>X</sub>	P <sub>Y</sub>	P <sub>Z</sub> 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)

maximum cross-sectional area is taken as 20.0 in<sup>2</sup> (129.03 cm<sup>2</sup>). The constraints are considered as:

- 1) Stress constraints (according to the ASD-AISC (1989) [22] code):

$$\begin{cases} \sigma_i^+ = 0.6F_y & \text{for } \sigma_i \geq 0 \\ \sigma_i^- & \text{for } \sigma_i < 0 \end{cases} \quad (9.15)$$

where  $\sigma_i^-$  is calculated according to the slenderness ratio:

**Table 9.5** Performance comparison for the 72-bar spatial truss

		Optimal cross-sectional areas ( $\text{in}^2$ )				
		Erbatur et al.	Camp and Bichon	Perez and Behdinan	Camp	Kaveh and Talatahari [2]
Element group	GA [21]	ACO [19]	PSO [20]	BB–BC [14]	$\text{in}^2$	$\text{cm}^2$
1	A <sub>1</sub> ~ A <sub>4</sub>	1.755	1.948	1.7427	1.8577	1.9042 12.2851
2	A <sub>5</sub> ~ A <sub>12</sub>	0.505	0.508	0.5185	0.5059	0.5162 3.3303
3	A <sub>13</sub> ~ A <sub>16</sub>	0.105	0.101	0.1000	0.1000	0.1000 0.6452
4	A <sub>17</sub> ~ A <sub>18</sub>	0.155	0.102	0.1000	0.1000	0.1000 0.6452
5	A <sub>19</sub> ~ A <sub>22</sub>	1.155	1.303	1.3079	1.2476	1.2582 8.1176
6	A <sub>23</sub> ~ A <sub>30</sub>	0.585	0.511	0.5193	0.5269	0.5035 3.2488
7	A <sub>31</sub> ~ A <sub>34</sub>	0.100	0.101	0.1000	0.1000	0.1000 0.6452
8	A <sub>35</sub> ~ A <sub>36</sub>	0.100	0.100	0.1000	0.1012	0.1000 0.6452
9	A <sub>37</sub> ~ A <sub>40</sub>	0.460	0.561	0.5142	0.5209	0.5178 3.3409
10	A <sub>41</sub> ~ A <sub>48</sub>	0.530	0.492	0.5464	0.5172	0.5214 3.3639
11	A <sub>49</sub> ~ A <sub>52</sub>	0.120	0.100	0.1000	0.1004	0.1000 0.6452
12	A <sub>53</sub> ~ A <sub>54</sub>	0.165	0.107	0.1095	0.1005	0.1007 0.6497
13	A <sub>55</sub> ~ A <sub>58</sub>	0.155	0.156	0.1615	0.1565	0.1566 1.0104
14	A <sub>59</sub> ~ A <sub>66</sub>	0.535	0.550	0.5092	0.5507	0.5421 3.4973
15	A <sub>67</sub> ~ A <sub>70</sub>	0.480	0.390	0.4967	0.3922	0.4132 2.6658
16	A <sub>71</sub> ~ A <sub>72</sub>	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 1689 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
No. of analyses		N/A	18,500	N/A	19,621	13,200

$$\sigma_i^- = \begin{cases} \left[ \left( 1 - \frac{\lambda_i^2}{2C_c^2} \right) F_y \right] \Bigg/ \left( \frac{5}{3} + \frac{3\lambda_i}{8C_c} - \frac{\lambda_i^3}{8C_c^3} \right) & \text{for } \lambda_i < C_c \\ \frac{12\pi^2 E}{23\lambda_i^2} & \text{for } \lambda_i \geq C_c \end{cases} \quad (9.16)$$

where  $E$  is the modulus of elasticity;  $F_y$  is the yield stress of steel;  $C_c$  is the slenderness ratio ( $\lambda_i$ ) dividing the elastic and inelastic buckling regions ( $C_c = \sqrt{2\pi^2 E/F_y}$ );  $\lambda_i$  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  $\pm 0.1969$  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

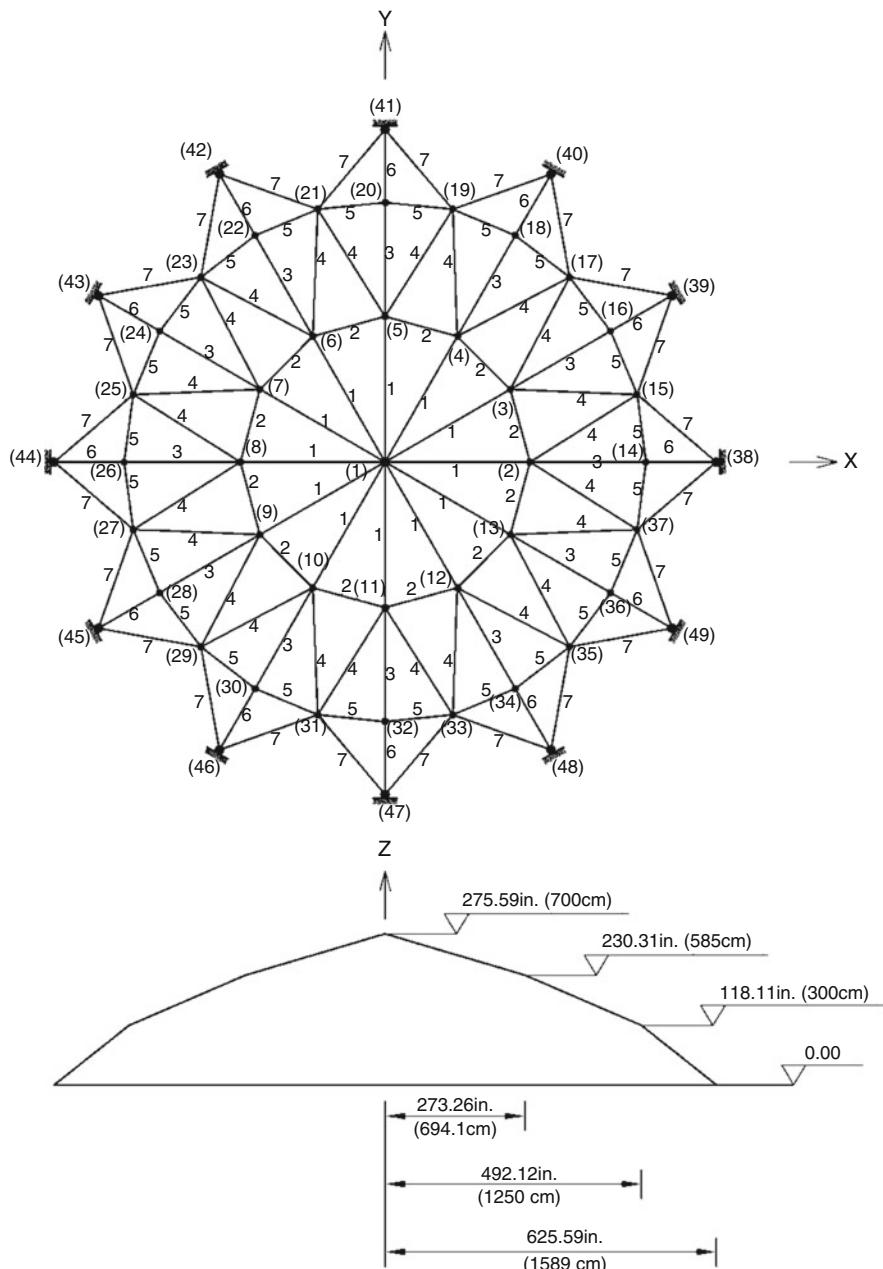


Fig. 9.5 Schematic of a 120-bar dome-shaped truss

**Table 9.6** Performance comparison for the 120-bar dome truss

		Optimal cross-sectional areas ( $\text{in}^2$ )					
		Kaveh et al.	Kaveh and Talatahari				Present work [2]
Element group	IACS [4]	PSOPC [15]	PSACO [15]	HPSACO [16]	BB–BC	$\text{in}^2$	$\text{cm}^2$
1 A <sub>1</sub>	3.026	3.040	3.026	3.095	3.026	3.037	19.596
2 A <sub>2</sub>	15.06	13.149	15.222	14.405	14.276	14.431	93.010
3 A <sub>3</sub>	4.707	5.646	4.904	5.020	4.986	5.130	33.094
4 A <sub>4</sub>	3.100	3.143	3.123	3.352	3.175	3.134	20.217
5 A <sub>5</sub>	8.513	8.759	8.341	8.631	8.617	8.591	55.427
6 A <sub>6</sub>	3.694	3.758	3.418	3.432	3.558	3.377	21.785
7 A <sub>7</sub>	2.503	2.502	2.498	2.499	2.510	2.500	16.129
Best weight (lb)	33,320.52	33,481.2	33,263.9	33,248.9	33,340.7	33,287.9	148,064 N
No. of analyses	3250	150,000	32,600	10,000	22,000	10,000	

methods. Except IACS which uses two auxiliary mechanisms for searching, the MBB–BC optimization and HPSACO have the best convergence rates.

### 9.3.2.4 A Square on Diagonal Double-Layer Grid

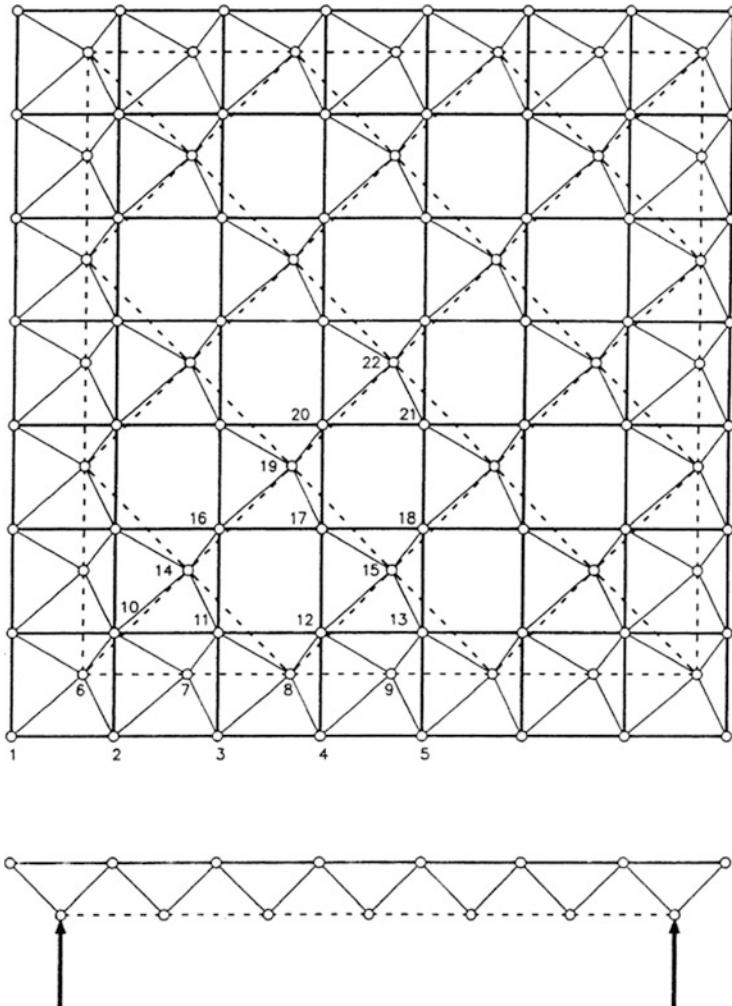
A double-layer grid of the type shown in Fig. 9.6 with a span of 21 m and the height of 1.5 m is chosen from [23]. The structure is simply supported at the corner nodes of the bottom layer.

The loading is assumed as a uniformly distributed load on the top layer with intensity of 155.5  $\text{kg}/\text{m}^2$ , 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  $\text{kg}/\text{cm}^3$  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  $\text{cm}^2$  with a lower limit of 0.1  $\text{cm}^2$ .

Stress, Euler buckling, and displacement constraints are considered in this problem. All the elements are subjected to the following stress constraints:

$$-1000 \leq \sigma_i \leq 1400 \text{ kg/cm}^2 \quad i = 1, 2, \dots, 47 \quad (9.17)$$

where  $i$  is the element number. Tubular members are considered with a diameter-to-thickness ratio of 10. Thus, Euler buckling is considered as:



**Fig. 9.6** Schematic of a square on diagonal double-layer grid [23]

$$\sigma_i^b = -10.1EA_i/8L_i^2 \quad i = 1, 2, \dots, 47 \quad (9.18)$$

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):

$$-1.5 \leq \delta_i \leq 1.5 \text{ cm} \quad i = 1, 2, 3 \quad (9.19)$$

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 Eq. (9.13) (250 iterations in average), while it is considered as 500 iterations for other examples. The results are presented in Table 9.7.

**Table 9.7** Performance comparison for the square on diagonal double-layer grid

Group	Members	Optimal cross-sectional areas ( $\text{cm}^2$ )				
		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

(continued)

**Table 9.7** (continued)

Group	Members	Optimal cross-sectional areas ( $\text{cm}^2$ )				
		GA	PSO	PSOPC	BB–BC	Present work [2]
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
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 (sec.)	1854	1420	1420	1249	631	

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 1249 s on a Core™ 2 Duo 3.0 GHz CPU. Also, the MBB–BC algorithm can find the best result in comparison to other algorithms. Figure 9.7 shows the convergence curves for the best and average of 50 runs for the proposed algorithm.

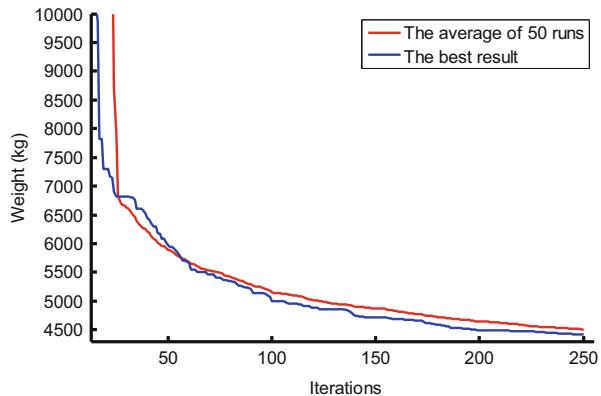
### 9.3.2.5 A 26-Story-Tower Spatial Truss

The 26-story-tower space truss containing 942 elements and 244 nodes 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<sup>3</sup> (2767.990 kg/m<sup>3</sup>), 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<sup>2</sup> (from 0.6452 to 129.032 cm<sup>2</sup>). 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).
- 3) The vertical load at each node in the third section is equal to  $-9$  kips ( $-40.032$  kN).

**Fig. 9.7** Convergence curves for the square on diagonal double-layer grid for the MBB–BC algorithm [2]



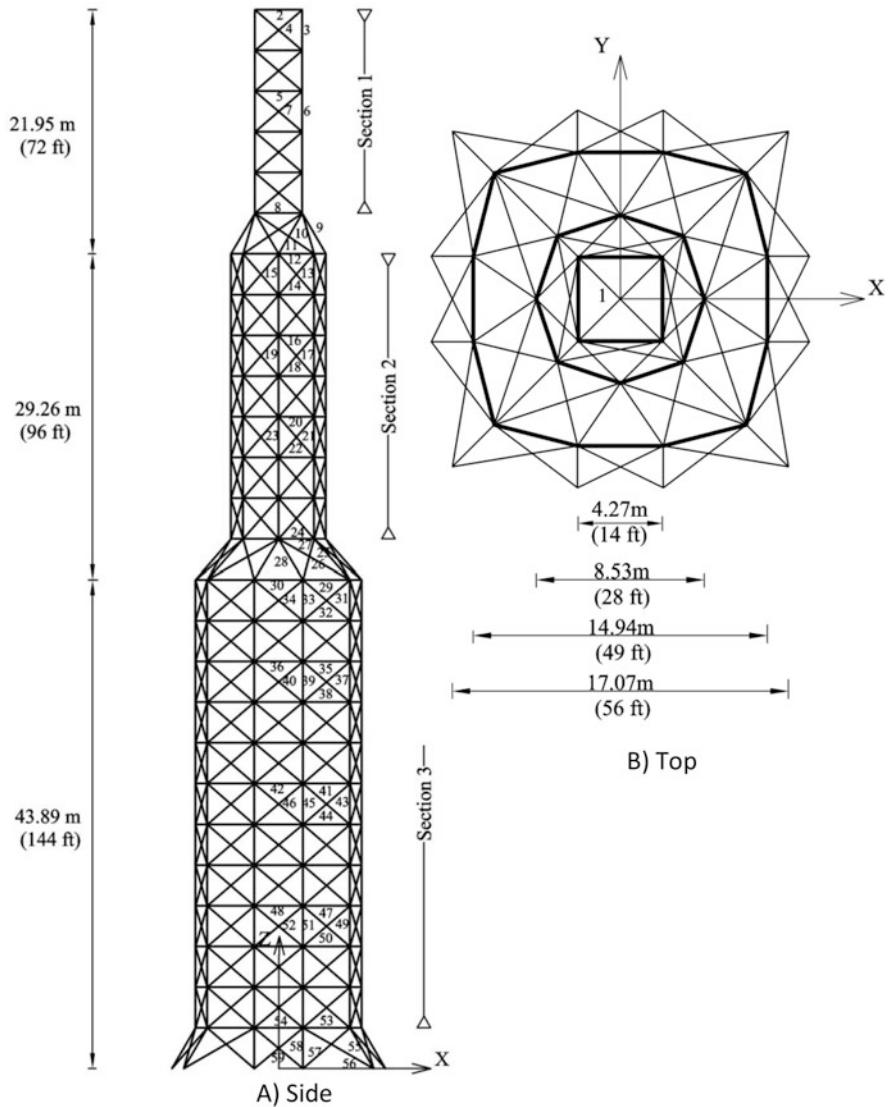
- 4) The horizontal load at each node on the right side in the x direction is equal to  $-1$  kips ( $-4.448$  kN).
- 5) The horizontal load at each node on the left side in the x direction is equal to  $1.5$  kips ( $6.672$  kN).
- 6) The horizontal load at each node on the front side in the y direction 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 has better performance in terms of the optimization time, the 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 of convergence curves for the proposed algorithm.

### 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



**Fig. 9.8** Schematic of a 26-story-truss tower

has low computational time and high convergence speed compared to BB-BC, specially when the number of design variables that 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

**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)	6640.6	9906.6	2621.3	1420.5
No. of analyses	50,000	50,000	50,000	30,000
Optimization time (sec.)	4450	3640	3162	1926

increasing the number of analyses. This problem is solved by using SOM which works as a search-space updating rule and reduces the number of analyses for convergence.

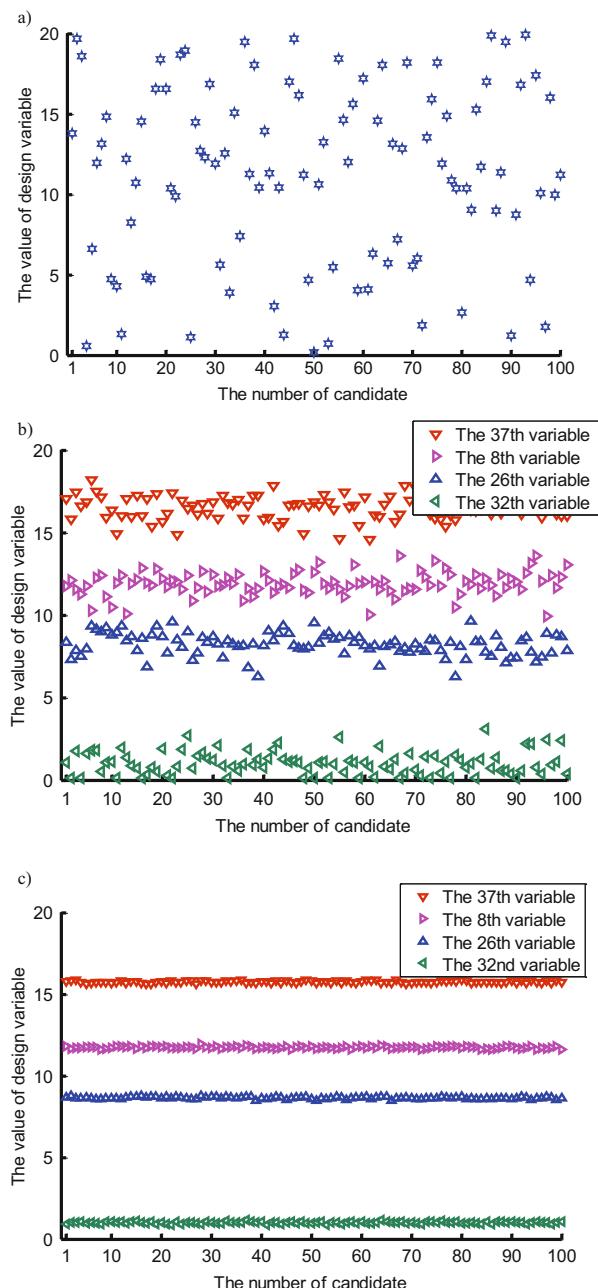
## 9.4 Optimal Design of Schwedler and Ribbed Domes Using MBB–BC Algorithm

### 9.4.1 Introduction

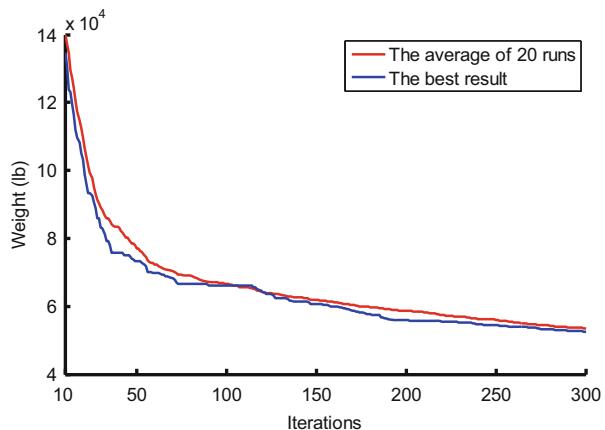
Covering large areas without intermediate supports has always been an attractive problem for architects and a challenging task for structural engineers. Dome structures are lightweight and elegant structures that provide economical solutions for covering large areas with their splendid aesthetic appearance. 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 domes does not only provide more realistic results but 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 the 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

**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 variables), (c) in the 300th iteration (for the 8th, 26th, 32nd, 37th variables) [2]



**Fig. 9.10** Convergence curves for the 26-story-tower truss for the MBB–BC algorithm [2]



pipe section 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 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:

$$\begin{aligned} \text{Find} \quad & \mathbf{X} = [x_1, x_2, \dots, x_{ng}], h, Nr \\ & x_i \in \{d_1, d_2, \dots, d_{37}\} \quad \text{to minimize} \quad V(\mathbf{X}) = \sum_{i=1}^{nm} x_i \cdot L_i \quad (9.20) \\ & h_i \in \{h_{\min}, h_{\min} + h^*, \dots, h_{\max}\} \end{aligned}$$

subjected to the following constraints:

##### Displacement Constraint

$$\delta_i \leq \delta_i^{\max} \quad i = 1, 2, \dots, nn \quad (9.21)$$

**Table 9.9** The allowable steel pipe sections taken from LRFD–AISC

	Type	Nominal diameter in	Weight per ft (lb)	Area in <sup>2</sup>	I in <sup>4</sup>	S in <sup>3</sup>	J in <sup>4</sup>	Z in <sup>3</sup>
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 <sup>1</sup> / <sub>4</sub>	2.27	0.669	0.195	0.235	0.470	0.324
8	ST	1 <sup>1</sup> / <sub>2</sub>	2.72	0.799	0.310	0.326	0.652	0.448
9	EST	1 <sup>1</sup> / <sub>4</sub>	3.00	0.881	0.242	0.291	0.582	0.414
10	EST	1 <sup>1</sup> / <sub>2</sub>	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 <sup>1</sup> / <sub>2</sub>	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 <sup>1</sup> / <sub>2</sub>	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 <sup>1</sup> / <sub>2</sub>	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 <sup>1</sup> / <sub>2</sub>	12.50	3.68	6.28	3.14	6.28	4.32
21	DEST	2 <sup>1</sup> / <sub>2</sub>	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

### Interaction Formula Constraints

$$\frac{P_u}{2\varphi_c P_n} + \left( \frac{M_{ux}}{\varphi_b M_{nx}} + \frac{M_{uy}}{\varphi_b M_{ny}} \right) \leq 1 \quad \text{For } \frac{P_u}{\varphi_c P_n} < 0.2 \quad (9.22)$$

$$\frac{P_u}{\varphi_c P_n} + \frac{8}{9} \left( \frac{M_{ux}}{\varphi_b M_{nx}} + \frac{M_{uy}}{\varphi_b M_{ny}} \right) \leq 1 \quad \text{For } \frac{P_u}{\varphi_c P_n} \geq 0.2 \quad (9.23)$$

### Shear Constraint

$$V_u \leq \varphi_v V_n \quad (9.24)$$

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  $j$ th 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  $i$ ;  $\delta_i$  is the displacement of node  $i$ ;  $\delta_i^{\max}$  is the permitted displacement for the  $i$ th node;  $nn$  is the total number of nodes;  $\varphi_c$  is the resistance factor ( $\varphi_c = 0.9$  for tension,  $\varphi_c = 0.85$  for compression);  $\varphi_b$  is the flexural resistance reduction factor ( $\varphi_b = 0.90$ );  $M_{ux}$  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:

$$P_n = A_g F_{cr} \quad (9.25)$$

where  $A_g$  is the gross area of a member and  $F_{cr}$  is calculated as follows:

$$F_{cr} = \left( 0.658^{\lambda_c^2} \right) \cdot f_y \quad \text{For } \lambda_c \leq 1.5 \quad (9.26)$$

$$F_{cr} = \left( \frac{0.877}{\lambda_c^2} \right) \cdot f_y \quad \text{For } \lambda_c > 1.5 \quad (9.27)$$

Here,  $f_y$  is the specified yield stress and  $\lambda_c$  is obtained from:

$$\lambda_c = \frac{kl}{\pi r} \sqrt{\frac{f_y}{E}} \quad (9.28)$$

where  $k$  is the effective length factor taken as 1;  $l$  is the length of a dome member;  $r$  is the governing radius of gyration about the axis of buckling; and  $E$  is the modulus of elasticity.

In Eq. (9.24),  $V_u$  is the factored service load shear;  $V_n$  is the nominal strength in shear; and  $\varphi_v$  represents the resistance factor for shear ( $\varphi_v = 0.90$ ).

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:

$$Mer^k = \varepsilon_1 \cdot V^k + \varepsilon_2 \cdot (\Phi^k)^{\varepsilon_3} \quad (9.29)$$

where  $Mer^k$  = merit function for the  $k$ th candidate;  $\varepsilon_1$ ,  $\varepsilon_2$ , and  $\varepsilon_3$  = coefficients of merit function; and  $\Phi^k$  = 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,  $\varepsilon_1$  is set to 1. The coefficient  $\varepsilon_2$  is taken as the volume of the structure, and the coefficient  $\varepsilon_3$  is set to 1.5, but gradually, it is increased to 3 (Ref. [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:

$$A_i^{c(k)} = \frac{\sum_{j=1}^N \frac{1}{Mer^j} \cdot A_i^{(k,j)}}{\sum_{j=1}^N \frac{1}{Mer^j}} \quad i = 1, 2, \dots, ng \quad (9.30)$$

where  $A_i^{(k,j)}$  is the  $i$ th component of the  $j$ th solution generated in the  $k$ th iteration and  $N$  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:

$$A_i^{(k+1,j)} = \alpha_2 A_i^{c(k)} + (1 - \alpha_2) \left( \alpha_3 A_i^{gbest(k)} + (1 - \alpha_3) A_i^{lbest(k,j)} \right) + \frac{r_j \alpha_1 (A_{\max} - A_{\min})}{k + 1} \quad \begin{cases} i = 1, 2, \dots, n_g \\ j = 1, 2, \dots, N \end{cases} \quad (9.31)$$

where  $r_j$  is a random number from a standard normal distribution which changes for each candidate;  $\alpha_1$  is a parameter for limiting the size of the search space;  $A_{\min} = 0.250$  in.<sup>2</sup>;  $A_{\max} = 21.3$  in.<sup>2</sup>;  $A_i^{lbest(k,j)}$  is the best position of the  $j$ th particle up to the iteration  $k$  and  $A_i^{gbest(k)}$  is the best position among all candidates up to the iteration  $k$ ; and  $\alpha_2$  and  $\alpha_3$  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 in the following:

$$A_i^{(k+1,j)} = Fix \left( \alpha_2 A_i^{c(k)} + (1 - \alpha_2) \left( \alpha_3 A_i^{gbest(k)} + (1 - \alpha_3) A_i^{lbest(k,j)} \right) + \frac{r_j \alpha_1 (A_{\max} - A_{\min})}{k + 1} \right) \quad (9.32)$$

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 behavior 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 in the following [24]:

**Step 1:** Set the load factor to a preselected 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 Ref. [30].

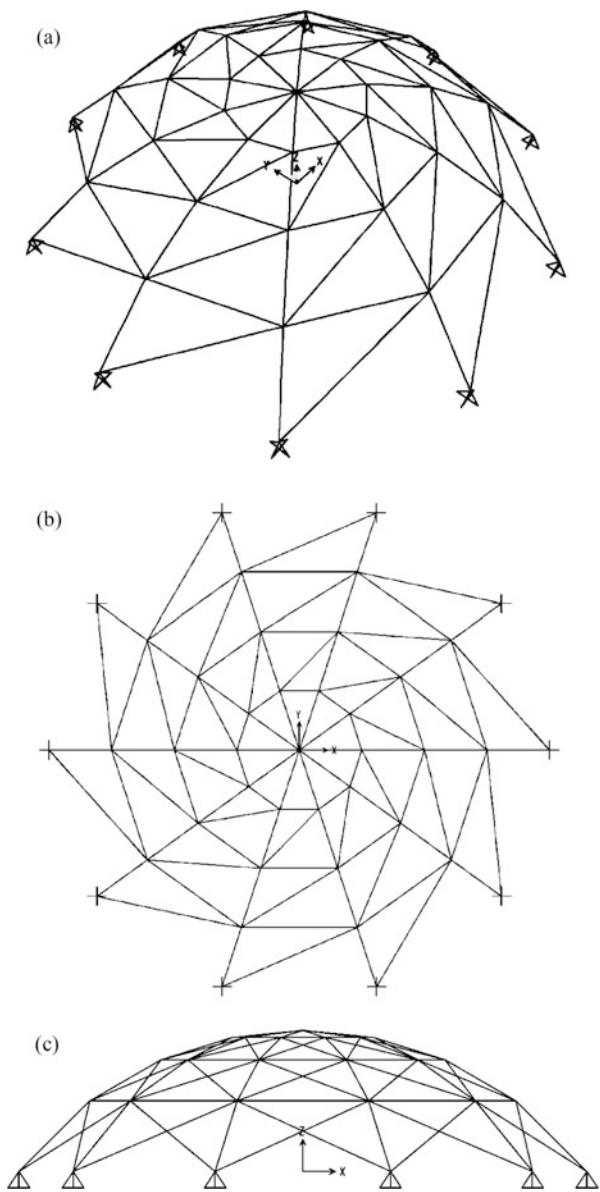
- Step 3:** Set up the nonlinear stiffness matrix for each member.
- Step 4:** Transform the member stiffness matrices from local coordinates into the global coordinate, and assemble the overall stiffness matrix.
- Step 5:** Check the stability of the dome. Calculate the determinant of the overall stiffness 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 Step 2 until differences between two successive sets of axial forces are smaller than a specific tolerance.
- Step 10:** Increase the load factor by preselected 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

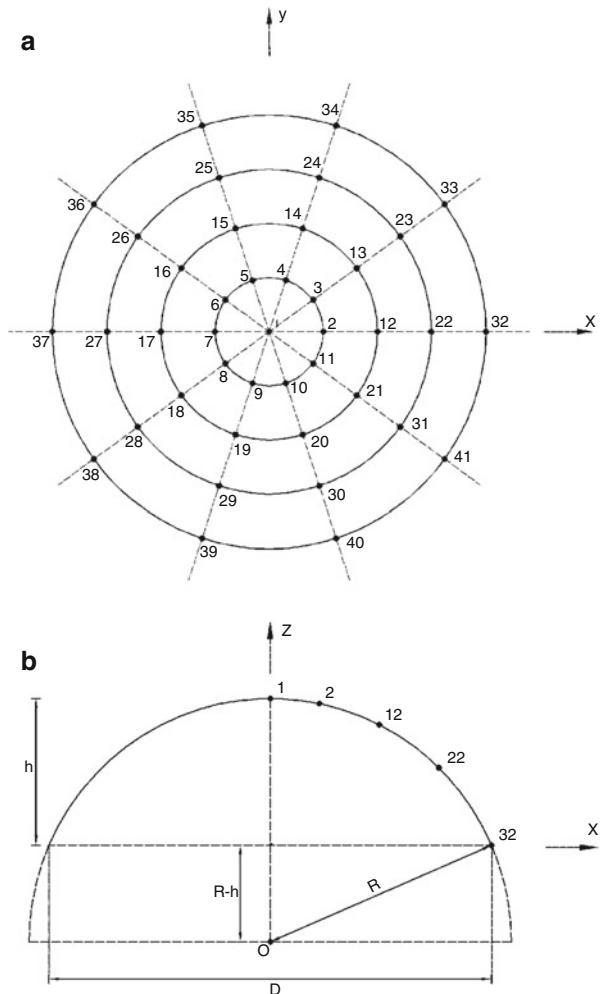
The configuration of a Schwedler dome is shown in Fig. 9.11. Schwedler, a German engineer, who introduced this type of dome in 1863, built numerous braced domes during his lifetime. A Schwedler dome, one of the most popular types of braced domes, consists of meridional *ribs* connected together to a 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 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 as 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 center of the coordinate system in  $x$ - $y$  plane. The coordinates of other joints in each ring are obtained as:

**Fig. 9.11** Schematic of a Schwedler dome [3]. (a) Three-dimensional view, (b) top view, (c) side view



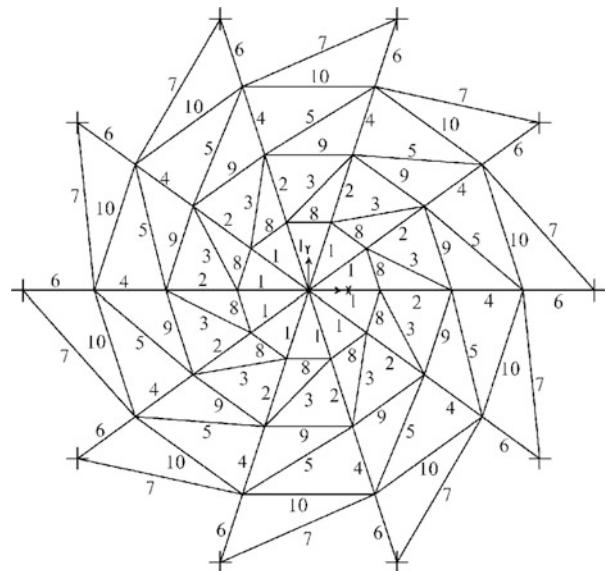
**Fig. 9.12** Nodal numbering and the corresponding coordinate system [3]. (a) Top view of the dome, (b) section of the dome



$$\left\{ \begin{array}{l} x_i = \frac{D}{2N} \cos \left( \frac{360}{4n_i} \left( i - \sum_{j=1}^{i-1} 4n_j - 1 \right) \right) \\ y_i = \frac{D}{2N} \sin \left( \frac{360}{4n_i} \left( i - \sum_{j=1}^{i-1} 4n_j - 1 \right) \right) \\ z_i = \sqrt{\left( R^2 - \frac{n_i^2 D^2}{4N^2} \right)} - (R - h) \end{array} \right. \quad (9.33)$$

where  $n_i$  is the number of the ring corresponding to the node  $i$  and  $R = (D^2 + 4h^2)/(8h)$  is the radius of the hemisphere as shown in Fig. 9.12b.

**Fig. 9.13** The Schwedler dome with the related member grouping [3]



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 the following equations:

$$\begin{cases} I = 10 \times (n_i - 1) + j + 1 \\ J = 10 \times (n_i - 1) + j + 2 \end{cases} \quad \begin{cases} j = 1, 2, 3, \dots, 9 \\ n_i = 1, 2, \dots, Nr - 1 \end{cases} \quad (9.34)$$

$$\begin{cases} I = 10 \times (n_i - 1) + 2 \\ J = 10 \times n_i + 1 \end{cases} \quad n_i = 1, 2, \dots, Nr - 1 \quad (9.35)$$

Also for the rib and diagonal groups, we have

$$\begin{cases} I = 10 \times (n_i - 1) + 2 + \text{Fix}\left(\frac{j-1}{2}\right) \\ J = 10 \times n_i + j - \text{Fix}\left(\frac{j-1}{2}\right) \end{cases} \quad \begin{cases} j = 2, 3, \dots, 20 \\ n_i = 1, 2, \dots, Nr - 1 \end{cases} \quad (9.36)$$

$$\begin{cases} I = 10 \times (n_i - 1) + 2 \\ J = 10 \times (n_i + 1) + 1 \end{cases} \quad n_i = 1, 2, \dots, Nr - 1 \quad (9.37)$$

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 Eq. (9.35). Equations (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:

$$\begin{cases} I = 10 \times (n_i - 1) + j + 1 \\ J = 10 \times n_i + j + 1 \end{cases} \quad \begin{cases} j = 1, 2, \dots, 10 \\ n_i = 1, 2, \dots, Nr - 1 \end{cases} \quad (9.38)$$

#### 9.4.6 Results and Discussion

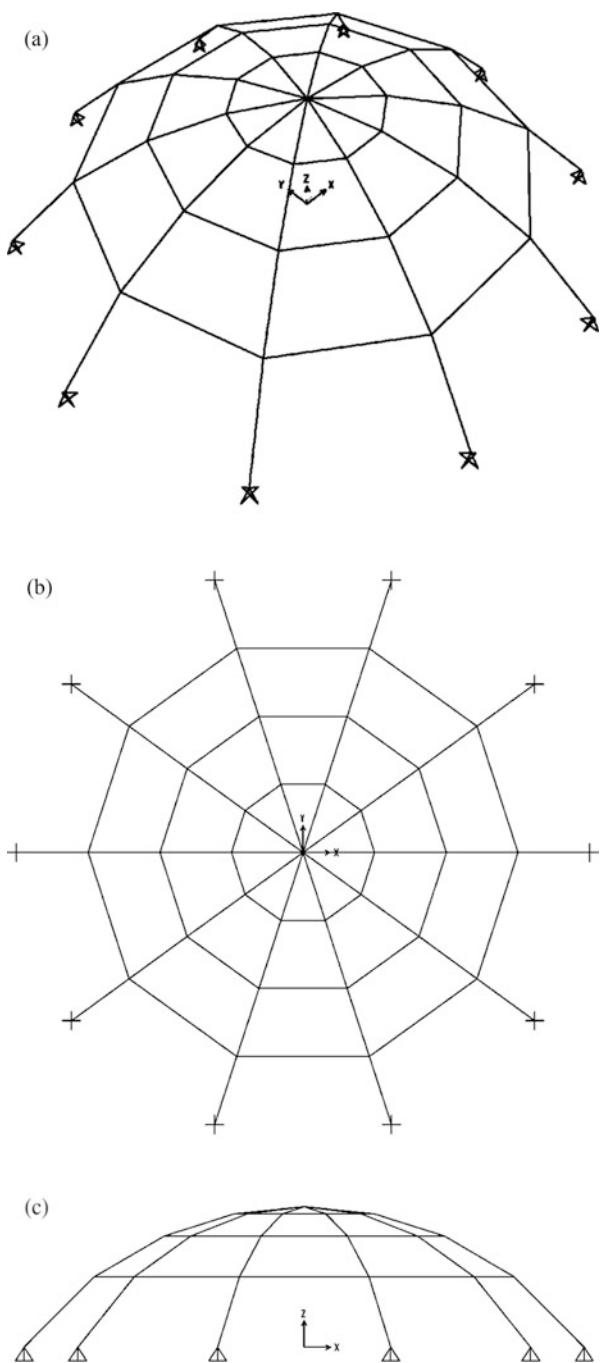
This section presents the optimum design of the Schwedler and ribbed domes using the MBB–BC algorithm. The modulus of elasticity for the steel is taken as 205 kN/mm<sup>2</sup>. The limitations imposed on the joint displacements are 28 mm in the  $z$  direction and 33 mm in the  $x$  and  $y$  directions for the first, second, and third 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.

##### 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:

**Fig. 9.14** Schematic of a ribbed dome [3]. (a) Three-dimensional view, (b) top view, (c) side view



**Table 9.10** Optimum design of the ribbed and Schwedler domes

Group number	Optimum section (designations)					
	Case 1		Case 2		Case 3	
	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 ( $\frac{1}{2}$ )	PIPST (6)	PIPST (3)	PIPST (12)	PIPST ( $\frac{3}{2}$ )
3	PIPST (5)	PIPST (5)	PIPST (10)	PIPST ( $\frac{1}{2}$ )	PIPST (10)	PIPST (6)
4	PIPST (8)	PIPST ( $\frac{1}{2}$ )	PIPST ( $\frac{1}{2}$ )	PIPST (3)	PIPST (8)	PIPST (4)
5	PIPST (5)	PIPST (5)	PIPST ( $\frac{1}{4}$ )	PIPST ( $\frac{1}{2}$ )	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 <sup>3</sup> )	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
$\bar{A}$ (cm <sup>2</sup> )	35.35	22.06	35.15	13.81	71.20	32.83

**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 ( $\bar{A}$ ) and the sum of the element lengths, written as

$$V(\mathbf{X}) = \bar{A} \cdot \sum_{i=1}^{nm} L_i \quad (9.39)$$

In Case 1,  $\bar{A}$  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 that support 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.

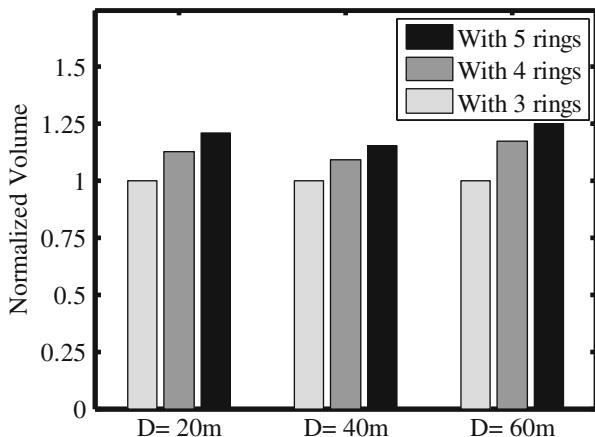
The Schwedler dome contains more appropriate sections and lighter weight than the ribbed dome for Case 3. In order to provide lateral stiffness, all rib members in the ribbed domes have very strong sections, and  $\bar{A}$  has a very large value, whereas the Schwedler dome has small  $\bar{A}$  because 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 on one hand, and on the other hand, it must have a large value to provide the 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 as 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

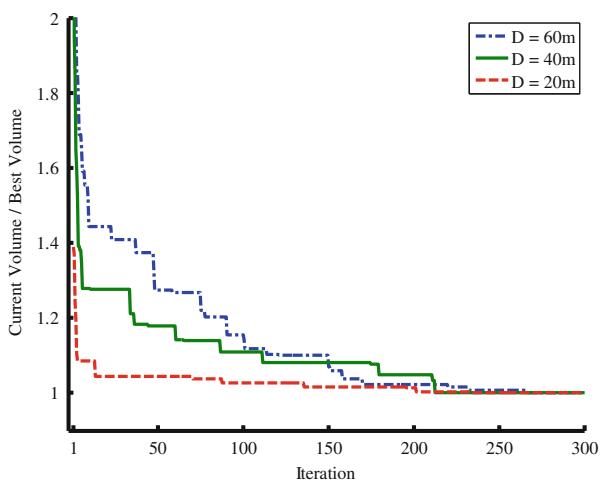
In order to investigate the efficiency of the Schwedler domes, two other domes with different diameters are considered: one with smaller diameter (20 m) and another with larger diameter (60 m). The loading condition is the same as Case 3 in the previous section. Figure 9.15 shows the normalized optimum 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 results unless a dome has a very large diameter in which case some elements will buckle if the number of rings is selected small.

The optimum height of the crown is 5.25, 10.75, and 18.5 m for domes with 20, 40, and 60 m diameters, and the ratio of the height to the diameter is equal to

**Fig. 9.15** The normalized optimum volume for the Schwedler domes with different number of rings [3]



**Fig. 9.16** The convergence curves for the Schwedler domes [3]



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 0.2–0.4 can be utilized as a good search space for the ratio of the height to the diameter.

Convergence curves 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.

**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 ( $\text{m}^3$ )	0.53	1.94	4.11
$\sum l_i$ (m)	294.25	591.10	913.27
$\bar{A}$ ( $\text{cm}^2$ )	18.16	32.83	45.08
Required materials to cover the space	0.170	0.154	0.145

## 9.5 Concluding Remarks

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. 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 and 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 are the best choice, and selecting a ratio of the height to the diameter from the range of [0.2,04] 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 Schewdler and ribbed domes; hybrid Big Bang–Big Crunch algorithm. *J Constr 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, UK
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 Publishers, Boston
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, Kiic 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. 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., UK

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] and 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 other 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) 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].

## 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:

$$\begin{aligned} \text{Find } \quad & \{X\} = [x_1, x_2, \dots, x_{ng}], \quad x_i \in D_i \\ \text{To minimize } \quad & W(\{X\}) = \sum_{i=1}^{ng} x_i \sum_{j=1}^{nm(i)} \rho_j \cdot L_j \\ \text{Subject to : } \quad & g_j(\{X\}) \leq 0 \quad j = 1, 2, \dots, n \end{aligned} \quad (10.1)$$

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  $x_i$ ;  $W(\{X\})$  presents weight of the structure;  $nm(i)$  is the number of members for the  $i$ th design variable;  $\rho_j$  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.

$$D_i = \{x_i | x_i \in [x_{i, \min}, x_{i, \max}]\} \quad (10.2)$$

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:

$$D_i = (d_{i, 1}, d_{i, 2}, \dots, d_{i, nm(i)}) \quad (10.3)$$

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:

$$f_{\text{cost}}(\{X\}) = (1 + \varepsilon_1 \cdot v)^{\varepsilon_2} \times W(\{X\}), \quad v = \sum_{j=1}^n \max[0, g_j(\{X\})] \quad (10.4)$$

where  $n$  represents the number of evaluated constraints for each individual design and  $v$  denotes the sum of the violations of the design. The constants  $\varepsilon_1$  and  $\varepsilon_2$  are selected considering the exploration and the exploitation rate of the search space. Here,  $\varepsilon_1$  is set to unity, and  $\varepsilon_2$  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,  $\varepsilon_2$  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:

$$\begin{cases} \sigma_i^+ = 0.6 F_y & \text{for } \sigma_i \geq 0 \\ \sigma_i^- & \text{for } \sigma_i < 0 \end{cases} \quad (10.5)$$

$$\sigma_i^- = \begin{cases} \left[ \left( 1 - \frac{\lambda_i^2}{2 C_c^2} \right) F_y \right] / \left( \frac{5}{3} + \frac{3\lambda_i}{8 C_c} + \frac{\lambda_i^3}{8 C_c^3} \right) & \text{for } \lambda_i \geq C_c \\ \frac{12 \pi^2 E}{23 \lambda_i^2} & \text{for } \lambda_i \leq C_c \end{cases} \quad (10.6)$$

where  $E$  is the modulus of elasticity;  $F_y$  is the yield stress of steel;  $c_c$  denotes the slenderness ratio ( $\lambda_i$ ) dividing the elastic and inelastic buckling regions ( $C_c = \sqrt{2\pi^2 E/F_y}$ );  $\lambda_i$  = the slenderness ratio ( $\lambda_i = k l_i / r_i$ );  $k$  = 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:

$$\delta_i - \delta_i^u \leq 0 \quad i = 1, 2, \dots, nn \quad (10.7)$$

where  $\delta_i$  is the nodal deflection;  $\delta_i^u$  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:

$$S_n = \sum_{i=1}^n X_i = X_1 + X_2 + \dots + X_n = \sum_{i=1}^{n-1} X_i + X_n = S_{n-1} + X_n \quad (10.8)$$

where  $S_n$  presents the random walk with  $n$  random steps and  $X_i$  is the  $i$ th 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's algorithm. In the Mantegna's algorithm, the step length  $S$  can be calculated by:

$$S = \frac{u}{|v|^{1/\beta}} \quad (10.9)$$

where  $\beta$  is a parameter between [1, 2] interval and considered to be 1.5;  $u$  and  $v$  are drawn from normal distribution as:

$$u \sim N(0, \sigma_u^2), \quad v \sim N(0, \sigma_v^2) \quad (10.10)$$

where

$$\sigma_u = \left\{ \frac{\Gamma(1 + \beta) \sin(\pi\beta/2)}{\Gamma[(1 + \beta)/2] \beta 2^{(\beta-1)/2}} \right\}^{1/\beta}, \quad \sigma_v = 1 \quad (10.11)$$

Studies show that the Lévy flights 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 the host birds are able to combat this parasitic 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

```

Objective function  $f(x)$ ,  $x = (x_1, x_2, \dots, x_d)$ ;  

Generate initial population of  $n$  host nests  

 $x_i$  ( $i = 1, 2, \dots, n$ );  

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  $F_i \geq F_j$   

        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]

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 latter assumption can be approximated by the fraction  $pa$  of the  $n$  nests which 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:

$$x_i^{(t+1)} = x_i^{(t)} + \alpha \cdot S \quad (10.12)$$

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 the Eq. (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 probability of a solution's component to be discovered. Therefore, a probability matrix is produced as:

$$P_{ij} = \begin{cases} 1 & \text{if } rand < pa \\ 0 & \text{if } rand \geq pa \end{cases} \quad (10.13)$$

where  $rand$  is a random number in  $[0, 1]$  interval and  $P_{ij}$  is discovering probability for the  $j$ th variable of the  $i$ 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:

##### 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 ( $\alpha$ ), discovering probability ( $pa$ ), and maximum number of analyses as the stopping criterion.

**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:

$$\text{nest}_{i,j}^{(0)} = x_{j,\min} + \text{rand.} (x_{j,\max} - x_{j,\min}) \quad (10.14\text{a})$$

where  $\text{nest}_{i,j}^{(0)}$  determines the initial value of the  $j$ th variable for the  $i$ th nest;  $x_{j,\min}$  and  $x_{j,\max}$  are the minimum and the maximum allowable values for the  $j$ th variable; and  $\text{rand}$  is a random number in the interval [0, 1]. For problems with discrete design variables, it is necessary to use a rounding function as:

$$\text{nest}_{i,j}^{(0)} = \text{ROUND} (x_{j,\min} + \text{rand.} (x_{j,\max} - x_{j,\min})) \quad (10.14\text{b})$$

### 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:

$$\text{nest}_i^{(t+1)} = \text{nest}_i^{(t)} + \alpha \cdot S \cdot (\text{nest}_i^{(t)} - \text{nest}_{\text{best}}^{(t)}) \cdot r \quad (10.15)$$

where  $\text{nest}_i^t$  is the  $i$ th nest current position;  $\alpha$  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  $\text{nest}_{\text{best}}^t$  is the position of the best nest so far.

### Alien Eggs Discovery

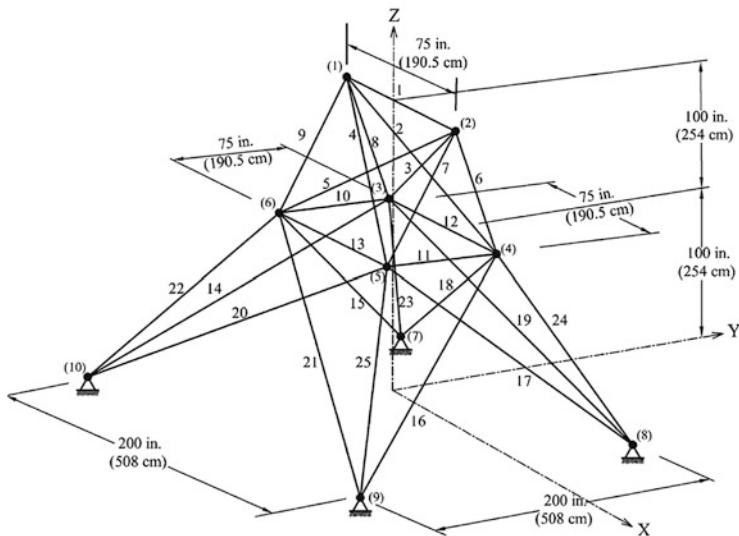
The alien eggs discovery is performed for all of the eggs using the probability matrix for each component of each solution. Existing eggs are replaced considering quality by newly generated ones from their current position by random walks with step size such as [7]:

$$\begin{aligned} S &= \text{rand.}(\text{nests}[\text{permute1}[i][j]] - \text{nests}[\text{permute2}[i][j]]) \\ \text{nest}^{(t+1)} &= \text{nest}^{(t)} + S.*P \end{aligned} \quad (10.16)$$

where  $\text{permute1}$  and  $\text{permute2}$  are different row permutation functions applied to the nests matrix and  $P$  is the probability matrix which was mentioned in Eq. (10.13).

### 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.



**Fig. 10.2** Schematic of a 25-bar space truss

### 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  $n = 7$  to  $20$  and  $pa = 0.15$  to  $0.35$  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  $0.1 \text{ lb/in}^3$  ( $2767.990 \text{ kg/m}^3$ ), and the modulus of elasticity is taken as  $10^7 \text{ psi}$  ( $68,950 \text{ MPa}$ ). 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.

### 10.2.5.2 Design of the 25-Bar Truss Utilizing Discrete Variables

In the first design of the 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,  $\pm 40$  ksi (275.80 MPa) for each member and  $\pm 0.35$  in ( $\pm 8.89$  mm) for each node in the x, y, and z directions. The range of discrete cross-sectional areas is from 0.1 to  $3.4 \text{ in}^2$  ( $0.6452\text{--}21.94 \text{ cm}^2$ ) with  $0.1 \text{ in}^2$  ( $0.6452 \text{ cm}^2$ ) 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 (2157.58 N), after 2000 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 the 25-Bar Truss Utilizing Continuous Variables

In the second design of the 25-bar truss, the structure is subjected to two load cases listed in Table 10.2. Maximum displacement limitations of  $\pm 0.35$  in ( $\pm 8.89$  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  $3.4 \text{ in}^2$  ( $0.06452\text{--}21.94 \text{ cm}^2$ ) [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 is obtained by IACS algorithm [11] in the aspects of low weight and number of analyses. The CS-based algorithm needs 6100 analyses to find the best solution while this number is equal to 9596, 15,000, 9875, 12,500, and 7000 analyses for a PSO-based algorithm; HS algorithm [12]; a 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 is very small, but the average weight obtained by the CS algorithm for 100 runs is better than others. The convergence curves for the best result and average weight of 100 runs are shown in Fig. 10.3. The important point is that although the CS requires 6100 analyses to achieve the 545.17 lb (2426.02 N), it can achieve the 545.76 lb (2428.63 N) after 2700 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 10.1** Performance comparison for the 25-bar spatial truss under single load case

Element group		Optimal cross-sectional areas ( $\text{in}^2$ )				
		GA [8]	GA [8]	ACO [9]	BB–BC phase 1, 2 [8]	Present work [4] $\text{in}^2$
						$\text{cm}^2$
1	A <sub>1</sub>	0.10	0.10	0.10	0.10	0.10
2	A <sub>2</sub> –A <sub>5</sub>	1.80	0.50	0.30	0.30	0.30
3	A <sub>6</sub> –A <sub>9</sub>	2.30	3.40	3.40	3.40	21.935
4	A <sub>10</sub> –A <sub>11</sub>	0.20	0.10	0.10	0.10	0.10
5	A <sub>12</sub> –A <sub>13</sub>	0.10	1.90	2.10	2.10	2.10
6	A <sub>14</sub> –A <sub>17</sub>	0.80	0.90	1.00	1.00	1.00
7	A <sub>18</sub> –A <sub>21</sub>	1.80	0.50	0.50	0.50	0.50
8	A <sub>22</sub> –A <sub>25</sub>	3.00	3.40	3.40	3.40	21.935
Best weight (lb)		546.01	485.05	484.85	484.85	484.85
Average weight (lb)		N/A	N/A	486.46	485.10	485.01
Number of analyses		800	15,000	7700	9000	2000

**Table 10.2** Loading conditions for the 25-bar spatial truss

Case	Node	F <sub>x</sub> kips (kN)	F <sub>y</sub> kips (kN)	F <sub>z</sub> 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

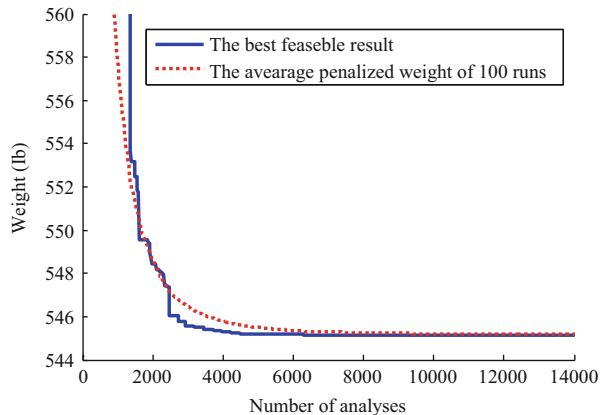
Element group		Compression ksi (MPa)	Tension ksi (MPa)
1	A <sub>1</sub>	35.092 (241.96)	40.0 (275.80)
2	A <sub>2</sub> –A <sub>5</sub>	11.590 (79.913)	40.0 (275.80)
3	A <sub>6</sub> –A <sub>9</sub>	17.305 (119.31)	40.0 (275.80)
4	A <sub>10</sub> –A <sub>11</sub>	35.092 (241.96)	40.0 (275.80)
5	A <sub>12</sub> –A <sub>13</sub>	35.092 (241.96)	40.0 (275.80)
6	A <sub>14</sub> –A <sub>17</sub>	6.759 (46.603)	40.0 (275.80)
7	A <sub>18</sub> –A <sub>21</sub>	6.959 (47.982)	40.0 (275.80)
8	A <sub>22</sub> –A <sub>25</sub>	11.082 (76.410)	40.0 (275.80)

#### 10.2.5.4 A 72-Bar Space Truss

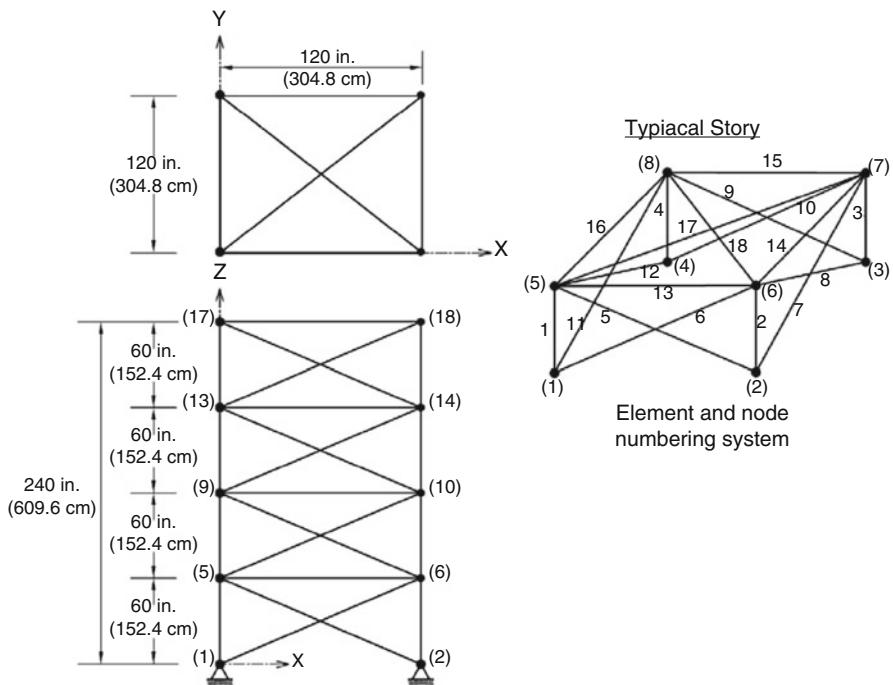
For the 72-bar spatial truss structure shown in Fig. 10.4 taken from [14], the material density is 0.1 lb/in<sup>3</sup> (2767.990 kg/m<sup>3</sup>), and the modulus of elasticity is 10<sup>7</sup> psi (68,950 MPa). The 72 structural members of this spatial truss are categorized into 16 groups using symmetry: (1) A<sub>1</sub>–A<sub>4</sub>, (2) A<sub>5</sub>–A<sub>12</sub>, (3) A<sub>13</sub>–A<sub>16</sub>, (4) A<sub>17</sub>–A<sub>18</sub>,

**Table 10.4** Performance comparison for the 25-bar spatial truss under multiple load cases

Element group		Optimal cross-sectional areas ( $\text{in}^2$ )						
		PSO [10]	HS [12]	IACS [11]	HPSACO [13]	HBB-BC [14]	CSS [10]	Present work [4]
							$\text{in}^2$	$\text{cm}^2$
1	A <sub>1</sub>	0.010	0.047	0.010	0.010	0.010	0.010	0.01 0.065
2	A <sub>2</sub> -A <sub>5</sub>	2.121	2.022	2.042	2.054	1.993	2.003	1.979 12.765
3	A <sub>6</sub> -A <sub>9</sub>	2.893	2.950	3.001	3.008	3.056	3.007	3.005 19.386
4	A <sub>10</sub> -A <sub>11</sub>	0.010	0.010	0.010	0.010	0.010	0.010	0.01 0.065
5	A <sub>12</sub> -A <sub>13</sub>	0.010	0.014	0.010	0.010	0.010	0.010	0.01 0.065
6	A <sub>14</sub> -A <sub>17</sub>	0.671	0.688	0.684	0.679	0.665	0.687	0.686 4.428
7	A <sub>18</sub> -A <sub>21</sub>	1.611	1.657	1.625	1.611	1.642	1.655	1.679 10.830
6	A <sub>22</sub> -A <sub>25</sub>	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 2426.02 (N)
Average weight (lb)		546.84	N/A	545.74	545.52	545.66	545.58	545.18 2426.05 (N)
Number of analyses		9596	15,000	3254	9875	12,500	7000	6100

**Fig. 10.3** Convergence curves for the 25-bar space truss under multiple load cases [4]

(5) A<sub>19</sub>-A<sub>22</sub>, (6) A<sub>23</sub>-A<sub>30</sub>, (7) A<sub>31</sub>-A<sub>34</sub>, (8) A<sub>35</sub>-A<sub>36</sub>, (9) A<sub>37</sub>-A<sub>40</sub>, (10) A<sub>41</sub>-A<sub>48</sub>, (11) A<sub>49</sub>-A<sub>52</sub>, (12) A<sub>53</sub>-A<sub>54</sub>, (13) A<sub>55</sub>-A<sub>58</sub>, (14) A<sub>59</sub>-A<sub>66</sub>, (15) A<sub>67</sub>-A<sub>70</sub>, and (16) A<sub>71</sub>-A<sub>72</sub>. In this example, designs for a multiple load cases using both discrete and continuous design variables are performed. The values and directions of the two load cases applied to the 72-bar spatial truss for both discrete and continuous



**Fig. 10.4** Schematic of a 72-bar space truss

**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)

designs are listed in Table 10.5. The members are subjected to the stress limits of  $\pm 25$  ksi ( $\pm 172.375$  MPa) for both discrete and continuous designs. Maximum displacement limitations of  $\pm 0.25$  in ( $\pm 6.35$  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  $p_a = 0.15$  and number of nests = 7, maximum number of analyses = 21,000.

**Table 10.6** Performance comparison for the 72-bar spatial truss with discrete variables

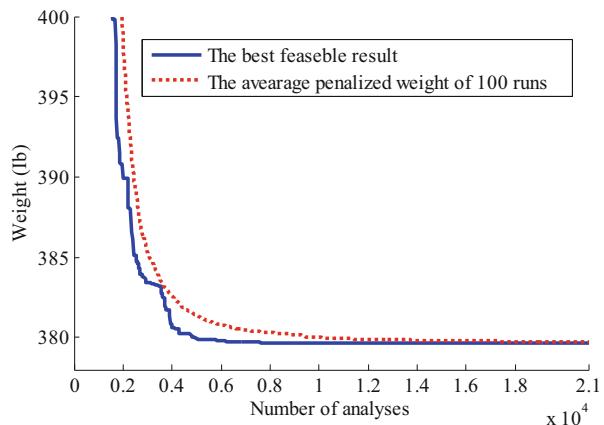
		Optimal cross-sectional areas ( $\text{in}^2$ )						
		GA [15]	PSOPC [15]	HPSO [15]	HPSACO [16]	ICA [15]	Present work [4]	
Element group							$\text{in}^2$	$\text{cm}^2$
1	A <sub>1</sub> –A <sub>4</sub>	0.196	4.490	4.970	1.800	1.99	1.800	11.613
2	A <sub>5</sub> –A <sub>12</sub>	0.602	1.457	1.228	0.442	0.442	0.563	3.632
3	A <sub>13</sub> –A <sub>16</sub>	0.307	0.111	0.111	0.141	0.111	0.111	0.716
4	A <sub>17</sub> –A <sub>18</sub>	0.766	0.111	0.111	0.111	0.141	0.111	0.716
5	A <sub>19</sub> –A <sub>22</sub>	0.391	2.620	2.880	1.228	1.228	1.266	8.168
6	A <sub>23</sub> –A <sub>30</sub>	0.391	1.130	1.457	0.563	0.602	0.563	3.632
7	A <sub>31</sub> –A <sub>34</sub>	0.141	0.196	0.141	0.111	0.111	0.111	0.716
8	A <sub>35</sub> –A <sub>36</sub>	0.111	0.111	0.111	0.111	0.141	0.111	0.716
9	A <sub>37</sub> –A <sub>40</sub>	1.800	1.266	1.563	0.563	0.563	0.563	3.632
10	A <sub>41</sub> –A <sub>48</sub>	0.602	1.457	1.228	0.563	0.563	0.442	2.852
11	A <sub>49</sub> –A <sub>52</sub>	0.141	0.111	0.111	0.111	0.111	0.111	0.716
12	A <sub>53</sub> –A <sub>54</sub>	0.307	0.111	0.196	0.250	0.111	0.111	0.716
13	A <sub>55</sub> –A <sub>58</sub>	1.563	0.442	0.391	0.196	0.196	0.196	1.265
14	A <sub>59</sub> –A <sub>66</sub>	0.766	1.457	1.457	0.563	0.563	0.602	3.884
15	A <sub>67</sub> –A <sub>70</sub>	0.141	1.228	0.766	0.442	0.307	0.391	2.523
16	A <sub>71</sub> –A <sub>72</sub>	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	1734.93 (N)
Number of analyses		N/A	150,000	50,000	5330	4500	4840	

### 10.2.5.5 Design of the 72-Bar Truss Using Discrete Variables

In this case, the discrete variables are selected from 64 discrete values from 0.111 to  $33.5 \text{ in}^2$  (71.613–21,612.860  $\text{mm}^2$ ). 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 (1734.93 N) that is 2.97 lb (13.22 N) lighter than the best result obtained by ICA algorithm [15].

**Fig. 10.5** Convergence curves for the 72-bar space truss with continuous variables [4]



#### 10.2.5.6 Design of the 72-Bar Truss Using Continuous Variables

In this case the minimum value for the cross-sectional areas is  $0.1 \text{ in}^2$  ( $0.6452 \text{ cm}^2$ ), and the maximum value is limited to  $4.00 \text{ in}^2$  ( $25.81 \text{ cm}^2$ ).

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 curves for 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 (1689.37 N), but achieves the 380 lb (1691 N) possible design after 4900 analyses. Table 10.7 compares the results of the CS to those of the previously reported methods in the literature.

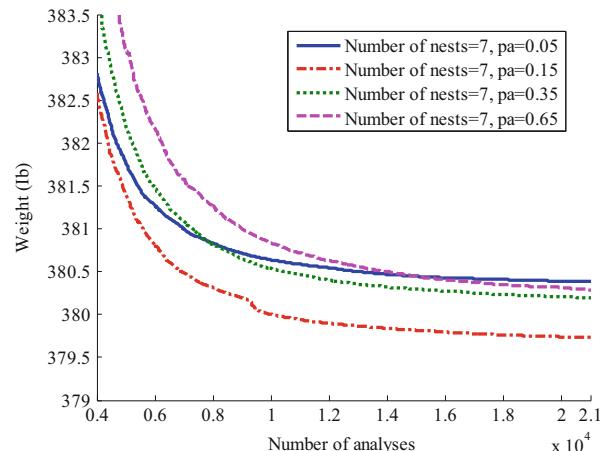
For further studies of one of the two CS parameters, we have tried this example alternatively for constant number of nests as 7 and various amounts of  $pa$  from the  $[0, 1]$  interval with 21,000 as the maximum number of analyses. The convergence curves for the average weight for 100 runs are shown in Fig. 10.6. According to this 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.

#### 10.2.5.7 Design of the 120-Bar Dome-Shaped Truss

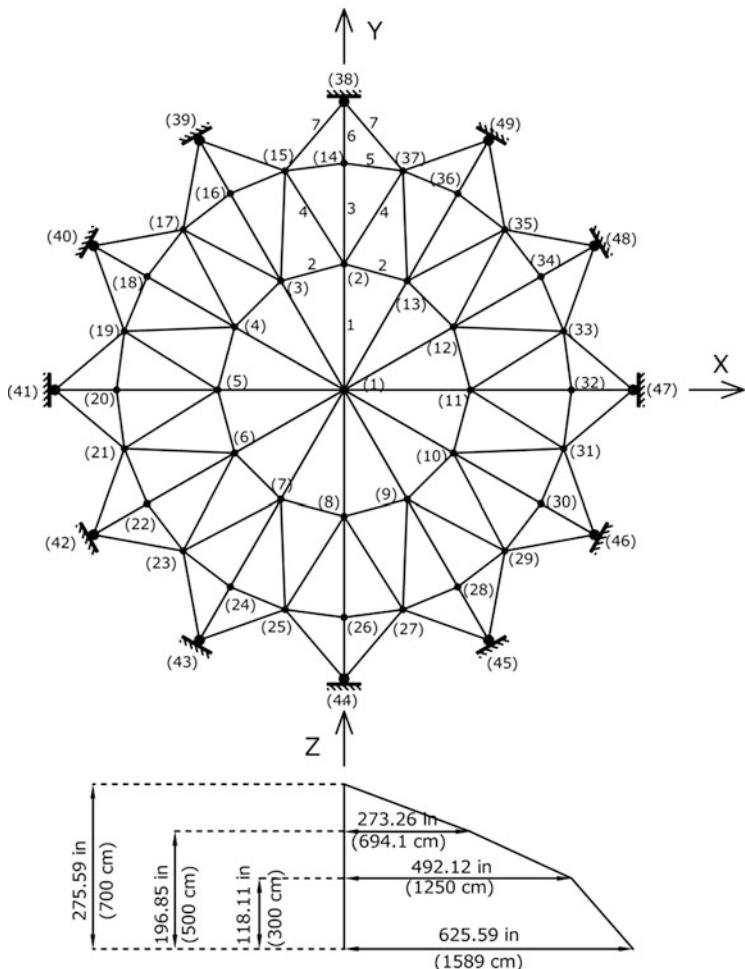
The topology, nodal numbering, and element grouping of the 120-bar dome truss are shown in Fig. 10.7. For clarity, not all the element groups are numbered in this figure. The 120 members are categorized into seven groups, because of symmetry. Other conditions of problem are as follows [8]: The modulus of elasticity is 30,450 ksi (210,000 MPa), and the material density is  $0.288 \text{ lb/in}^3$  ( $7971.810 \text{ kg/m}^3$ ).

**Table 10.7** Performance comparison for the 72-bar spatial truss with continuous variables

Element group	Optimal cross-sectional areas ( $\text{in}^2$ )					
	GA [14]	ACO [9]	PSO [14]	BB–BC [8]	HBB–BC [14]	Present work [4] $\text{in}^2$
1 A <sub>1</sub> –A <sub>4</sub>	1.755	1.948	1.7427	1.8577	1.9042	1.9122
2 A <sub>5</sub> –A <sub>12</sub>	0.505	0.508	0.5185	0.5059	0.5162	0.5101
3 A <sub>13</sub> –A <sub>16</sub>	0.105	0.101	0.1000	0.1000	0.1000	0.1000
4 A <sub>17</sub> –A <sub>18</sub>	0.155	0.102	0.1000	0.1000	0.1000	0.1000
5 A <sub>19</sub> –A <sub>22</sub>	1.155	1.303	1.3079	1.2476	1.2582	1.2577
6 A <sub>23</sub> –A <sub>30</sub>	0.585	0.511	0.5193	0.5269	0.5035	0.5128
7 A <sub>31</sub> –A <sub>34</sub>	0.100	0.101	0.1000	0.1000	0.1000	0.1000
8 A <sub>35</sub> –A <sub>36</sub>	0.100	0.100	0.1000	0.1012	0.1000	0.1000
9 A <sub>37</sub> –A <sub>40</sub>	0.460	0.561	0.5142	0.5209	0.5178	0.5229
10 A <sub>41</sub> –A <sub>48</sub>	0.530	0.492	0.5464	0.5172	0.5214	0.5177
11 A <sub>49</sub> –A <sub>52</sub>	0.120	0.100	0.1000	0.1004	0.1000	0.1000
12 A <sub>53</sub> –A <sub>54</sub>	0.165	0.107	0.1095	0.1005	0.1007	0.1000
13 A <sub>55</sub> –A <sub>58</sub>	0.155	0.156	0.1615	0.1565	0.1566	0.1566
14 A <sub>59</sub> –A <sub>66</sub>	0.535	0.550	0.5092	0.5507	0.5421	0.5406
15 A <sub>67</sub> –A <sub>70</sub>	0.480	0.390	0.4967	0.3922	0.4132	0.4152
16 A <sub>71</sub> –A <sub>72</sub>	0.520	0.592	0.5619	0.5922	0.5756	0.5701
Weight (lb)	385.76	380.24	381.91	379.85	379.66	379.63
Average weight (lb)	N/A	383.16	N/A	382.08	381.85	379.73
Number of analyses	N/A	18,500	N/A	19,621	13,200	10,600

**Fig. 10.6** Convergence curves for the average weight of 100 runs, with constant number of nests and different values of  $pa$  [4]

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



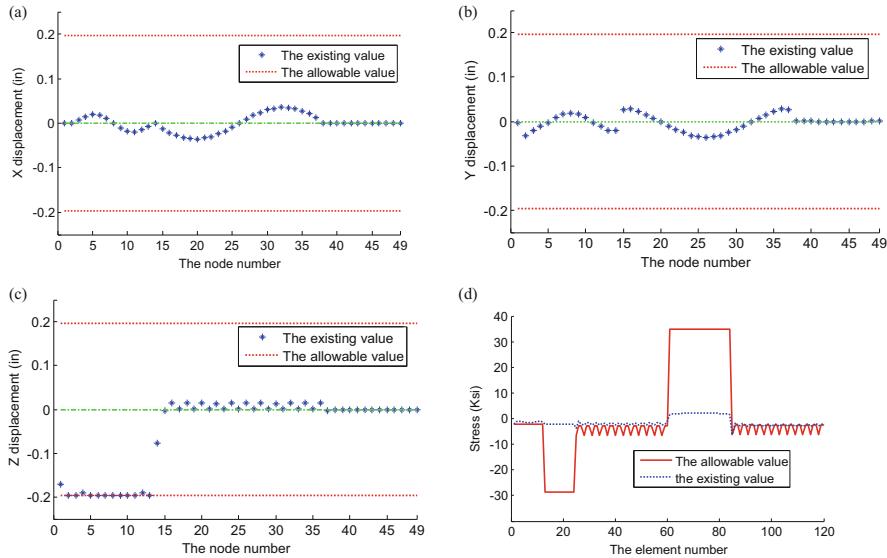
**Fig. 10.7** Schematic of a 120-bar dome-shaped truss

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 Eqs. (10.5) and (10.6)] and displacement limitations of  $\pm 0.1969$  in ( $\pm 5$  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 6300 analyses to find the best solution while this number is equal to 150,000, 32,600, 10,000, 10,000,

**Table 10.8** Performance comparison for the 120-bar dome-shaped truss with continuous variables

		Optimal cross-sectional areas ( $\text{in}^2$ )						
		PSOPC [13]	PSACO [13]	HPSACO [14]	HBB-BC [10]	CSS [20]	ICA [4]	Present work [4] $\text{cm}^2$
1	A <sub>1</sub>	3.040	3.026	3.095	3.037	3.027	3.0244	19.512
2	A <sub>2</sub>	13.149	15.222	14.405	14.431	14.606	14.7168	94.947
3	A <sub>3</sub>	5.646	4.904	5.020	5.130	5.044	5.2446	5.0800
4	A <sub>4</sub>	3.143	3.123	3.352	3.134	3.139	3.1413	3.1374
5	A <sub>5</sub>	8.759	8.341	8.631	8.591	8.543	8.4541	8.5012
6	A <sub>6</sub>	3.758	3.418	3.432	3.377	3.367	3.3567	54.847
7	A <sub>7</sub>	2.502	2.498	2.499	2.500	2.497	2.4947	21.303
Best weight (lb)		33,481.2	33,263.9	33,248.9	33,287.9	33,251.9	33,256.2	16,106
Average weight (lb)		N/A	N/A	N/A	N/A	N/A	33,250.42	147,964.37 (N)
Number of analyses		150,000	32,600	10,000	10,000	7000	6000	147,977.10 (N)
								6300



**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

7000, and 6000 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 [20], 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 [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.

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 performance. In fact, apart from the step size parameter  $\alpha$  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:

$$\begin{aligned}
 & \text{Find } \{x\} = [x_1, x_2, \dots, x_{ng}], x_i \in D_i \\
 & \text{to minimize } W(\{x\}) = \sum_{i=1}^{nm} \rho_i x_i L_i \\
 & \text{subject to : } g_i(\{x\}) \leq 0, j = 1, 2, \dots, n
 \end{aligned} \tag{10.17}$$

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 which include strength constraints of the American Institute of Steel Construction load and resistance factor design (AISC [17]) 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:

$$D_i = (d_{i,1}, d_{i,2}, \dots, d_{i,r(i)}) \quad (10.18)$$

then the problem can be considered as a discrete one, where  $r(i)$  is the number of available discrete values for the  $i$ th 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:

$$f_{cost}(\{X\}) = (1 + \varepsilon_1 \cdot v)^{\varepsilon_2} \times W(\{X\}), \quad v = \sum_{j=1}^n \max[0, g_j(\{x\})] \quad (10.19)$$

where  $n$  represents the number of evaluated constraints for each individual design, and  $v$  denotes the sum of the violations of the design. The constants  $\varepsilon_1$  and  $\varepsilon_2$  are selected considering the exploration and the exploitation rate of the search space. Here,  $\varepsilon_1$  is set to unity;  $\varepsilon_2$  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,  $\varepsilon_2$  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:

$$\frac{\Delta_T}{H} - R \leq 0 \quad (10.20)$$

Inter-story drift constraints:

$$\frac{d_i}{h_i} - R_I \leq 0, \quad i = 1, 2, \dots, ns \quad (10.21)$$

Strength constraints:

$$\begin{aligned} \frac{P_u}{2\varphi P_n} + \left( \frac{M_{ux}}{\varphi_b M_{nx}} + \frac{M_{uy}}{\varphi_b M_{ny}} \right) - 1 &\leq 0, \quad \text{for } \frac{P_u}{\varphi_c P_n} < 0.2 \\ \frac{P_u}{\varphi_c P_n} + \frac{8}{9} \left( \frac{M_{ux}}{\varphi_b M_{nx}} + \frac{M_{uy}}{\varphi_b M_{ny}} \right) - 1 &\leq 0, \quad \text{for } \frac{P_u}{\varphi_c P_n} < 0.2 \end{aligned} \quad (10.22)$$

where  $\Delta_T$  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  $i$ th 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);  $\varphi_c$  is the resistance factor ( $\varphi_c = 0.9$  for tension,  $\varphi_c = 0.85$  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  $\varphi_b$  denotes the flexural resistance reduction factor ( $\varphi_b = 0.90$ ). The nominal tensile strength for yielding in the gross section is computed as:

$$P_n = A_g \cdot F_y \quad (10.23)$$

and the nominal compressive strength of a member is computed as:

$$P_n = A_g \cdot F_{cr} \quad (10.24)$$

$$\begin{aligned} F_{cr} &= \left( 0.658 \lambda_c^2 \right) F_y, \quad \text{for } \lambda_c \leq 1.5 \\ F_{cr} &= \left( \frac{0.877}{\lambda_c^2} \right) F_y, \quad \text{for } \lambda_c > 1.5 \end{aligned} \quad (10.25)$$

$$\lambda_c = \frac{kl}{r\pi} \sqrt{\frac{F_y}{E}} \quad (10.26)$$

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 [18].

### 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 LRFD-AISC 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. The first example is also used for adjusting algorithm parameters, and the obtained results are used in other examples.

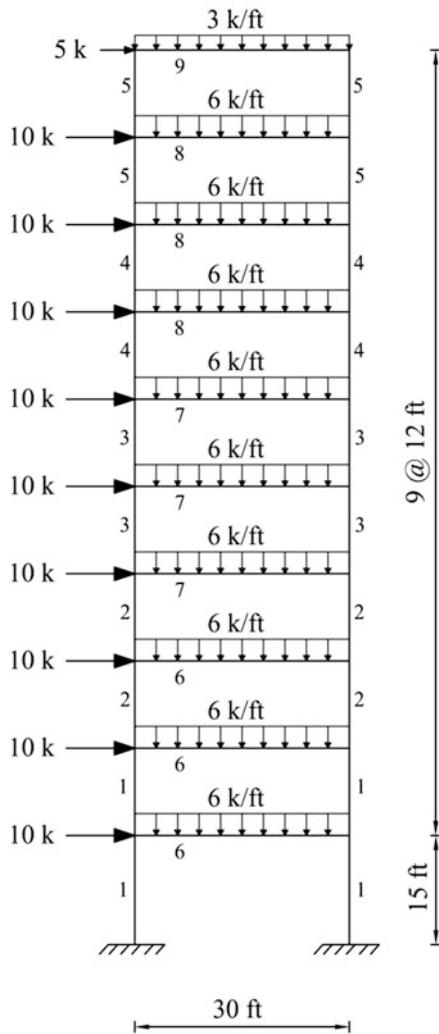
#### 10.3.3.1 A 1-Bay 10-Story Frame

Figure 10.9 shows the topology, the service loading conditions, and the numbering of member groups for a 1-bay 10-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 LRFD–AISC-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 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 [18], 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  $n = 15$  to 25 and  $pa = 0.15$  to 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 to 10 and various amounts 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

**Fig. 10.9** Schematic of a 1-bay 10-story frame



optimal weight and the number of analyses in the format: average  $\pm$  one standard deviation (success rate) (Yang [1]). Therefore,  $62,088.25 \pm 41.66$  (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  $pa = 0.3$  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, 25] 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

**Table 10.9** Performance of the CS for 1-bay 10-story frame with various amounts of  $pa$ 

CS parameters	Best run		100 runs	
	Min weight (lb)	Min $Noa$	Weight (lb)	$Noa$
$Pa = 0.1, n = 10$	62,074.368	9320	$62,195.35 \pm 188.77$ (45 %)	$17,400 \pm 5900$ (45 %)
$Pa = 0.2, n = 10$	62,074.368	6040	$62,133.05 \pm 149.07$ (74 %)	$16,480 \pm 5600$ (74 %)
$Pa = 0.25, n = 10$	62,074.368	6980	$62,111.01 \pm 130.74$ (81 %)	$14,360 \pm 5360$ (81 %)
$Pa = 0.3, n = 10$	62,074.368	7100	$62,088.25 \pm 41.66$ (90 %)	$14,640 \pm 4520$ (90 %)
$Pa = 0.4, n = 10$	62,074.368	6280	$62,110.47 \pm 60.9$ (74 %)	$13,780 \pm 3800$ (74 %)
$Pa = 0.5, n = 10$	62,074.368	8820	$62,156.84 \pm 173.54$ (63 %)	$15,120 \pm 4580$ (63 %)

**Table 10.10** Performance of the CS for 1-bay 10-story frame with various amounts of  $n$ 

CS parameters	Best run		100 runs	
	Min weight (lb)	Min $Noa$	Weight (lb)	$Noa$
$n = 5, Pa = 0.3$	62,074.368	4560	$62,672.28 \pm 854.16$ (41 %)	$8140 \pm 4640$ (41 %)
$n = 7, Pa = 0.3$	62,074.368	4438	$62,186.96 \pm 240.12$ (58 %)	$10,528 \pm 3920$ (58 %)
$n = 10, Pa = 0.3$	62,074.368	7100	$62,088.25 \pm 41.66$ (90 %)	$14,640 \pm 4520$ (90 %)
$n = 15, Pa = 0.3$	62,074.368	11,610	$62,109.71 \pm 66.48$ (76 %)	$19,920 \pm 4860$ (76 %)
$n = 20, Pa = 0.3$	62,074.368	13,280	$62,116.00 \pm 66.89$ (71 %)	$23,320 \pm 3800$ (71 %)
$n = 25, Pa = 0.3$	62,074.368	16,400	$62,177.38 \pm 137.74$ (45 %)	$24,900 \pm 3250$ (45 %)

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 in 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. [19] using GA, Camp et al. [20] 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 is obtained by IACO algorithm. The best design developed by CS weighted 62,074 (lb) with 4438 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  $63,308 \pm 684$  and  $63,279 \pm 618$  for ACO and IACO algorithms, respectively, and  $62,923 \pm 1.74$  for 30 runs by HS; CS results in  $62,186.96 \pm 240.12$  for 100 runs that is better than others.

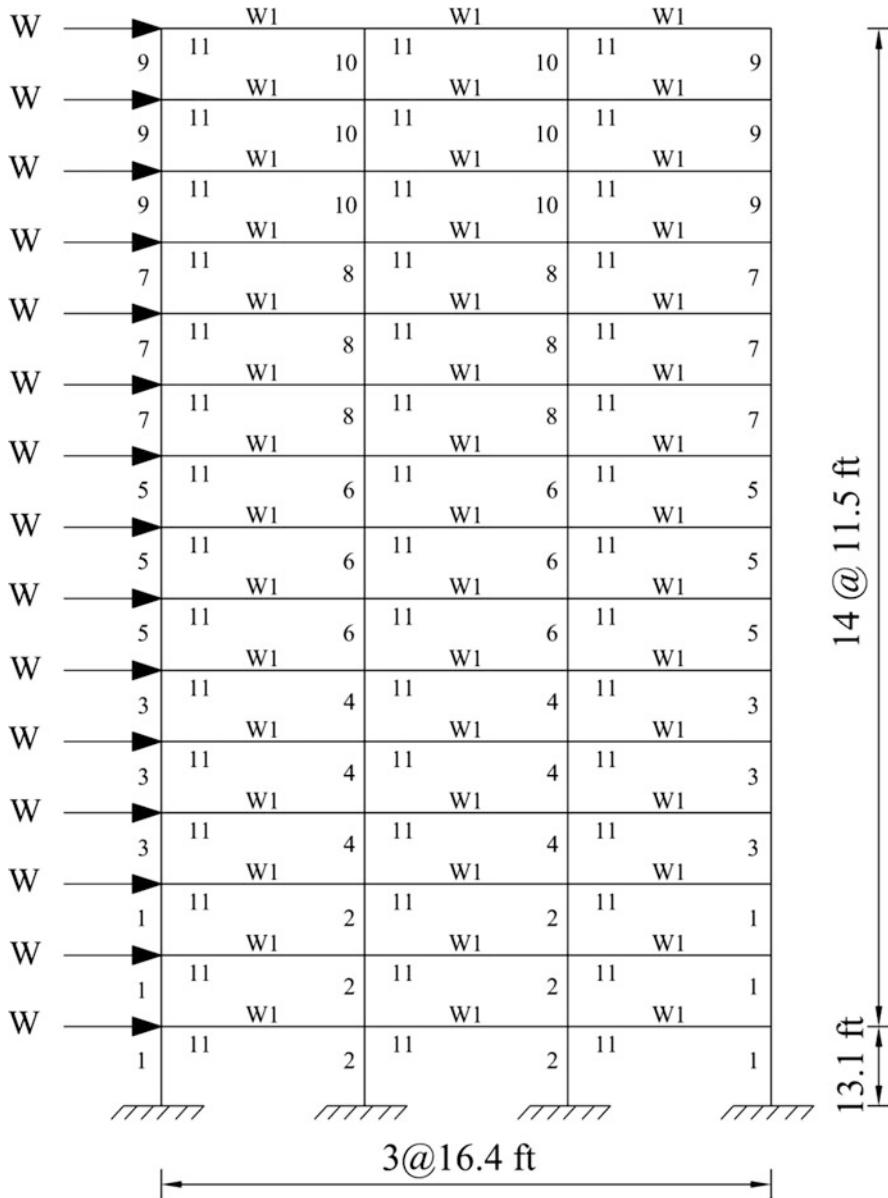
**Table 10.11** Performance comparison for the 1-bay 10-story frame

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	3000	5100	3690	2500	4438

### 10.3.3.2 A 3-Bay 15-Story Frame

The configuration, the service loading conditions, and the numbering of member groups for a 3-bay 15-story frame are shown in Fig. 10.10. The loads are  $W = 6.75$  kips and  $w_1 = 3.42$  kips/ft. All 64 columns are grouped into nine 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  $87,784 \pm 942$  lb. The convergence curves 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 4700 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 tenth story, and 0.46 in for fourth story, respectively.



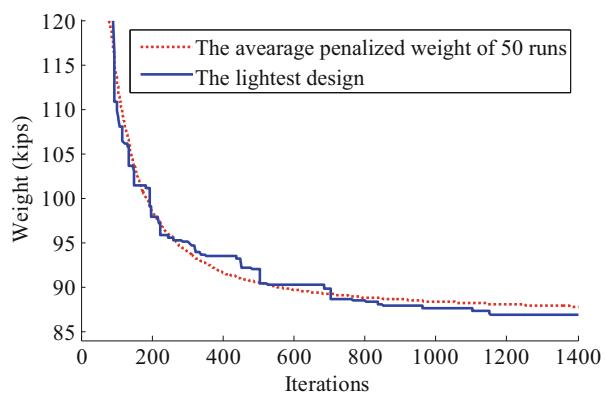
**Fig. 10.10** Schematic of a 3-bay 15-story frame [5]

### 10.3.3.3 A 3-Bay 24-Story Frame

The last example is a 3-bay 24-story frame shown in Fig. 10.12. Camp et al. [20], 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

**Table 10.12** Performance comparison for the 3-bay 15-story frame

Element group	Optimal W-shaped sections			
	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	9500	6000	16,170

**Fig. 10.11** The best and average design convergence curves of the 3-bay 15-story frame [5]

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  $W = 5761.85$  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.

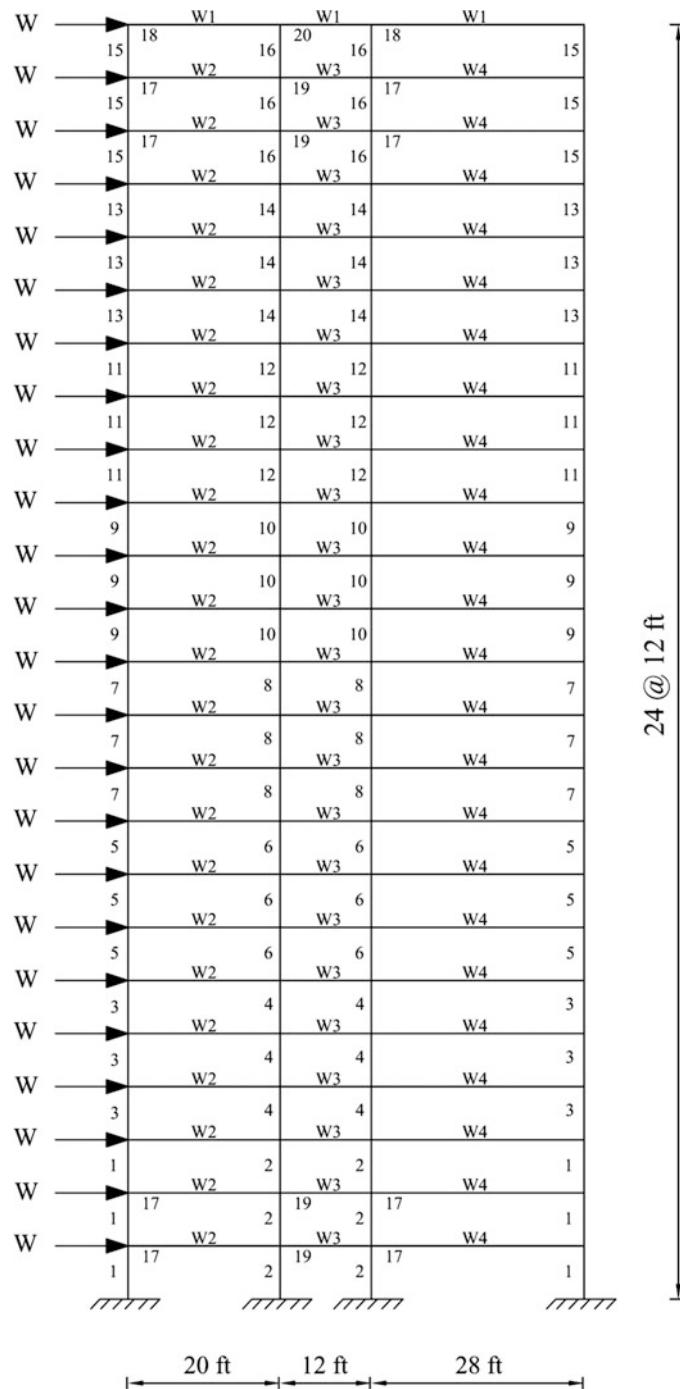


Fig. 10.12 Schematic of a 3-bay 24-story frame

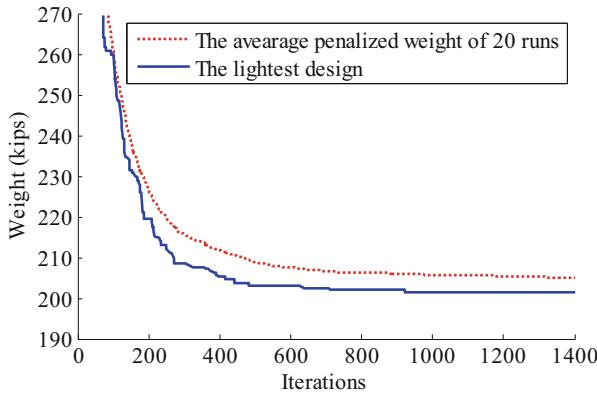
**Table 10.13** Performance comparison for the 3-bay 24-story frame

Element group	Optimal W-shaped sections				
	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	9924	10,500	7500	15,918

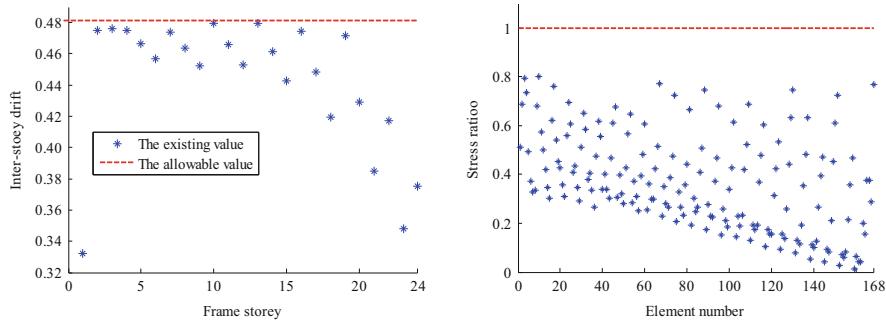
The optimum designs of the algorithms are listed in Table 10.13. The best design of previous works is due to ICA algorithm, weights 212,640 lb with 10,500 as the number of analyses. The lightest design of 20 independent runs for CS algorithm weights 201,451 lb that is 5.3 % lighter than developed by ICA.

The convergence curves 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 3528 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 4533 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  $229,555 \pm 4561$  lb and  $222,620 \pm 5800$  lb with 15,500 and 14,651 as average number of analyses, respectively.

The maximum value of sway at the top story, stress ratio, and inter-story drift is 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 and 17 stories.



**Fig. 10.13** The best and average design history of 3-bay 24-story frame [5]



**Fig. 10.14** Comparison of allowable and existing values for 3-bay 24-story frame using the CS [5]. (a) Inter-story drift. (b) Stress ratio

### 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 LRFD–AISC specifications. The CS algorithm is comprised of three major components as follows: 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 curves of the examples, the ability of algorithm to local search is not very efficient as its exploration.

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  $\alpha$ , 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.

Three steel frames with various numbers 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, UK
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. Iran 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, IL
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. Kaveh A, Talatahari S (2008) Ant colony optimization for design of space trusses. Int J Space Struct 23:167–181
12. Lee KS, Geem ZW (2004) A new structural optimization method based on the harmony search algorithm. Comput Struct 82:781–798
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 Constr Steel Res 65:1558–1568

17. AISC (2001) Manual of steel construction: load and resistance factor design. AISC, Chicago, IL
18. Dumonteil P (1992) Simple equations for effective length factors. Eng J AISE 29(3):1115
19. Pezeshk S, Camp CV, Chen D (2000) Design of nonlinear framed structures using genetic optimization. J Struct Eng ASCE 126:382–388
20. Camp CV, Bichon BJ, Stovall SP (2005) Design of steel frames using ant colony optimization. J Struct Eng ASCE 131:369–379
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 sociopolitically 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, forms 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 toward optimum points. Here, ICA is utilized to optimize the skeletal structures which are based on [1, 2].

This algorithm is proposed by Atashpaz et al. [3, 4] and is a sociopolitically 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 imperialists 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 address optimization of skeletal structures problems which can be considered as a suitable field to investigate the efficiency of the new algorithm. This chapter covers both the discrete and continuous structural design problems. The comparison of the results of the ICA with some well-known metaheuristics demonstrates the efficiency of the present algorithm.

## 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:

$$\begin{aligned} \text{Find } \quad & \{x\} = [x_1, x_2, \dots, x_{ng}], \\ & x_i \in D_i \\ \text{to minimize } & W(\{x\}) = \sum_{i=1}^{nm} \rho_i \cdot x_i \cdot L_i \\ \text{subject to: } & g_j(\{x\}) \leq 0 \quad j = 1, 2, \dots, n \end{aligned} \quad (11.1)$$

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;  $\rho_i$  denotes the material density of member  $i$ ;  $L_i$  and  $x_i$  are the length and the cross section 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:

$$D_i = \{x_i | x_i \in [x_{i,\min}, x_{i,\max}]\} \quad (11.2)$$

where  $x_{i,\min}$  and  $x_{i,\max}$  are minimum and maximum allowable values for the design variable  $i$ , respectively. If the design variables are supposed to be selected from a list:

$$D_i = \{d_{i,1}, d_{i,2}, \dots, d_{i,r(i)}\} \quad (11.3)$$

then the problem is considered as a discrete one, where  $r(i)$  is the number of available discrete values for the  $i$ th 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:

$$f_{\text{cost}}(\{x\}) = (1 + \varepsilon_1 \cdot v)^{\varepsilon_2} \times W(\{x\}), \quad v = \sum_{i=1}^n \max[0, v_i] \quad (11.4)$$

where  $n$  represents the number of evaluated constraints for each individual design. The constants  $\varepsilon_1$  and  $\varepsilon_2$  are selected considering the exploration and the exploitation rate of the search space. Here,  $\varepsilon_1$  is set to unity;  $\varepsilon_2$  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,  $\varepsilon_2$  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 Constraints for Truss Structures

For truss structures, the stress limitations of the members are imposed according to the provisions of ASD-AISC [6] as follows:

$$\begin{cases} \sigma_i^+ = 0.6F_y & \text{for } \sigma_i \geq 0 \\ \sigma_i^- & \text{for } \sigma_i < 0 \end{cases} \quad (11.5)$$

where  $\sigma_i^-$  is calculated according to the slenderness ratio:

$$\sigma_i^- = \begin{cases} \left[ \left( 1 - \frac{\lambda_i^2}{2C_C^2} \right) F_y \right] / \left( \frac{5}{3} + \frac{3\lambda_i}{8C_C} - \frac{\lambda_i^3}{8C_C^3} \right) & \text{for } \lambda_i < C_C \\ \frac{12\pi^2 E}{23\lambda_i^2} & \text{for } \lambda_i \geq C_C \end{cases} \quad (11.6)$$

where  $E$  is the modulus of elasticity;  $F_y$  is the yield stress of steel;  $C_C$  denotes the slenderness ratio ( $\lambda_i$ ) dividing the elastic and inelastic buckling regions; and  $\lambda_i$  represents the slenderness ratio.

The other constraint is the limitation of the nodal displacements:

$$\delta_i \leq \delta_i^u \quad i = 1, 2, \dots, nn \quad (11.7)$$

where  $\delta_i$  is the nodal deflection;  $\delta_i^u$  is the allowable deflection of node  $i$ ; and  $nn$  is the number of nodes.

### 11.2.2 Constraints for Steel Frames

Optimal design of frame structures is subjected to the following constraints according to LRFD–AISC provisions [7]:

#### Maximum lateral displacement

$$\frac{\Delta_T}{H} \leq R \quad (11.8)$$

### Inter-story displacement constraints

$$i = 1, 2, \dots, ns \quad \frac{d_i}{h_i} \leq R_I, \quad (11.9)$$

### The strength constraints

$$\begin{aligned} \frac{P_u}{2\varphi_c P_n} + \left( \frac{M_{ux}}{\varphi_b M_{nx}} + \frac{M_{uy}}{\varphi_b M_{ny}} \right) &\leq 1, \quad \text{For } \frac{P_u}{\varphi_c P_n} < 0.2 \\ \frac{P_u}{\varphi_c P_n} + \frac{8}{9} \left( \frac{M_{ux}}{\varphi_b M_{nx}} + \frac{M_{uy}}{\varphi_b M_{ny}} \right) &\leq 1, \quad \text{For } \frac{P_u}{\varphi_c P_n} \geq 0.2 \end{aligned} \quad (11.10)$$

where  $\Delta_T$  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  $i$ th 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);  $\varphi_c$  is the resistance factor ( $\varphi_c = 0.9$  for tension,  $\varphi_c = 0.85$  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  $\varphi_b$  denotes the flexural resistance reduction factor ( $\varphi_b = 0.90$ ). The nominal tensile strength for yielding in the gross section is computed as:

$$P_n = A_g \cdot F_y \quad (11.11)$$

and the nominal compressive strength of a member is computed as:

$$P_n = A_g \cdot F_{cr} \quad (11.12)$$

$$\begin{aligned} F_{cr} &= \left( 0.658 \lambda_c^2 \right) F_y, \quad \text{For } \lambda_c \leq 1.5 \\ F_{cr} &= \left( \frac{0.877}{\lambda_c^2} \right) F_y, \quad \text{For } \lambda_c > 1.5 \end{aligned} \quad (11.13)$$

$$\lambda_c = \frac{kl}{r\pi} \sqrt{\frac{F_y}{E}} \quad (11.14)$$

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 steps of the algorithm are 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:

$$x_{i,j}^{(0)} = x_{i,\min} + \text{rand} \cdot (x_{i,\max} - x_{i,\min}) \quad (11.15)$$

where  $x_{i,j}^{(0)}$  determines the initial value of the  $i$ th variable for the  $j$ th country;  $x_{i,\min}$  and  $x_{i,\max}$  are the minimum and the maximum allowable values for the  $i$ th variable; and  $\text{rand}$  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 Eq. (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  $N_{\text{country}}$ , and the number of the most powerful countries to form the empires is equal to  $N_{\text{imp}}$ . The remaining  $N_{\text{col}}$  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 proportional to its power. In order to proportionally divide the colonies among the imperialists, a normalized cost for an imperialist is defined as:

$$C_j = f_{\text{cost}}^{(\text{imp},j)} - \max_i \left( f_{\text{cost}}^{(\text{imp},i)} \right) \quad (11.16)$$

where  $f_{\text{cost}}^{(\text{imp},j)}$  is the cost of the  $j$ th imperialist and  $C_j$  is its normalized cost. The colonies are divided among empires based on their power or normalized cost, and for the  $j$ th empire it will be as follows:

$$NC_j = \text{Round} \left( \left| \frac{C_j}{\sum_{i=1}^{N_{imp}} C_i} \right| \cdot N_{col} \right) \quad (11.17)$$

where  $NC_j$  is the initial number of colonies associated with the  $j$ th empire which are 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]:

$$\{x\}_{new} = \{x\}_{old} + U(0, \beta \times d) \times \{V_1\} \quad (11.18)$$

where  $\beta$  is a parameter with a value greater than one and  $d$  is the distance between colony and imperialist.  $\beta > 1$  persuades the colonies to get closer to the imperialist state from both sides.  $\beta >> 1$  gradually results in a divergence of colonies from the imperialist state, while a very close value to 1 for  $\beta$  reduces the search ability of the algorithm.  $\{V_1\}$  is a vector for which the starting 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  $\theta$ . In this figure  $\theta$  is a random number with uniform distribution as:

$$\theta = U(-\gamma, +\gamma) \quad (11.19)$$

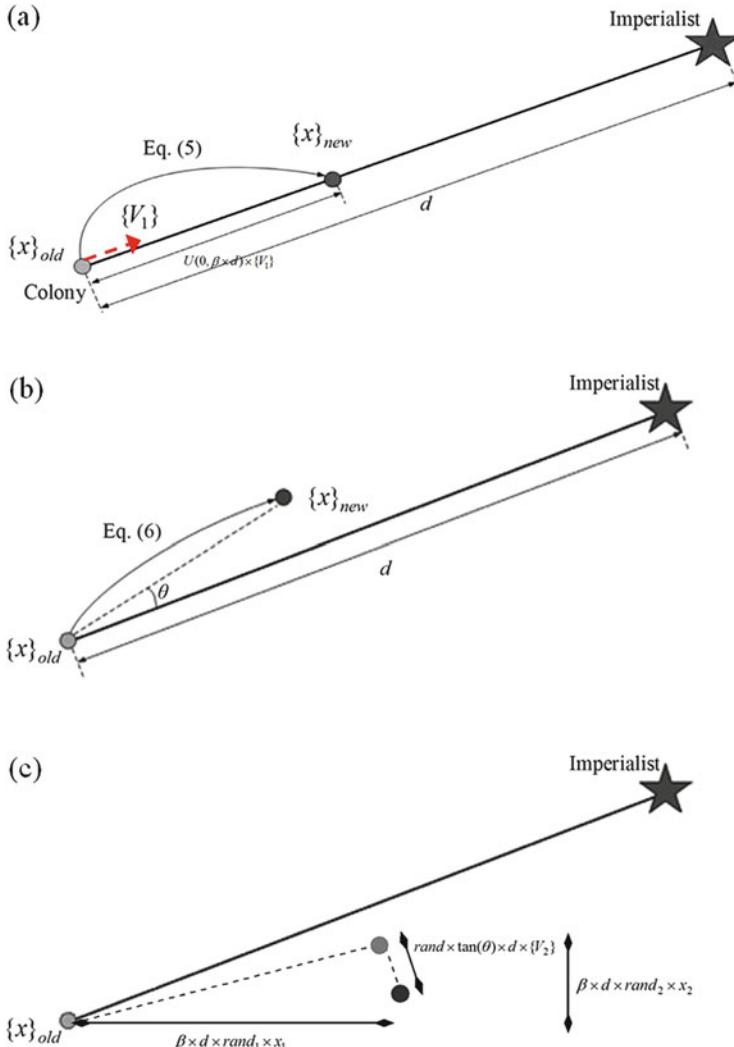
where  $\gamma$  is a parameter that adjusts the deviation from the original direction. In most of the implementations, a value of about 2 for  $\beta$  [3] and about 0.1 (Rad) for  $\gamma$  results in a good convergence of the countries to the global minimum.

In order to improve the performance of the ICA, we change the movement step as follows:

First: different random values are utilized for different components of the solution vector in place of only one value [Eq. (11.18)] as:

$$\{x\}_{new} = \{x\}_{old} + \beta \times d \times \{rand\} \otimes \{V_1\} \quad (11.20)$$

where  $\{V_1\}$  is the base vector starting from the previous location of the colony and directed to the imperialist;  $\{rand\}$  is a random vector and the sign “ $\otimes$ ” denotes an element-by-element multiplication. Since these random values are not necessarily



**Fig. 11.1** Movements 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

the same, the colony is deviated automatically without using the definition of  $\theta$ . However, for having a suitable exploration ability, the utilization of  $\theta$  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 utilizing this vector in place of using  $\theta$  as:

$$\begin{aligned} \{x\}_{new} = & \{x\}_{old} + \beta \times d \times \{rand\} \otimes \{V_1\} + U(-1, +1) \times \tan(\theta) \times \\ & d \times \{V_2\}, \quad \{V_1\} \cdot \{V_2\} = 0, \quad \|\{V_2\}\| = 1 \end{aligned} \quad (11.21)$$

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 above, 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 the empires will have a likelihood of taking possession of the mentioned colonies.

The 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:

$$TC_j = f_{cost}^{(imp,j)} + \xi \cdot \frac{\sum_{i=1}^{NC_j} f_{cost}^{(col,i)}}{NC_j} \quad (11.22)$$

where  $TC_n$  is the total cost of the  $j$ th empire and  $\xi$  is a positive number which is considered to be less than 1. A small value for  $\xi$  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  $\xi$  is found to be a suitable value in most of the implementations [3]. Similar to Eq. (11.16), the normalized total cost is defined as:

$$NTC_j = TC_j - \max_i (TC_i) \quad (11.23)$$

where  $NTC_j$  is the normalized total cost of the  $j$ th empire. Having the normalized total cost, the possession probability of each empire is evaluated by:

$$P_j = \left| \frac{NTC_j}{\sum_{i=1}^{N_{imp}} NTC_i} \right| \quad (11.24)$$

### 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 predefined value or the amount of improvement in the best result reduces to a predefined value, the searching process is stopped.

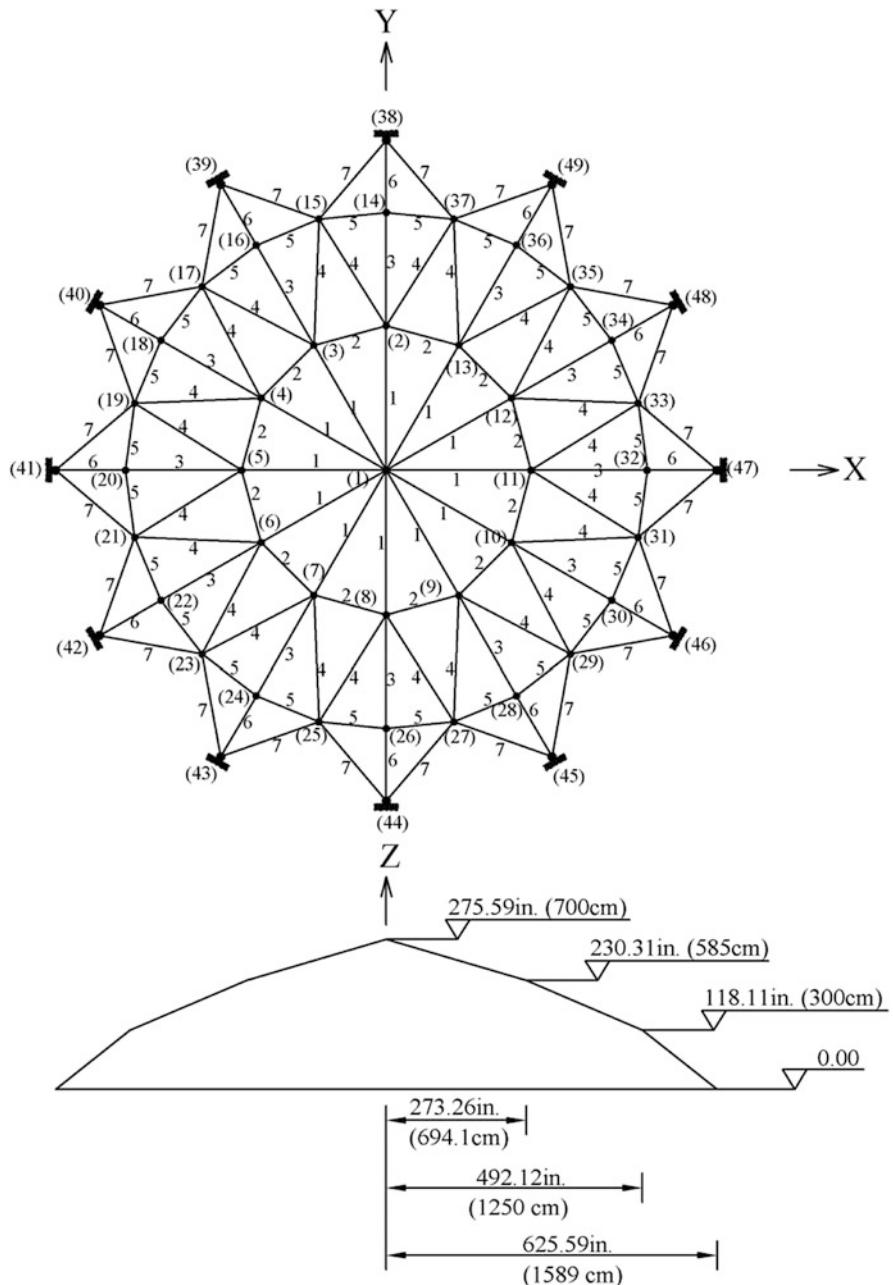
The movement of colonies toward their relevant imperialist states along with competition among empires and also the collapse mechanism will hopefully cause all the countries to converge to a state in which there exists just one empire in the world and all the other countries are colonies of that empire. 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 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<sup>3</sup> (7971.810 kg/m<sup>3</sup>). 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



**Fig. 11.2** Schematic of a 120-bar dome-shaped truss

**Table 11.1** Performance comparison for the 120-bar dome truss

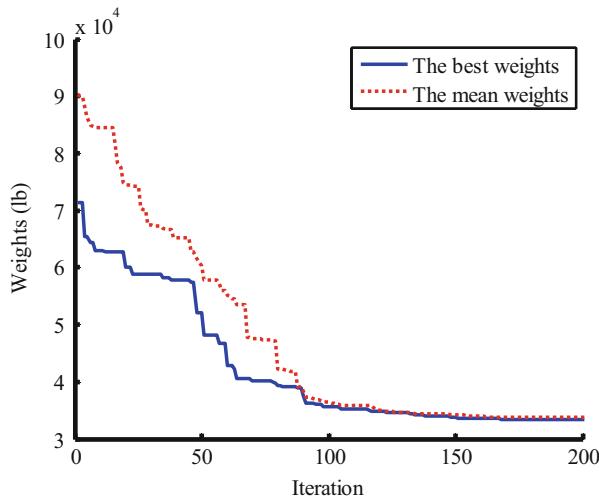
		Optimal cross-sectional areas (in. <sup>2</sup> )					Present work [2]	
Element group		PSOPC [13]	PSACO [13]	HPSACO [13]	HBB–BC [14]	CSS [6]	<i>in</i> <sup>2</sup>	<i>cm</i> <sup>2</sup>
1	A <sub>1</sub>	3.040	3.026	3.095	3.037	3.027	3.0275	19.532
2	A <sub>2</sub>	13.149	15.222	14.405	14.431	14.606	14.4596	93.288
3	A <sub>3</sub>	5.646	4.904	5.020	5.130	5.044	5.2446	33.836
4	A <sub>4</sub>	3.143	3.123	3.352	3.134	3.139	3.1413	20.266
5	A <sub>5</sub>	8.759	8.341	8.631	8.591	8.543	8.4541	54.543
6	A <sub>6</sub>	3.758	3.418	3.432	3.377	3.367	3.3567	21.656
7	A <sub>7</sub>	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	7000	6000	

the rest of the nodes. The minimum cross-sectional area of all members is 0.775 in<sup>2</sup> (2 cm<sup>2</sup>), and the maximum cross-sectional area is taken as 20.0 in<sup>2</sup> (129.03 cm<sup>2</sup>). The constraints are stress constraints [as defined by Eqs. (11.5), (11.6)] and displacement limitations of  $\pm 0.1969$  in ( $\pm 5$  mm) imposed on all nodes in *x*, *y*, and *z* directions.

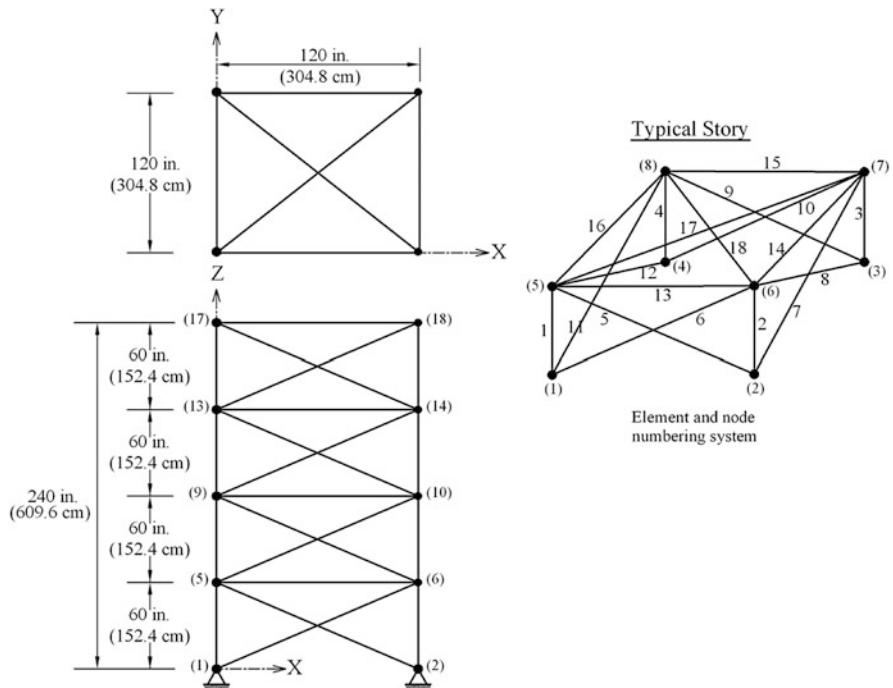
Table 11.1 shows the best solution vectors, the corresponding weights, and the required number of analyses for convergence of the present algorithm and some other metaheuristic algorithms. ICA-based algorithm needs 6000 analyses to find the best solution while this number is equal to 150,000, 32,600, 10,000, 10,000, and 7000 analyses for a PSO-based algorithm [9]; a PSO and ACO hybrid algorithm [9]; a combined algorithm based on PSO, ACO, and HS [9]; an improved BB–BC method using PSO properties [10]; and the CSS algorithm [11], 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 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 %.

### 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<sup>3</sup> (2767.990 kg/m<sup>3</sup>), and the modulus of elasticity is 10,000 ksi (68,950 MPa). The members are subjected to the stress limits of  $\pm 25$  ksi



**Fig. 11.3** Convergence curves for the dome-shaped truss obtained by the ICA [2]



**Fig. 11.4** Schematic of a 72-bar spatial truss

( $\pm 172.375$  MPa). The nodes are subjected to the displacement limits of  $\pm 0.25$  in ( $\pm 0.635$  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)  $A_{17}-A_{18}$ , (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

**Table 11.2** The available cross-section areas of the AISC code

No.	in. <sup>2</sup>	mm <sup>2</sup>	No.	in. <sup>2</sup>	mm <sup>2</sup>
1	0.111	(71.613)	33	3.840	(2477.414)
2	0.141	(90.968)	34	3.870	(2496.769)
3	0.196	(126.451)	35	3.880	(2503.221)
4	0.250	(161.290)	36	4.180	(2696.769)
5	0.307	(198.064)	37	4.220	(2722.575)
6	0.391	(252.258)	38	4.490	(2896.768)
7	0.442	(285.161)	39	4.590	(2961.284)
8	0.563	(363.225)	40	4.800	(3096.768)
9	0.602	(388.386)	41	4.970	(3206.445)
10	0.766	(494.193)	42	5.120	(3303.219)
11	0.785	(506.451)	43	5.740	(3703.218)
12	0.994	(641.289)	44	7.220	(4658.055)
13	1.000	(645.160)	45	7.970	(5141.925)
14	1.228	(792.256)	46	8.530	(5503.215)
15	1.266	(816.773)	47	9.300	(5999.988)
16	1.457	(939.998)	48	10.850	(6999.986)
17	1.563	(1008.385)	49	11.500	(7419.430)
18	1.620	(1045.159)	50	13.500	(8709.660)
19	1.800	(1161.288)	51	13.900	(8967.724)
20	1.990	(1283.868)	52	14.200	(9161.272)
21	2.130	(1374.191)	53	15.500	(9999.980)
22	2.380	(1535.481)	54	16.000	(10,322.560)
23	2.620	(1690.319)	55	16.900	(10,903.204)
24	2.630	(1696.771)	56	18.800	(12,129.008)
25	2.880	(1858.061)	57	19.900	(12,838.684)
26	2.930	(1890.319)	58	22.000	(14,193.520)
27	3.090	(1993.544)	59	22.900	(14,774.164)
28	1.130	(729.031)	60	24.500	(15,806.420)
29	3.380	(2180.641)	61	26.500	(17,096.740)
30	3.470	(2238.705)	62	28.000	(18,064.480)
31	3.550	(2290.318)	63	30.000	(19,354.800)
32	3.630	(2341.931)	64	33.500	(21,612.860)

**Table 11.3** Loading conditions for the 72-bar spatial truss

Node	Case 1			Case 2		
	P <sub>X</sub> kips (kN)	P <sub>Y</sub> kips (kN)	P <sub>Z</sub> kips (kN)	P <sub>X</sub>	P <sub>Y</sub>	P <sub>Z</sub> 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 11.4** Optimal design comparison for the 72-bar spatial truss

Element group	Optimal cross-sectional areas (in <sup>2</sup> )				
	GA [12]	PSOPC [13]	HPSO [13]	HPSACO [9]	Present work [2]
1 A <sub>1</sub> ~ A <sub>4</sub>	0.196	4.490	4.970	1.800	1.99
2 A <sub>5</sub> ~ A <sub>12</sub>	0.602	1.457	1.228	0.442	0.442
3 A <sub>13</sub> ~ A <sub>16</sub>	0.307	0.111	0.111	0.141	0.111
4 A <sub>17</sub> ~ A <sub>18</sub>	0.766	0.111	0.111	0.111	0.141
5 A <sub>19</sub> ~ A <sub>22</sub>	0.391	2.620	2.880	1.228	1.228
6 A <sub>23</sub> ~ A <sub>30</sub>	0.391	1.130	1.457	0.563	0.602
7 A <sub>31</sub> ~ A <sub>34</sub>	0.141	0.196	0.141	0.111	0.111
8 A <sub>35</sub> ~ A <sub>36</sub>	0.111	0.111	0.111	0.111	0.141
9 A <sub>37</sub> ~ A <sub>40</sub>	1.800	1.266	1.563	0.563	0.563
10 A <sub>41</sub> ~ A <sub>48</sub>	0.602	1.457	1.228	0.563	0.563
11 A <sub>49</sub> ~ A <sub>52</sub>	0.141	0.111	0.111	0.111	0.111
12 A <sub>53</sub> ~ A <sub>54</sub>	0.307	0.111	0.196	0.250	0.111
13 A <sub>55</sub> ~ A <sub>58</sub>	1.563	0.442	0.391	0.196	0.196
14 A <sub>59</sub> ~ A <sub>66</sub>	0.766	1.457	1.457	0.563	0.563
15 A <sub>67</sub> ~ A <sub>70</sub>	0.141	1.228	0.766	0.442	0.307
16 A <sub>71</sub> ~ A <sub>72</sub>	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	5330	4500

(178.43 kg) for the HPSACO [8]. The weight of the GA-based algorithm is equal to 427.203 lb (193.77 kg) [12]. The PSOPC and the standard PSO algorithms do not find optimal results when the maximum number of iterations is reached [13]. The HPSO and HPSACO algorithms get the optimal solution after 50,000 [13] and 5330 [9] analyses while it takes only 4500 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 3-bay 15-story frame structure [5] is shown in Fig. 11.5. The displacement and AISC combined strength constraints are the performance constraints of this frame. The sway of the top story is limited to 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 considered as non-braced along its length, and the unbraced length for each beam member is specified as one-fifth of the span length.

The optimum design of the frame is obtained after 6000 analyses by using the ICA, having the minimum weight of 417.46 kN (93.85 kips). The optimum designs for HBB–BC [14], HPSACO, PSOPC, and PSO [5] have the weights of 434.54 (97.65 kN), 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 9900 analyses [14], and the HSPACO needs 6800 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 inter-story drift is equal to 1.04 cm.

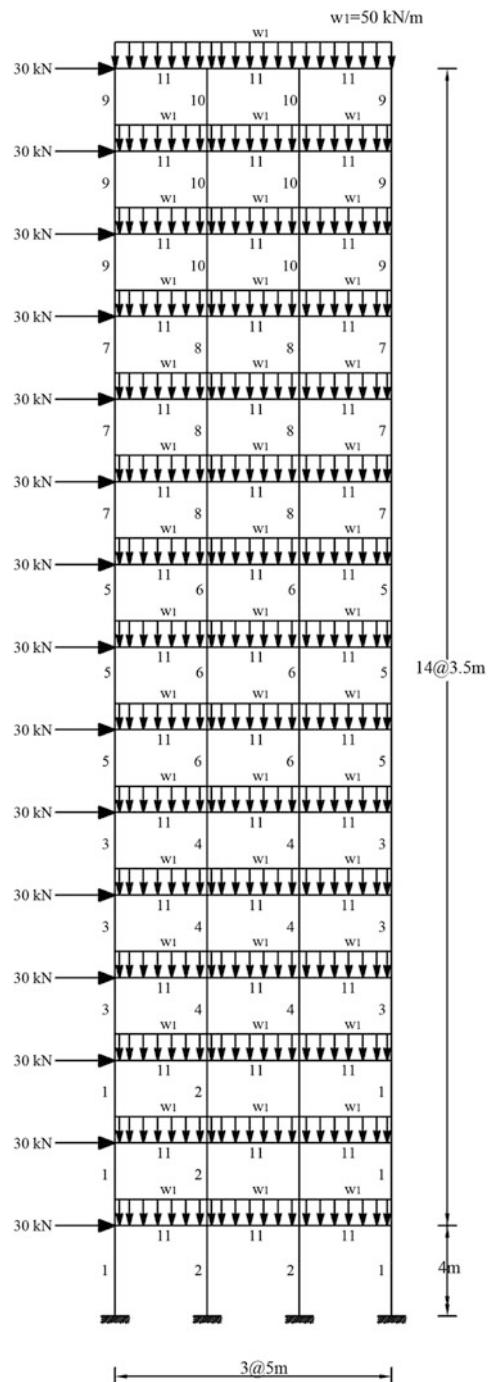
### 11.4.4 Design of a 3-Bay 24-Story Frame

Figure 11.7 shows the topology and the service loading conditions of a 3-bay 24-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], and Kaveh and Talatahari utilized a hybrid PSO and BB–BC algorithm to solve this example [14].

The frame is designed following the LRFD specifications 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 Ref. [14].

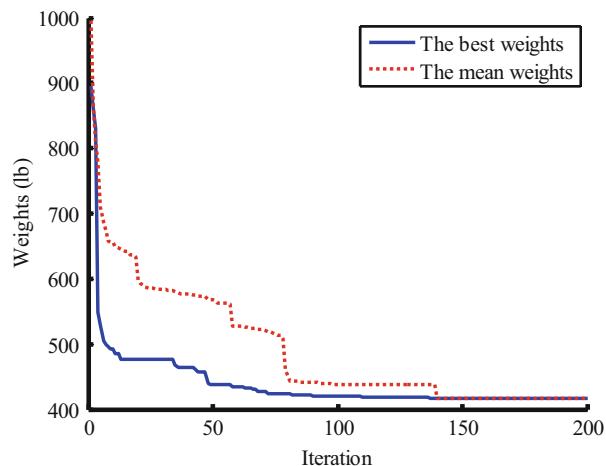
Table 11.6 lists the designs developed by the ICA, the HBB–BC algorithm [14], the ant colony algorithm [16], and harmony search [17]. The ICA algorithm required 7500 frame analyses to converge to a solution, while 10,500 analyses were required by HBB–BC [14], 15,500 analyses by ACO [16], and 13,924 analyses 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 [14], respectively. The global sway at the top story is 25.52 cm (10.05 in.) which is less than the maximum sway. The maximum

**Fig. 11.5** Schematic of a 3-bay 15-story frame



**Table 11.5** Optimal design comparison for the 3-bay 15-story frame

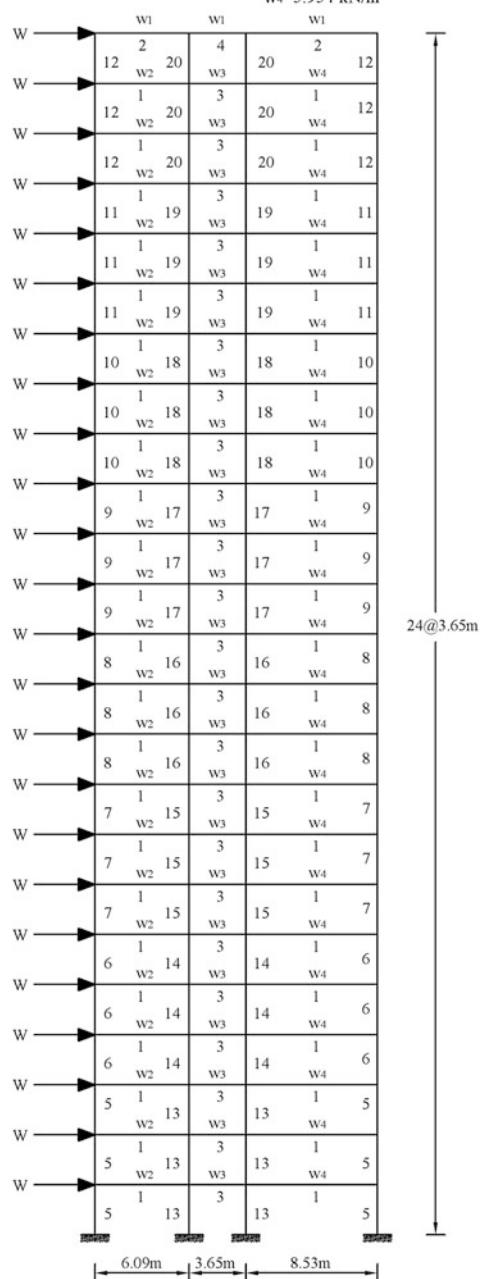
Element group	Optimal W-shaped sections				
	PSO [5]	PSOPC [5]	HPSACO [5]	HBB-BC [14]	Present work [2]
1	W33 × 118	W26 × 129	W21 × 111	W24 × 117	W24 × 117
2	W33 × 263	W24 × 131	W18 × 158	W21 × 132	W21 × 147
3	W24 × 76	W24 × 103	W10 × 88	W12 × 95	W27 × 84
4	W36 × 256	W33 × 141	W30 × 116	W18 × 119	W27 × 114
5	W21 × 73	W24 × 104	W21 × 83	W21 × 93	W14 × 74
6	W18 × 86	W10 × 88	W24 × 103	W18 × 97	W18 × 86
7	W18 × 65	W14 × 74	W21 × 55	W18 × 76	W12 × 96
8	W21 × 68	W26 × 94	W26 × 114	W18 × 65	W24 × 68
9	W18 × 60	W21 × 57	W10 × 33	W18 × 60	W10 × 39
10	W18 × 65	W18 × 71	W18 × 46	W10 × 39	W12 × 40
11	W21 × 44	W21 × 44	W21 × 44	W21 × 48	W21 × 44
Weight (kN)	496.68	452.34	426.36	434.54	417.466
No. of required analyses	50,000	50,000	6800	9900	6000

**Fig. 11.6** Convergence curves for the 3-bay 15-story frame obtained by the ICA [2]

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.

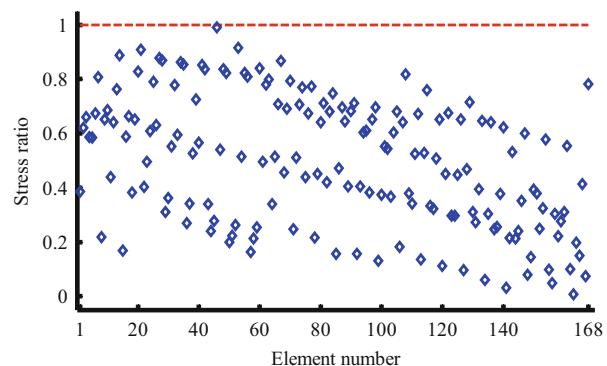
**Fig. 11.7** Schematic of a 3-bay 24-story frame

$W=25,628 \text{ kN}$   
 $w_1=4,378 \text{ kN/m}$   
 $w_2=6,362 \text{ kN/m}$   
 $w_3=6,917 \text{ kN/m}$   
 $w_4=5,954 \text{ kN/m}$



**Table 11.6** Optimal design comparison for the 3-bay 24-story frame

Element group	Optimal W-shaped sections			
	Camp et al. [16]	Degertekin [17]		
	ACO	HS	HBB-BC [14]	Present work [2]
1	W30 × 90	W30 × 90	W30 × 90	W30 × 90
2	W8 × 18	W10 × 22	W21 × 48	W21 × 50
3	W24 × 55	W18 × 40	W18 × 46	W24 × 55
4	W8 × 21	W12 × 16	W8 × 21	W8 × 28
5	W14 × 145	W14 × 176	W14 × 176	W14 × 109
6	W14 × 132	W14 × 176	W14 × 159	W14 × 159
7	W14 × 132	W14 × 132	W14 × 109	W14 × 120
8	W14 × 132	W14 × 109	W14 × 90	W14 × 90
9	W14 × 68	W14 × 82	W14 × 82	W14 × 74
10	W14 × 53	W14 × 74	W14 × 74	W14 × 68
11	W14 × 43	W14 × 34	W14 × 38	W14 × 30
12	W14 × 43	W14 × 22	W14 × 30	W14 × 38
13	W14 × 145	W14 × 145	W14 × 159	W14 × 159
14	W14 × 145	W14 × 132	W14 × 132	W14 × 132
15	W14 × 120	W14 × 109	W14 × 109	W14 × 99
16	W14 × 90	W14 × 82	W14 × 82	W14 × 82
17	W14 × 90	W14 × 61	W14 × 68	W14 × 68
18	W14 × 61	W14 × 48	W14 × 48	W14 × 48
19	W14 × 30	W14 × 30	W14 × 34	W14 × 34
20	W14 × 26	W14 × 22	W14 × 26	W14 × 22
Weight (kN)	980.63	956.13	960.90	946.25
No. of required analyses	15,500	13,924	10,500	7500

**Fig. 11.8** The values of the stress ratios of elements for the ICA result [2]

## 11.5 Discussions

Many of the 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 imperialist competitive algorithm (ICA) which is a sociopolitically motivated optimization algorithm.

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 toward the powerful imperialist and thus to the optimum points. 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 and (2) deviation through orthogonal colony-imperialistic contacting line instead of using  $\theta$ .

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 previous location of agents (velocity), while the PSO requires two positions saving memory (the current position and the previous 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, Chapter 5 of a book titled: Harmony search algorithms for structural design. Springer, Berlin, Heidelberg
6. American Institute of Steel Construction (AISC) (1989) Manual of steel construction—allowable stress design, 9th edn. AISC, Chicago, IL
7. American Institute of Steel Construction (AISC) (2001) Manual of steel construction—load resistance factor design, 3rd edn. AISC, Chicago, IL
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 (2009) Particle swarm optimizer, ant colony strategy and harmony search scheme hybridized for optimization of truss structures. *Comput Struct* 87(5–6):267–283
10. Kaveh A, Talatahari S (2009) Size optimization of space trusses using Big Bang–Big Crunch algorithm. *Comput Struct* 87(17–18):1129–1140
11. Kaveh A, Talatahari S (2010) A novel heuristic optimization method: charged system search. *Acta Mech* 213(3–4):267–286
12. Wu SJ, Chow PT (1995) Steady-state genetic algorithms for discrete optimization of trusses. *Comput Struct* 56(6):979–991
13. 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
14. 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
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 metaheuristic 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 are 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

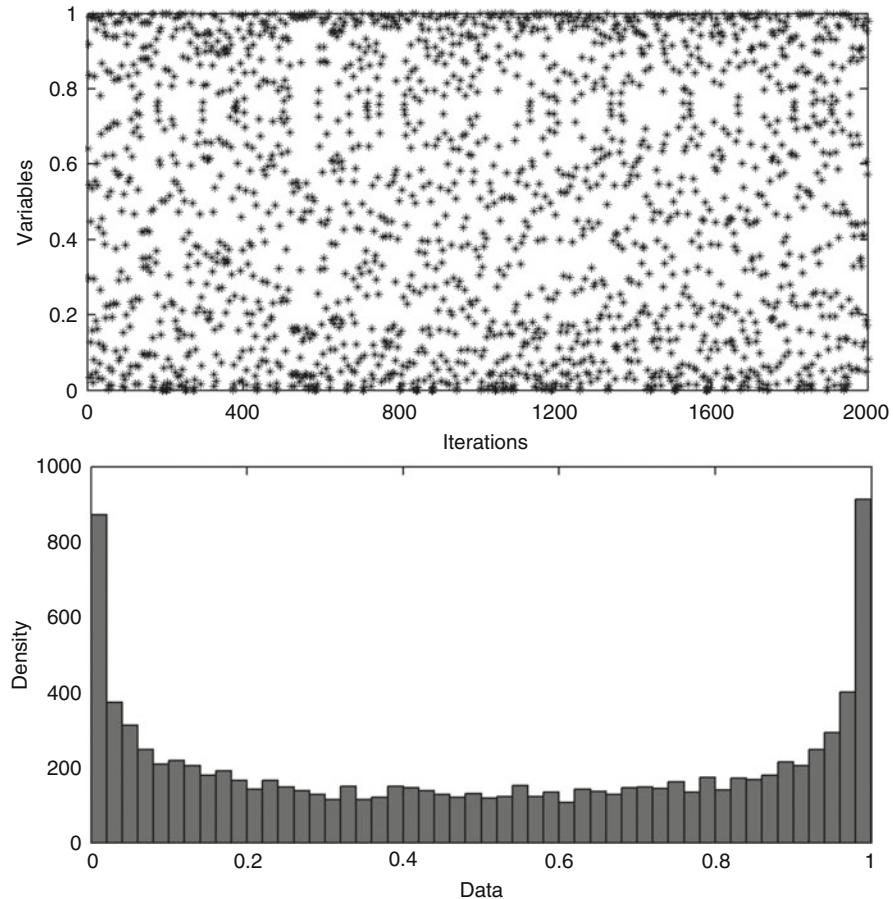
parameters in the metaheuristic's equations. The convergence properties of metaheuristics are closely connected to the random sequence applied on their operators during a run. In particular, when starting some optimizations with different random numbers, experience shows that the results may be very close but not equal and require different numbers of generations to reach the same optimal value. The random number generation algorithms, on which most used metaheuristic 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 metaheuristic 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, where it describes the evolution of these algorithms along with some improvements, their combinations with various methods, and 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 multiphase 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 mathematics 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



**Fig. 12.1** An example of chaotic map (logistic map)

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.

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:

$$x_{t+1} = F(x_t) \quad (12.1)$$

where  $x(t) \in \Re^n$ ,  $t = 1, 2, 3, \dots$ , and  $F$  are a map from  $\Re^n$  to itself.

Let  $F^{(p)}$  denote 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  $\epsilon$ -neighborhood  $N_\epsilon(x)$  of a point  $x$  is defined by:

$$N_\varepsilon(x) = \{y \in \Re^n \mid \|x - y\| \leq \varepsilon\} \quad (12.2)$$

where  $\|\cdot\|$  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 exists 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  $x, y \in S$  ( $x \neq y$ )

$$\lim_{n \rightarrow \infty} \sup \|F^{(n)}(x) - F^{(n)}(y)\| > 0,$$

and for any  $x \in S$  and any periodic point  $y$  of  $F$ ,

$$\lim_{n \rightarrow \infty} \sup \|F^{(n)}(x) - F^{(n)}(y)\| > 0.$$

- (c) There exists an uncountable subset  $S_0 \subset S$  such that for any  $x, y \in S_0$ ,

$$\lim_{n \rightarrow \infty} \inf \|F^{(n)}(x) - F^{(n)}(y)\| = 0$$

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 famous 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 is evidencing chaotic behavior (May [10]):

$$x_{k+1} = ax_k(1 - x_k) \quad (12.3)$$

In this equation,  $x_k$  is the  $k$ th chaotic number, with  $k$  denoting the iteration number. Obviously,  $x_k \in (0, 1)$  under the conditions that the initial  $x_0 \in (0, 1)$  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:

$$x_{k+1} = \begin{cases} x_k/0.7 & x_k < 0.7 \\ 10/3x_k(1 - x_k) & otherwise \end{cases} \quad (12.4)$$

### 12.2.3 Sinusoidal Map

This iterator (May [10]) is represented by:

$$x_{k+1} = ax_k^2 \sin(\pi x_k) \quad (12.5)$$

For  $a = 2.3$  and  $x_0 = 0.7$ , it has the following simplified form:

$$x_{k+1} = \sin(\pi x_k) \quad (12.6)$$

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:

$$\begin{aligned}x_{k+1} &= \begin{cases} 0 & x_k = 0 \\ 1/x_k \bmod(1) & \text{otherwise} \end{cases} \\ 1/x_k \bmod(1) &= \frac{1}{x_k} - \left[ \frac{1}{x_k} \right]\end{aligned}\quad (12.7)$$

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:

$$x_{k+1} = x_k + b - (a/2\pi) \sin(2\pi x_k) \bmod(1) \quad (12.8)$$

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:

$$x_{k+1} = 2.3(x_k)^{2 \sin(\pi x_k)} \quad (12.9)$$

### 12.2.7 Hénon Map

This map is a nonlinear two-dimensional map most frequently employed for testing purposes, and it is represented by:

$$x_{k+1} = 1 - ax_k^2 + bx_{k-1} \quad (12.10)$$

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  $v = 400$ ,  $r = 3$ , and  $\alpha = 12.6695$ . 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 chaos optimization algorithms (COA).

## 12.4 Chaotic Update of Internal Parameters for Metaheuristics

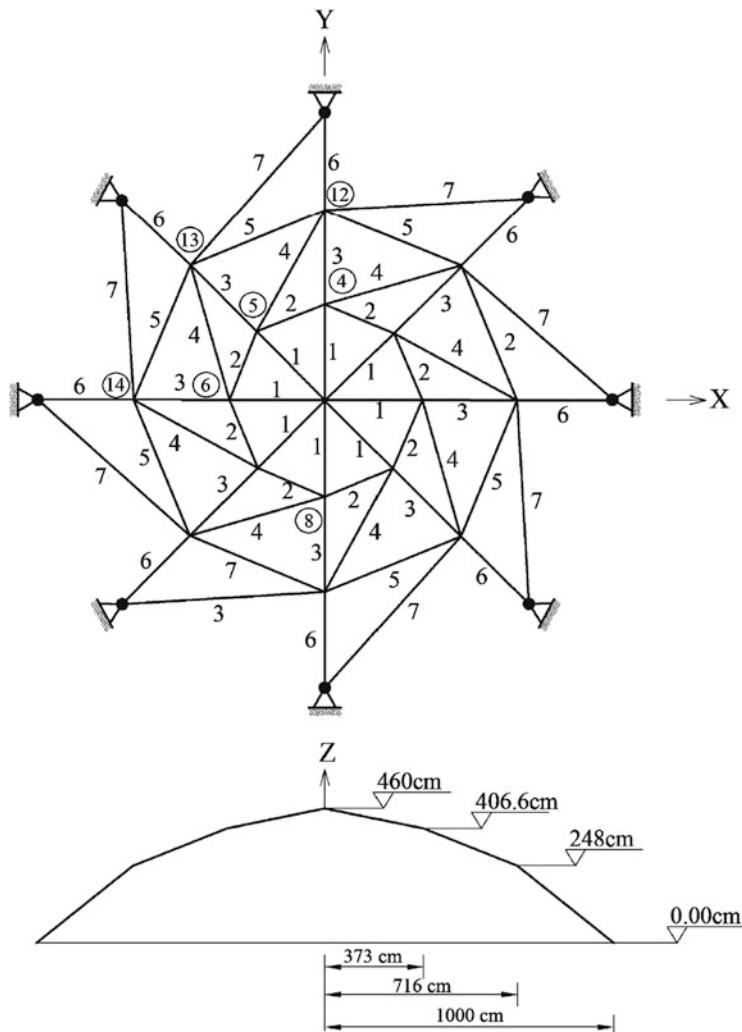
For simulating complex phenomena, sampling, numerical analysis, decision making, and in particular in metaheuristic optimization, random sequences are needed with a long period and reasonable uniformity. On the other hand, as mentioned before, chaos is a deterministic, random-like process found in nonlinear dynamical system which is non-periodic, 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 systems may be considered as sources of randomness (Schuster [18]; Coelho and Mariani [19]; Alatas [20]).

However, metaheuristics are nontypical; 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. Similar utilizations 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]) has 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 the original paper 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 original ICA, especially when the standard deviations of the results are compared. The standard deviation of the new algorithm is much better than the original 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;



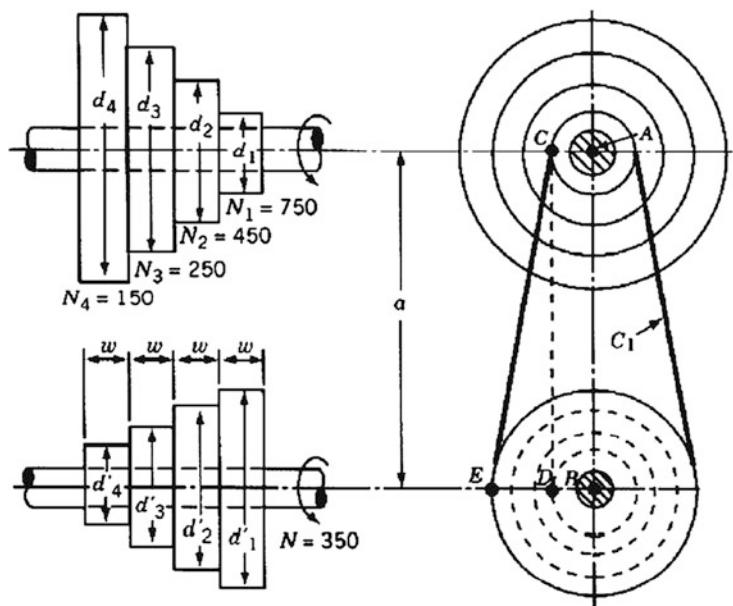
**Fig. 12.2** Schematic of a 56-bar dome spatial truss structure [26]

**Table 12.1** Optimal design comparison for the 56-bar dome truss

	<i>ICA</i>	<i>CICA-1</i>	<i>CICA-2</i>	<i>CICA-3</i>	<i>CICA-4</i>
<i>Best weight (kg)</i>	546.14	546.13	546.16	546.15	546.15
<i>Average weight (kg)</i>	547.91	546.21	546.31	546.24	546.34
<i>Std dev (kg)</i>	5.791	0.49	0.62	0.56	0.59

design of a welded beam; design of a 4-story, 2-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 four-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 the design of a four-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 three are equality constraints and the remaining are inequality constraints. The constraints are imposed to assure the same belt length for all the steps, tension ratios, and power transmitted by the belt. The four-step pulley is designed to transmit at least 0.75 hp ( $0.75 \cdot 745.6998$  W), with an input speed of 350 rpm and output speeds of 750, 450, 250, and 150 rpm. This problem is 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].



**Fig. 12.3** Schematic of a four-step cone pulley [28]

**Table 12.2** Statistical results of the four-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

Due to the simplicity and potentials 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 chaos optimization algorithm (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:

**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:

$$x_i^0 = X_{L_i} + cm_i^0(X_{U_i} - X_{L_i}), \quad i = 1, 2, \dots, n \quad (12.13)$$

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:

$$x_i^{k+1} = X_{L_i} + cm_i^{k+1}(X_{U_i} - X_{L_i}), \quad i = 1, 2, \dots, n \quad (12.14)$$

**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.

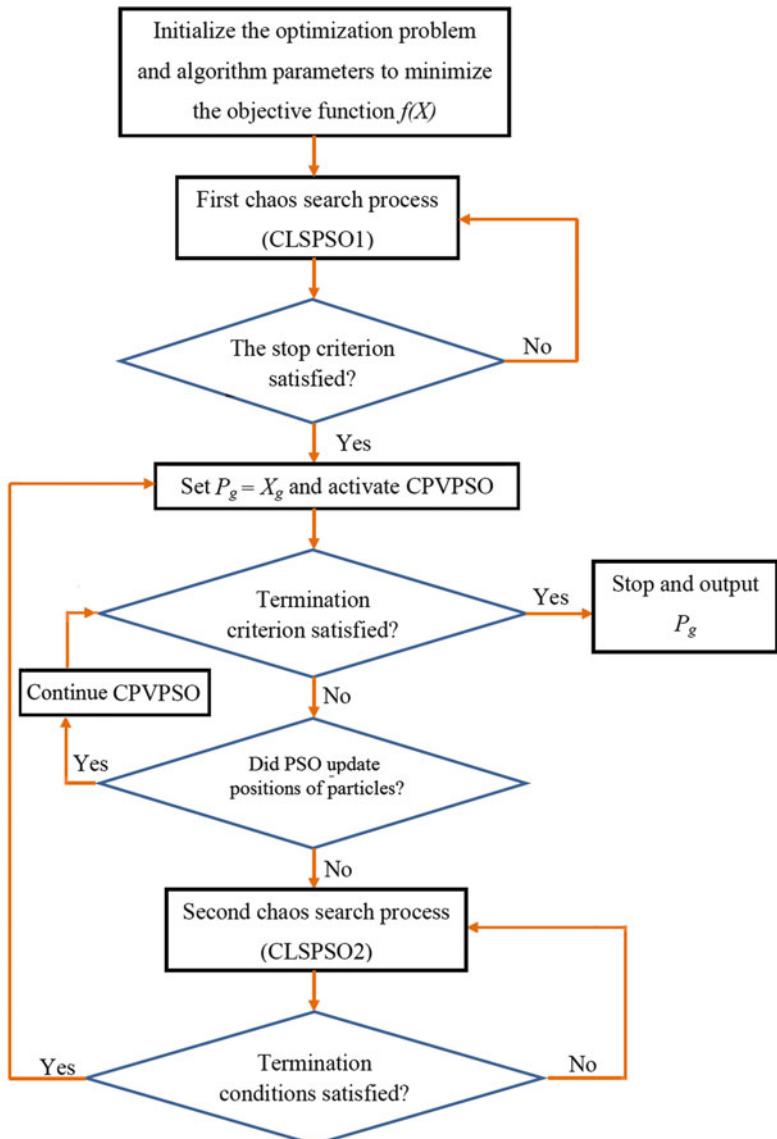
Else if  $k > N$  is satisfied then stop.

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], and 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 nonlinear 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

Chaotic swarming of particles (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 predefined 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.



**Fig. 12.4** Flowchart of the CSP algorithm

### 12.6.1 The Original 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  $j$ th particle at the  $k$ th 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:

$$\begin{aligned} V_j^{k+1} &= wV_j^k + c_1 \times r_{1j}^k \otimes (P_j^k - X_j^k) + c_2 \times r_{2j}^k \otimes (P_g^k - X_j^k) \\ X_j^{k+1} &= X_j^k + V_j^k \end{aligned} \quad (12.15)$$

where  $w$  is an inertia weight to control the influence of the previous velocity;  $r_{1j}^k$  and  $r_{2j}^k$  are random numbers uniformly distributed in the range of  $(0,1)$ ;  $c_1$  and  $c_2$  are two acceleration constants, namely, cognitive and social parameter, respectively;  $P_j^k$  is the best position of the  $j$ th particle up to iteration  $k$ ; and  $P_g^k$  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:

$$w^{k+1} = w^k \times D_r \times rand \quad (12.16)$$

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

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 Eq. (12.3).

In order to control the values of PSO parameters by using chaotic maps,  $r_{1j}^k$ ,  $r_{2j}^k$ , and  $rand$  are generated from the iterations of logistic map instead of using classical random number generator as:

$$\begin{aligned} V_j^{k+1} &= w^k \times V_j^k + c_1 \times cm^k \otimes (P_j^k - X_j^k) + c_2 \times cm^k \otimes (P_g^k - X_j^k) \\ w^{k+1} &= w^k \times D_r \times cm^k \end{aligned} \quad (12.17)$$

### 12.6.3 The CLSPSO Phase

In this phase, COA is introduced in the PSO formulation. This is a kind of multiphase 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  $j$ th particle as:

$$X_j^0 = X_{\min} + cm_j^0(X_{\max} - X_{\min}) \quad (12.18)$$

**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:

$$X_j^{t+1} = X_g^t + (2cm_j^t - 1)(X_g^t - X_j^t) \quad (12.19)$$

**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  $P_g^k$ .

**Step 3:** Set  $X_g^i = P_g^k$ .

**Step 4:** Update the global best position of the particles using the chaotic map by the following equation:

$$X_g^{i+1} = X_g^i + (2cm^i - 1) \frac{X_{\max} - X_{\min}}{k} \quad (12.20)$$

**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  $P_g^k = X_g^{i+1}$ .

**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  $P_g^k$  and  $f^*$  as the result of the CLSPSO2.

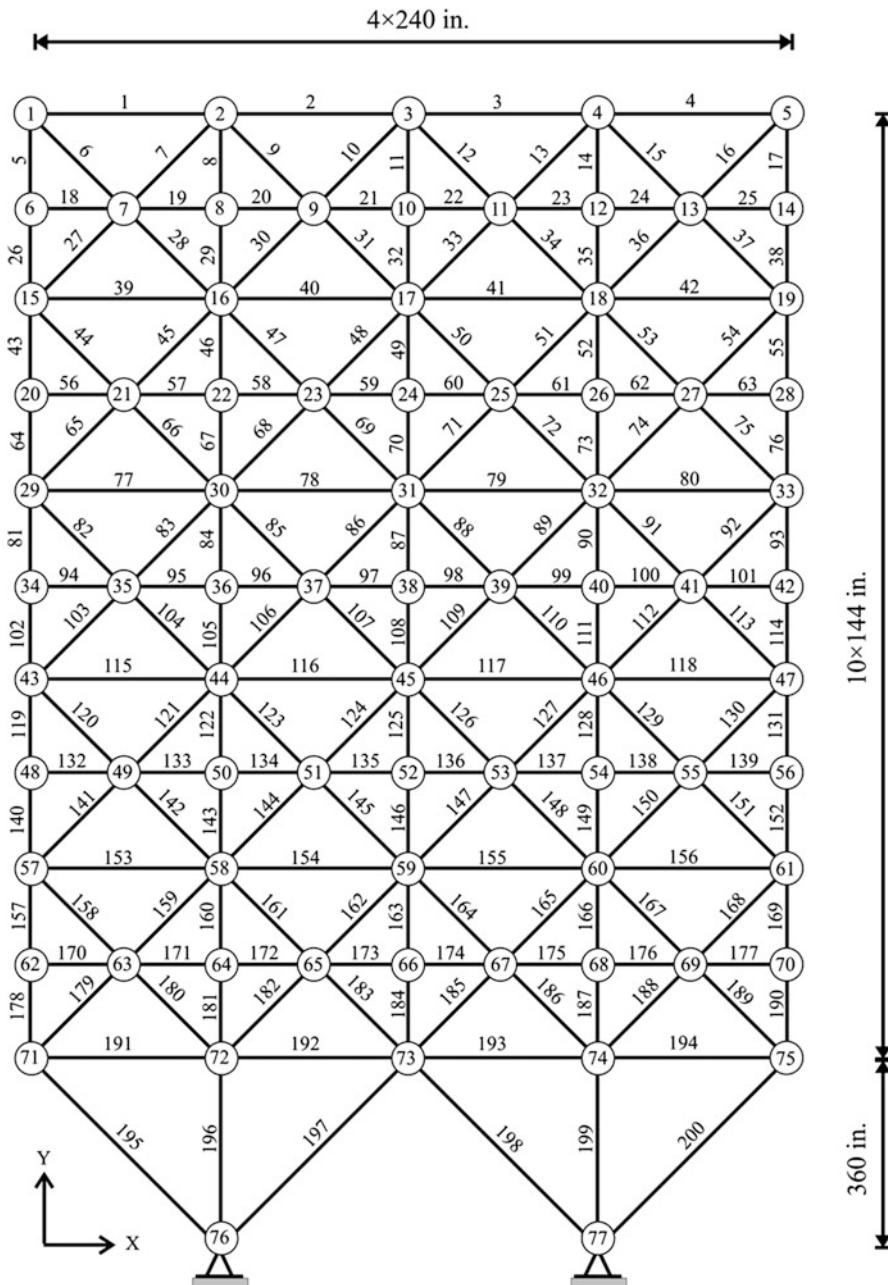
#### 12.6.4 Design Examples

In order to test the performance of the CSP method, two large-scale test problems are adapted from the original paper of Kaveh et al. [37] which previously treated by other investigators: the weight minimization of a 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 2500, 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<sup>3</sup> (7933.410 kg/m<sup>3</sup>) and 30 GPa (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 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, 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<sup>3</sup> (2767.990 kg/m<sup>3</sup>) and the modulus of elasticity is 10 GPa (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<sup>2</sup> (from 0.6452 to 1290.32 cm<sup>2</sup>). Loading conditions are presented in Table 12.4.

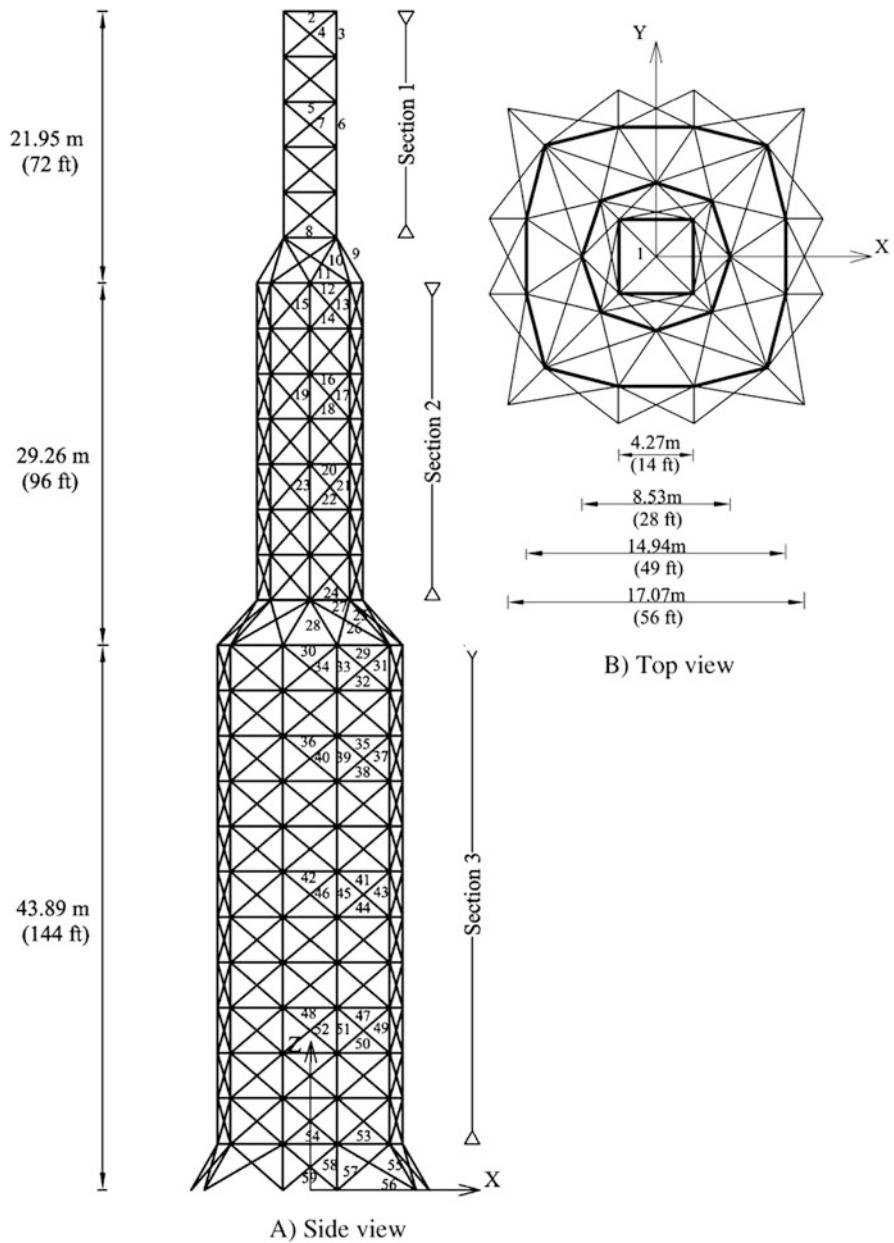


**Fig. 12.5** Schematic of a 200-bar truss structure

**Table 12.3** Comparison of optimized designs for the 200-bar truss

Element group	CMLPSA	GA	SAHS	CSP
$A_{1\sim 4}$	0.1468	0.3469	0.154	0.1480
$A_{5, 8, 11, 14, 17}$	0.9400	1.0810	0.941	0.9460
$A_{19\sim 24}$	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
$A_{39\sim 42}$	0.1000	0.1000	0.100	0.1010
$A_{43, 46, 49, 52, 55}$	3.1042	3.5650	3.108	3.1072
$A_{57\sim 62}$	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
$A_{77\sim 80}$	0.1912	0.4400	0.191	0.1944
$A_{81, 84, 87, 90, 93}$	5.4284	5.9520	5.428	5.4299
$A_{95\sim 100}$	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
$A_{115\sim 118}$	0.1327	0.3470	0.151	0.1349
$A_{119, 122, 125, 128, 131}$	7.9717	8.5250	7.973	7.9747
$A_{133\sim 138}$	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
$A_{135\sim 156}$	0.4196	1.7639	0.422	0.4225
$A_{157, 160, 163, 166, 169}$	10.8636	13.3006	10.892	10.8685
$A_{171\sim 176}$	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
$A_{191\sim 194}$	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
<i>Best weight (lb)</i>	25,445.6	28,533.1	25,491.9	25,467.9
<i>Average weight (lb)</i>	N/A	N/A	25,610.2	25,547.6
<i>Std dev (lb)</i>	N/A	N/A	141.85	135.09
<i>No. of analyses</i>	9650	51,360	19,670	31,700

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



**Fig. 12.6** Schematic of a 26-story space tower: (a) side view, (b) top view

**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)

**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

than GA, PSO, BB–BC, and HBB–BC, respectively. Table 12.6 provides the statistic information for this example [37].

These results clearly demonstrate 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 Concluding Remarks

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 stand-alone algorithms but also to be combined with other optimization methods. Even the standard metaheuristics have been successfully implemented in various applications; however, many modifications 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. The first 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 chaotic swarming of particles (CSP). This method is a kind of multiphase 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.

Though we have already seen some examples of successful combinations of chaos and metaheuristic algorithms, there still remain many open problems due to the existence of many inherent uncertain factors.

## 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 Circuits 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. In: Proceedings of the 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, USA
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. *Phys D* 59:365–377
14. Zaslavskii GM (1987) The simplest case of a strange attractor. *Phys Lett A* 69(3):145–147
15. Glover F, Kochenberger GA (2003) Handbook of metaheuristic. Kluwer Academic Publishers, Boston, MA
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 GmH, D-6940, Weinheim
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 Comput Integr 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-Ikhichi 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

## Enhanced Colliding Bodies Optimization

### 13.1 Introduction

Colliding bodies optimization (CBO) was employed for size optimization of skeletal structures in Chap. 7. In this chapter, the enhanced colliding bodies optimization (ECBO) is presented that utilizes memory to save some historically best solution and uses a random procedure to avoid local optima which is also applied to skeletal structures [1, 2]. The capability of the CBO and ECBO is compared through three trusses and two frame structures. The design constraints of steel frames are imposed according to the provisions of LRFD–AISC.

Colliding bodies optimization (CBO) is a new metaheuristic search algorithm that is developed by Kaveh and Mahdavi [3]. CBO is based on the governing laws of one-dimensional collision between two bodies from the physics that one object collides with other objects and they move toward a minimum energy level. The CBO is simple in concept, depends on no internal parameters, and does not use memory for saving the best-so-far solutions.

The remainder of this chapter is organized as follows: In Sect. 13.2, the structural mathematical formulations of the structural optimization problems are presented, and a brief explanation of the LRFD–AISC is provided. After an explanation of the CBO, the ECBO algorithm is presented in Sect. 13.3. Section 13.4 includes the mathematical optimization problems, and Sect. 13.5 contains five standard benchmark examples. The last section concludes the chapter.

### 13.2 Structural Optimization

In this study, the objective is to minimize the weight of the structure while satisfying some constraints on strength and displacement. The mathematical formulation of these problems can be expressed as follows:

$$\begin{aligned}
 & \text{Find} && \{X\} = [x_1, x_2, \dots, x_{ng}] \\
 & \text{to minimize} && W(\{X\}) = \sum_{i=1}^{nm} \rho_i x_i L_i \\
 & \text{subjected to :} && \begin{cases} g_j(\{X\}) \leq 0, j = 1, 2, \dots, nc \\ x_{i \min} \leq x_i \leq x_{i \max} \end{cases}
 \end{aligned} \tag{13.1}$$

where  $\{X\}$  is the vector containing the design variables;  $ng$  is the number of design variables;  $W(\{X\})$  presents the weight of the structure;  $nm$  is the number of elements of the structure;  $\rho_i$  and  $L_i$  denote the material density and the length of the  $i$ th member, respectively;  $x_{i \min}$  and  $x_{i \max}$  are the lower and upper bounds of the design variable  $x_i$ , respectively;  $g_j(\{X\})$  denotes design constraints; and  $nc$  is the number of the constraints. To handle the constraints, the well-known penalty approach is employed.

Strength and displacement constraints for steel frames are imposed according to the provisions of LRFD-AISC specification [4]. These constraints are briefly explained in the following:

(a) Maximum lateral displacement

$$\frac{\Delta_T}{H} - R \leq 0 \tag{13.2}$$

where  $\Delta_T$  is the maximum lateral displacement;  $H$  is the height of the frame structure; and  $R$  is the maximum drift index which is equal to 1/300.

(b) The inter-story displacements

$$\frac{d_i}{h_i} - R_I \leq 0, \quad i = 1, 2, \dots, ns \tag{13.3}$$

where  $d_i$  is the inter-story drift;  $h_i$  is the story height of the  $i$ th floor;  $ns$  is the total number of stories; and  $R_I$  presents the inter-story drift index (1/300).

(c) Strength constraints

$$\begin{cases} \frac{Pu}{2\varphi_c P_n} + \frac{M_u}{\varphi_b M_n} - 1 \leq 0, & \text{for } \frac{Pu}{\varphi_c P_n} < 0.2 \\ \frac{Pu}{\varphi_c P_n} + \frac{8M_u}{9\varphi_b M_n} - 1 \leq 0, & \text{for } \frac{Pu}{\varphi_c P_n} \geq 0.2 \end{cases} \tag{13.4}$$

where  $P_u$  is the required strength (tension or compression);  $P_n$  is the nominal axial strength (tension or compression);  $\varphi_c$  is the resistance factor ( $\varphi_c = 0.9$  for tension,  $\varphi_c = 0.85$  for compression);  $M_u$  is the required flexural strengths;  $M_n$  is the nominal flexural strengths; and  $\varphi_b$  denotes the flexural resistance reduction factor ( $\varphi_b = 0.90$ ). The nominal tensile strength for yielding in the gross section is calculated by

$$P_n = A_g \cdot F_y \quad (13.5)$$

And the nominal compressive strength of a member is computed as

$$P_n = A_g \cdot F_{cr} \quad (13.6)$$

$$\begin{cases} F_{cr} = (0.658 \lambda_c^2) F_y, & \text{for } \lambda_c \leq 1.5 \\ F_{cr} = \left( \frac{0.877}{\lambda_c^2} \right) F_y, & \text{for } \lambda_c > 1.5 \end{cases} \quad (13.7)$$

$$\lambda_c = \frac{kl}{r\pi} \sqrt{\frac{F_y}{E}} \quad (13.8)$$

where  $A_g$  is the cross-sectional area of a member and  $k$  is the effective length factor that is calculated by [5]:

$$k = \sqrt{\frac{1.6G_A G_B + 4.0(G_A + G_B) + 7.5}{G_A + G_B + 7.5}} \quad (13.9)$$

where  $G_A$  and  $G_B$  are stiffness ratios of columns and girders at two end joints, A and B, of the column section being considered, respectively.

In this chapter, the penalty function method is utilized as a constraint handling approach so the constrained objective function is expressed as follows:

$$Mer(\{X\}) = \left( 1 + \epsilon_1 \cdot \sum_{i=1}^{nc} \max[0, g_i(\{X\})] \right)^{\epsilon_2} \times W(\{X\}) \quad (13.10)$$

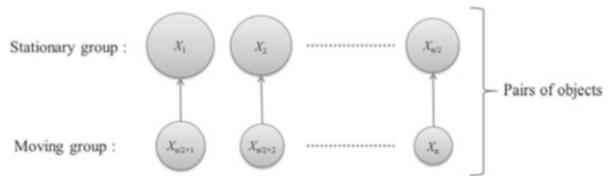
where  $\{X\}$  is a solution vector;  $Mer(\{X\})$  is the merit function;  $g_i(\{X\})$  denotes design constraints ( $g_i(\{X\}) \leq 0$ );  $nc$  is the number of the constraints; and  $W(\{X\})$  is weight of the structure. For two problems,  $\epsilon_1$  is set to unity, and  $\epsilon_2$  is taken as 1.5 at the start and linearly increases to 3.

### 13.3 An Enhanced Colliding Bodies Optimization (ECBO)

#### 13.3.1 A Brief Explanation of the CBO Algorithm

The colliding bodies optimization (CBO) is a new metaheuristic algorithm introduced by Kaveh and Mahdavi [3] and contains a number of colliding bodies (CBs) where each one collides with other objects to explore the search space. Each CB,  $X_i$ , has a specified mass defined as:

**Fig. 13.1** The pairs of CBs for collision



$$m_k = \frac{\frac{1}{fit(k)}}{\sum_{i=1}^n \frac{1}{fit(i)}}, \quad k = 1, 2, \dots, n \quad (13.11)$$

where  $fit(i)$  represents the objective function value of the  $i$ th CB and  $n$  is the number of colliding bodies.

After sorting colliding bodies according to their objective function values in an increasing order, two equal groups are created: (i) stationary group and (ii) moving group (Fig. 13.1). Moving objects collide to stationary objects to improve their positions and push stationary objects toward better positions.

The velocities of the stationary and moving bodies before collision ( $v_i$ ) are calculated by

$$v_i = 0, \quad i = 1, 2, \dots, \frac{n}{2} \quad (13.12)$$

$$v_i = x_{i-\frac{n}{2}} - x_i, \quad i = \frac{n}{2} + 1, \frac{n}{2} + 2, \dots, n \quad (13.13)$$

The velocities of stationary and moving CBs after the collision ( $v'_i$ ) are evaluated by:

$$v'_i = \frac{\left(m_{i+\frac{n}{2}} + \epsilon m_{i-\frac{n}{2}}\right)v_{i+\frac{n}{2}}}{m_i + m_{i+\frac{n}{2}}} \quad i = 1, 2, \dots, \frac{n}{2} \quad (13.14)$$

$$v'_i = \frac{\left(m_i - \epsilon m_{i-\frac{n}{2}}\right)v_i}{m_i + m_{i-\frac{n}{2}}} \quad i = \frac{n}{2} + 1, \frac{n}{2} + 2, \dots, n \quad (13.15)$$

$$\epsilon = 1 - \frac{iter}{iter_{max}} \quad (13.16)$$

where  $\epsilon$  is the coefficient of restitution (COR) and  $iter$  and  $iter_{max}$  are the current iteration number and the total number of iteration for optimization process, respectively.

New positions of each group are stated by the following formulas:

$$x_i^{new} = x_i + rand \circ v'_i, \quad i = 1, 2, \dots, \frac{n}{2} \quad (13.17)$$

$$x_i^{new} = x_{i-\frac{n}{2}} + rand \circ v'_i, \quad i = \frac{n}{2} + 1, \frac{n}{2} + 2, \dots, n \quad (13.18)$$

where  $x_i^{new}$ ,  $x_i$ , and  $v'_i$  are the new position, previous position, and the velocity after the collision of the  $i$ th CB, respectively.  $rand$  is a random vector uniformly distributed in the range of  $[-1, 1]$ , and the sign “ $\circ$ ” denotes an element-by-element multiplication.

### 13.3.2 The ECBO Algorithm

In the enhanced colliding bodies optimization (ECBO), a memory that saves a number of historically best CBs is utilized to improve the performance of the CBO and reduce the computational cost. Furthermore, ECBO changes some components of CBs randomly to prevent premature convergence. In order to introduce the ECBO, the following steps are developed and its pseudo code is provided in Fig. 13.2.

#### Level 1: Initialization

**Step 1:** The initial locations of CBs are created randomly in an  $m$ -dimensional search space.

$$x_i^0 = x_{\min} + random \circ (x_{\max} - x_{\min}), \quad i = 1, 2, \dots, n \quad (13.19)$$

where  $x_i^0$  is the initial solution vector of the  $i$ th CB.  $x_{\min}$  and  $x_{\max}$  are the minimum and the maximum allowable variables vectors; and  $random$  is a random vector with each component being in the interval  $[0, 1]$ .

#### Level 2: Search

**Step 1:** The value of the mass for each CB is calculated in Eq. (13.11).

**Step 2:** Colliding memory (CM) is considered to save some historically best CB vectors and their related mass and objective function values. The size of the CM is taken as  $n/10$  in this study. At each iteration, solution vectors that are saved in the CM are added to the population and the same number of the current worst CBs is deleted.

**Step 3:** CBs are sorted according to their objective function values in an increasing order. To select the pairs of CBs for collision, they are divided into two equal groups: (i) stationary group and (ii) moving group (Fig. 13.1).

**Step 4:** The velocities of stationary and moving bodies before collision are evaluated in Eqs. (13.12) and (13.13), respectively.

**Step 5:** The velocities of stationary and moving bodies after collision are calculated in Eqs. (13.14) and (13.15), respectively.

---

```

procedure Enhanced Colliding Bodies Optimization (ECBO)
    for all CBS      /* 30 and 40 CBS are considered for truss and frame problems, respectively */
        Initial location is created randomly
        The values of objective function and mass are evaluated
    end for
    while maximum iterations is not fulfilled      /* Maximum evaluation number is set to 20,000 analyses */
        Colliding memory (CM) is updated      /* The size of the CM is taken as n/10 */
        The population is updated
        CBS are sorted according to their objective function values in an increasing order
        for all CBS
            The velocity before collision is evaluated by Eq. (13.12) or (13.13)
            The velocity after collision is evaluated by Eq. (13.14) or (13.15)
            New location is updated by Eq. (13.17) or (13.18)
            If  $rn_i < pro$       /*  $pro$  is set to 0.25 */
                 $k \leftarrow \text{random\_int}(1,m)$ 
                 $k$ th dimension is regenerated randomly in its allowable range
            end if
        end for
    end while

end procedure

```

---

**Fig. 13.2** Pseudo code of the enhanced colliding bodies optimization

**Step 6:** The new location of each CB is evaluated in Eqs. (13.17) or (13.18).

**Step 7:** A parameter like  $pro$  within  $(0, 1)$  is introduced and specified whether a component of each CB must be changed or not. For each CB  $pro$  is compared with  $rn_i$  ( $i = 1, 2, \dots, n$ ) which is a random number uniformly distributed within  $(0, 1)$ . If  $rn_i < pro$ , one dimension of  $i$ th CB is selected randomly and its value regenerated by:

$$x_{ij} = x_{j,\min} + \text{random.} (x_{j,\max} - x_{j,\min}) \quad (13.20)$$

where  $x_{ij}$  is the  $j$ th variable of the  $i$ th CB and  $x_{j,\min}$  and  $x_{j,\max}$  are the lower and upper bounds of the  $j$ th variable. In this chapter, the value of  $pro$  is set to 0.25.

Here, the value of  $pro$  is set to 0.25.

**Level 3:** Terminal condition check

**Step 1:** After the predefined maximum evaluation number, the optimization process is terminated.

## 13.4 Mathematical Optimization Problems

To verify the efficiency of the new algorithm, some numerical examples are considered from literature. Six mathematical examples are investigated in this section. In Sect. 13.5, ECBO is employed to find an optimum design of truss structures with continuous and discrete search domains. To reduce statistical errors, each test is repeated 30 and 20 times independently for mathematical functions and truss problems, respectively. The algorithms are coded in MATLAB software, and the structures are analyzed using the direct stiffness method.

Six benchmark functions are optimized using CBO and ECBO in this section. Their results are compared to some variations of particle swarm optimization (PSO) algorithm which were reported in Ref. [6] to validate the performance of the proposed method. These algorithms include GPSO with a fixed inertia weight  $\omega = 0.4$  [7], RPSO, CLPSO [8], and ALC-PSO [6].

The description of these functions is provided in Table 13.1. The first two functions ( $f_1-f_2$ ) are unimodal functions and  $f_3-f_6$  are multimodal functions. The number of dimensions of each test function is set to 30, and to make the comparisons meaningful, the size of population is taken as 20 in algorithms. For all examples, the optimization process is terminated after  $2 \times 10^5$  function evaluations. In this table,  $\epsilon$  is the predefined accuracy from optimum. Results of the present study and some variations of PSO are provided in Table 13.2. The parameter “success rate” stands for the percentage of the successful runs that acceptable solutions are found. Also, the mean, best, standard deviation and numbers of function evaluations (FEs) for 30 independent runs are reported in this table.

On unimodal functions, convergence speed and accuracy are more important parameters for comparing algorithms because finding global optimum is relatively easy. ECBO converges faster than other methods for the second test function, and GPSO has the best FEs in the first test function. The global minimum is more difficult to locate on multimodal functions, and therefore reliability, accuracy, and convergence speed are studied for comparison. Results show that ECBO has better performance on  $f_3$  and  $f_4$ . To sum up, comparison of the results demonstrates that ECBO has a good performance in both unimodal and multimodal functions.

## 13.5 Design Examples

Four benchmark structural examples are considered to verify the efficiency of the proposed algorithms. These examples contain:

- The 25-bar space truss
- The 72-bar space truss
- The 582-bar space truss
- The 3-bay 15-story frame
- The 3-bay 24-story frame

**Table 13.1** Benchmark mathematical functions

Function		n	Domain	Optimum	$\epsilon$	Name
Unimodal	$f_1(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i)^2 + (x_i - 1)^2]$	30	$[-10, 10]^30$	0	100	Rosenbrock
	$f_2(x) = \sum_{i=1}^n [(x_i + 0.5)]^2$	30	$[-100, 100]^30$	0	0	Step
Multimodal	$f_3(x) = \sum_{i=1}^n [x_i^2 - 10 \cos(2\pi x_i) + 10]$	30	$[-5, 12, 5, 12]^30$	0	10	Rastrigin
	$f_4(x) = \sum_{i=1}^n [y_i^2 - 10 \cos(2\pi y_i) + 10],$ $y_i = \begin{cases} x_i, &  x_i  < 0.5 \\ \frac{\text{round}(2x_i)}{2}, &  x_i  \geq 0.5 \end{cases}$	30	$[-5, 12, 5, 12]^30$	0	10	Noncontinuous Rastrigin
	$f_5(x) = -20 \exp\left(-0.2 \sqrt{\frac{1}{n} \sum_{i=1}^n x_i^2}\right) - \exp\left(\frac{1}{n} \sum_{i=1}^n \cos(2\pi x_i) + 20 + e\right)$	30	$[-32, 32]^30$	0	0.01	Ackley
	$f_6(x) = \%_{400} \sum_{i=1}^n x_i^2 + \prod_{i=1}^n \cos(x_i/\sqrt{i}) + 1$	30	$[-600, 600]^30$	0	0.01	Griewank

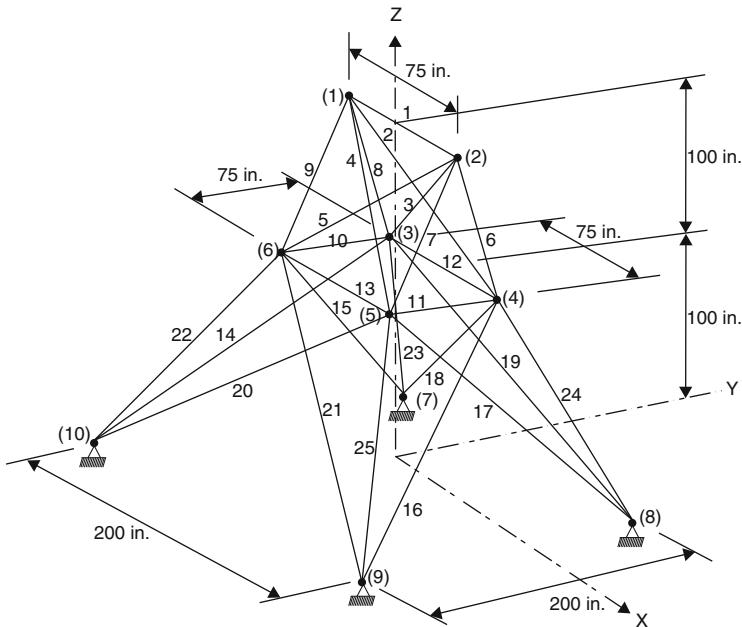
**Table 13.2** Performance comparison for the test functions

		Present work [2]					
Function		GPSO ( $\omega = 0.4$ ) [6]	RPSO [6]	CLPSO [6]	ALC-PSO [6]	CBO	ECBO
$f_1$	Success rate	100	100	100	100	86.7	100
	Mean	11.657	20.57	16.957	7.613	57.295	20.14
	Best	$2.362 \times 10^{-4}$	$2.545 \times 10^{-1}$	$9.698 \times 10^{-1}$	$3.729 \times 10^{-7}$	$2.415 \times 10^{-4}$	$1.46 \times 10^{-2}$
	Std dev	14.999	12.459	12.791	6.658	49.24	30.13
$f_2$	FEs	5348	100,433	59,985	6406	26,435	13,329
	Success rate	100	100	100	100	90	100
	Mean	$2.667 \times 10^{-1}$	0	0	0	$1.667 \times 10^{-1}$	0
	Best	0	0	0	0	0	0
$f_3$	Std dev	$4.498 \times 10^{-1}$	0	0	0	$5.921 \times 10^{-1}$	0
	FEs	29,085	90,102	28,038	13,176	23,775	10,637
	Success rate	0	0	100	100	0	100
	Mean	25.24	38.76	$2.440 \times 10^{-14}$	$2.528 \times 10^{-14}$	85.002	0
$f_4$	Best	14.92	12.93	0	$7.105 \times 10^{-15}$	32.834	0
	Std dev	5.209	8.633	$5.979 \times 10^{-14}$	$1.376 \times 10^{-14}$	21.782	0
	FEs	N/A	N/A	103,459	74,206	N/A	15,571
	Success rate	30	0	100	100	0	100
	Mean	10	33.60	$1.33 \times 10^{-1}$	$1.251 \times 10^{-11}$	105.767	0
	Best	1	11	0	0	63	0
	Std dev	15.411	10.884	$3.461 \times 10^{-1}$	$6.752 \times 10^{-11}$	24.474	0
	FEs	64,994	N/A	113,585	58,900	N/A	12,007

(continued)

Table 13.2 (continued)

Function		Present work [2]			
		GPso ( $\omega = 0.4$ ) [6]	RPSO [6]	CLPSO [6]	ALC-PSO [6]
$f_5$	Success rate	23.3	100	100	0
	Mean	$1.101 \times 10^{-14}$	$2.664 \times 10^{-14}$	$2.487 \times 10^{-14}$	$1.148 \times 10^{-14}$
	Best	$7.694 \times 10^{-15}$	$1.125 \times 10^{-14}$	$1.835 \times 10^{-14}$	$7.694 \times 10^{-15}$
	Std dev	$2.273 \times 10^{-15}$	$5.445 \times 10^{-14}$	$4.181 \times 10^{-15}$	$2.941 \times 10^{-15}$
	FEs	11,171	125,779	66,771	58,900
	Success rate	56.7	73.3	100	60
$f_6$	Mean	$1.646 \times 10^{-2}$	$8.169 \times 10^{-3}$	$2.007 \times 10^{-14}$	$1.221 \times 10^{-2}$
	Best	0	0	0	$1.095 \times 10^{-1}$
	Std dev	$1.690 \times 10^{-2}$	$1.780 \times 10^{-2}$	$8.669 \times 10^{-14}$	$1.166 \times 10^{-14}$
	FEs	8265	122,650	66,649	10,161
					16,523
					22,774



**Fig. 13.3** Schematic of the 25-bar space truss

In CBO and ECBO, the population of 30 and 40 CBs is utilized for truss and frame problems, respectively. Also, the predefined maximum evaluation number is considered as 20,000 analyses for all examples. Because of the stochastic nature of the algorithms, each example has been solved 20 times independently. In all problems, CBs are allowed to select discrete values from the permissible list of cross sections (real numbers are rounded to the nearest integer in the each iteration). The algorithms are coded in MATLAB, and the structures are analyzed using the direct stiffness method.

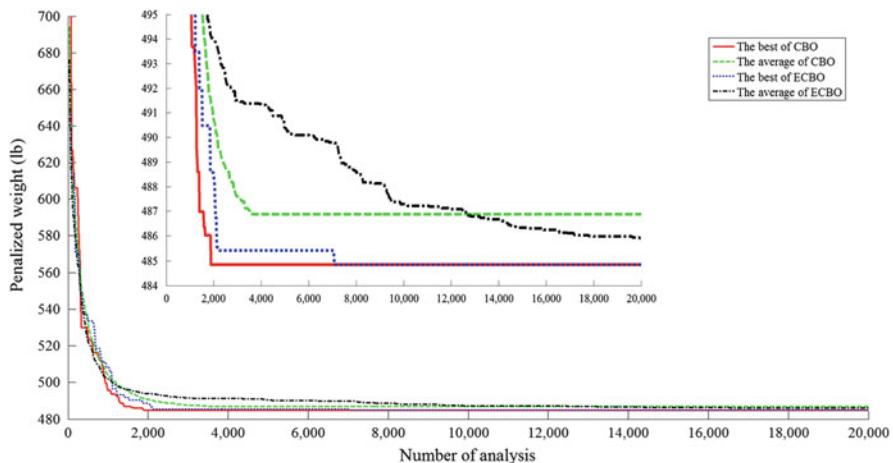
### 13.5.1 A 25-Bar Space Truss

The 25-bar transmission tower is a well-known benchmark problem, and Fig. 13.3 shows the topology and nodal numbering of this truss. Twenty-five members are categorized into eight groups, as follows: (1) A<sub>1</sub>, (2) A<sub>2</sub>–A<sub>5</sub>, (3) A<sub>6</sub>–A<sub>9</sub>, (4) A<sub>10</sub>–A<sub>11</sub>, (5) A<sub>12</sub>–A<sub>13</sub>, (6) A<sub>14</sub>–A<sub>17</sub>, (7) A<sub>18</sub>–A<sub>21</sub>, and (8) A<sub>22</sub>–A<sub>25</sub>. The material density is 0.1 lb/in<sup>3</sup> (2767.99 kg/m<sup>3</sup>), and the modulus of elasticity is 10<sup>7</sup> psi (68,950 MPa) for all elements. A single load case {(kips) (kN)} is applied to the structure as follows: {(0, -10, -10) (0, -44.5, -44.5)} acting on node 1, {(1, -10, -10) (4.45, -44.5, -44.5)} acting on node 2, {(0.6, 0, 0) (2.67, 0, 0)} acting on node 3, and {(0.5, 0, 0) (2.225, 0, 0)} acting on node 4. The allowable stresses are  $\pm 40$  ksi (275.80 MPa) for each member, and maximum displacement limitations of

**Table 13.3** Optimal design obtained for the 25-bar space truss

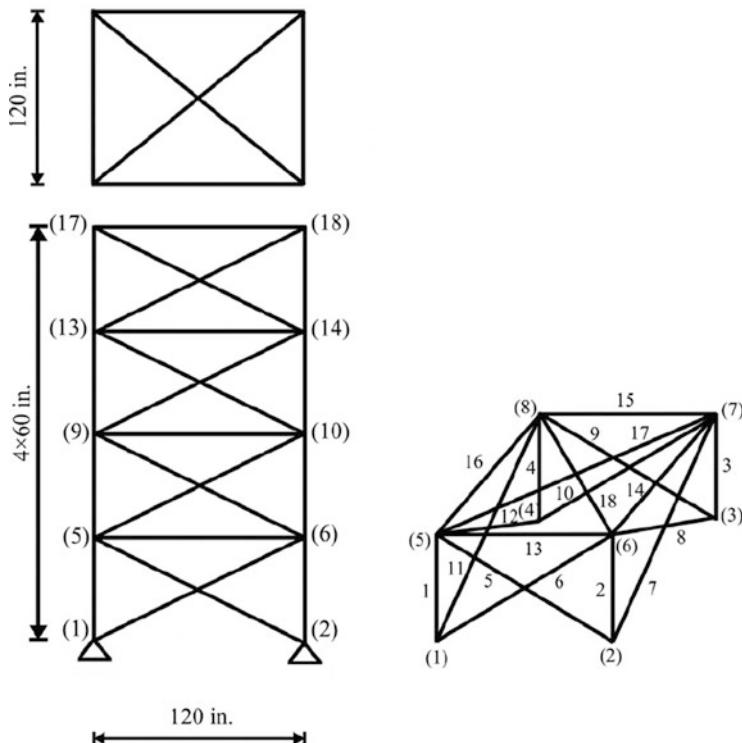
Element group	Members	Areas (in <sup>2</sup> )					
		ACO [9]	GA [10]	BB–BC Phase 2 [10]	ABC [11]	IRO [12]	Present work [1] CBO
1	A <sub>1</sub>	0.1	0.10	0.10	0.1	0.1	0.1
2	A <sub>2</sub> –A <sub>5</sub>	0.3	0.50	0.30	0.3	0.3	0.3
3	A <sub>6</sub> –A <sub>9</sub>	3.4	3.40	3.40	3.4	3.4	3.4
4	A <sub>10</sub> –A <sub>11</sub>	0.1	0.10	0.10	0.1	0.1	0.1
5	A <sub>12</sub> –A <sub>13</sub>	2.1	1.90	2.10	2.1	2.1	2.1
6	A <sub>14</sub> –A <sub>17</sub>	1	0.90	1.00	1	1	1
7	A <sub>18</sub> –A <sub>21</sub>	0.5	0.50	0.50	0.5	0.5	0.5
8	A <sub>22</sub> –A <sub>25</sub>	3.4	3.40	3.40	3.4	3.4	3.4
Weight (lb)		484.85	485.05	484.85	484.85	484.85	484.85
Mean weight (lb)		48,646	N/A	485.10	485.05	486.87	485.89
Constraint tolerance (%)	None	None	None	None	None	None	None

1 in<sup>2</sup> = 6.4516 cm<sup>2</sup>

**Fig. 13.4** Convergence curves for the 25-bar space truss

$\pm 0.35$  in ( $\pm 8.89$  mm) are imposed on every node in every direction. The range of discrete cross-sectional areas is from 0.1 to 3.4 in<sup>2</sup> (0.6452–21.94 cm<sup>2</sup>) with 0.1 in<sup>2</sup> (0.6452 cm<sup>2</sup>) increment.

Optimal structures found by ACO [9], GA [10], BB–BC [10], ABC [11], IRO [12], CBO, and ECBO are summarized in Table 13.3. The best designs of all methods are identical and there is no penalty for these results. The average weight of independent runs obtained by IRO is better than the other methods (484.90 lb); however, this value is approximately equal to other ones (the mean weights of ACO, BB–BC, ABC, CBO, and ECBO are 486.46 lb, 485.10 lb, 485.05 lb,



**Fig. 13.5** Schematic of the 72-bar space truss

486.87 lb, and 485.89 lb, respectively). The ECBO performs better than the CBO in terms of average weight, but the best solution of these algorithms is found after 7050 and 2040 analyses, respectively. Figure 13.4 illustrates the convergence curves of the best and average results obtained by the proposed methods.

The algorithm is coded in MATLAB and structures are analyzed using the direct stiffness method.

### 13.5.2 A 72-Bar Space Truss

The 72-bar space truss is shown in Fig. 13.5 as the second design example. The elements are divided into sixteen groups, because of 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)  $A_{71}-A_{72}$ . The material density is  $0.1 \text{ lb/in}^3$  ( $2767.990 \text{ kg/m}^3$ ), and the modulus of elasticity is  $10^7 \text{ psi}$  ( $68,950 \text{ MPa}$ ). The structure is subjected to two load cases listed in Table 13.4. The members are

**Table 13.4** Loading conditions for the 72-bar space truss

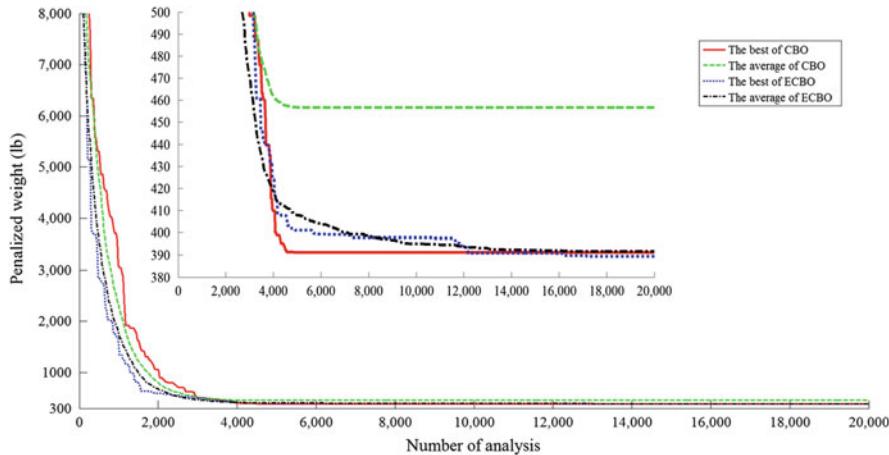
Node	Condition 1			Condition 2		
	$F_x$ kips (kN)	$F_y$ kips (kN)	$F_z$ kips (kN)	$F_x$ kips (kN)	$F_y$ kips (kN)	$F_z$ kips (kN)
17	0.0	0.0	-5.0 (-22.25)	-5.0 (-22.25)	5.0 (-22.25)	-5.0 (-22.25)
18	0.0	0.0	-5.0 (-22.25)	0.0	0.0	0.0
19	0.0	0.0	-5.0 (-22.25)	0.0	0.0	0.0
20	0.0	0.0	-5.0 (-22.25)	0.0	0.0	0.0

**Table 13.5** Optimal design obtained for the 72-bar space truss

Element group	Members	Areas ( $\text{in}^2$ )					
		HPSO [13]	HPSACO [15]	PSOPC [13]	ICA [13]	IRO [12]	Present work [1] CBO
1	A <sub>1</sub> –A <sub>4</sub>	4.970	1.800	4.490	1.99	1.99	2.13
2	A <sub>5</sub> –A <sub>12</sub>	1.228	0.442	1.457	0.442	0.563	0.563
3	A <sub>13</sub> –A <sub>16</sub>	0.111	0.141	0.111	0.111	0.111	0.111
4	A <sub>17</sub> –A <sub>18</sub>	0.111	0.111	0.111	0.141	0.111	0.111
5	A <sub>19</sub> –A <sub>22</sub>	2.880	1.228	2.620	1.228	1.228	1.228
6	A <sub>23</sub> –A <sub>30</sub>	1.457	0.563	1.130	0.602	0.563	0.442
7	A <sub>31</sub> –A <sub>34</sub>	0.141	0.111	0.196	0.111	0.111	0.141
8	A <sub>35</sub> –A <sub>36</sub>	0.111	0.111	0.111	0.141	0.111	0.111
9	A <sub>37</sub> –A <sub>40</sub>	1.563	0.563	1.266	0.563	0.563	0.442
10	A <sub>41</sub> –A <sub>48</sub>	1.228	0.563	1.457	0.563	0.442	0.563
11	A <sub>49</sub> –A <sub>52</sub>	0.111	0.111	0.111	0.111	0.111	0.111
12	A <sub>53</sub> –A <sub>54</sub>	0.196	0.250	0.111	0.111	0.111	0.111
13	A <sub>55</sub> –A <sub>58</sub>	0.391	0.196	0.442	0.196	0.196	0.196
14	A <sub>59</sub> –A <sub>66</sub>	1.457	0.563	1.457	0.563	0.563	0.563
15	A <sub>67</sub> –A <sub>70</sub>	0.766	0.442	1.228	0.307	0.391	0.391
16	A <sub>71</sub> –A <sub>72</sub>	1.563	0.563	1.457	0.602	0.563	0.563
Weight (lb)		933.09	393.380	941.82	392.84	389.33	391.23
Mean weight (lb)	N/A	N/A	N/A	N/A	408.17	456.69	391.59
Constraint tolerance (%)	None	None	None	None	None	None	None

subjected to the stress limitations of  $\pm 25$  ksi ( $\pm 172.375$  MPa). Maximum displacement of the uppermost nodes was not allowed to exceed  $\pm 0.25$  in ( $\pm 6.35$  mm), for each node, in all directions. In this case, the discrete variables are selected from 64 discrete values from 0.111 to  $33.5 \text{ in}^2$  ( $71.613$ – $21,612.860 \text{ mm}^2$ ) [13].

Table 13.5 provides a comparison between some optimal designs reported in the literature and the present works. The ECBO algorithm achieves the best weight

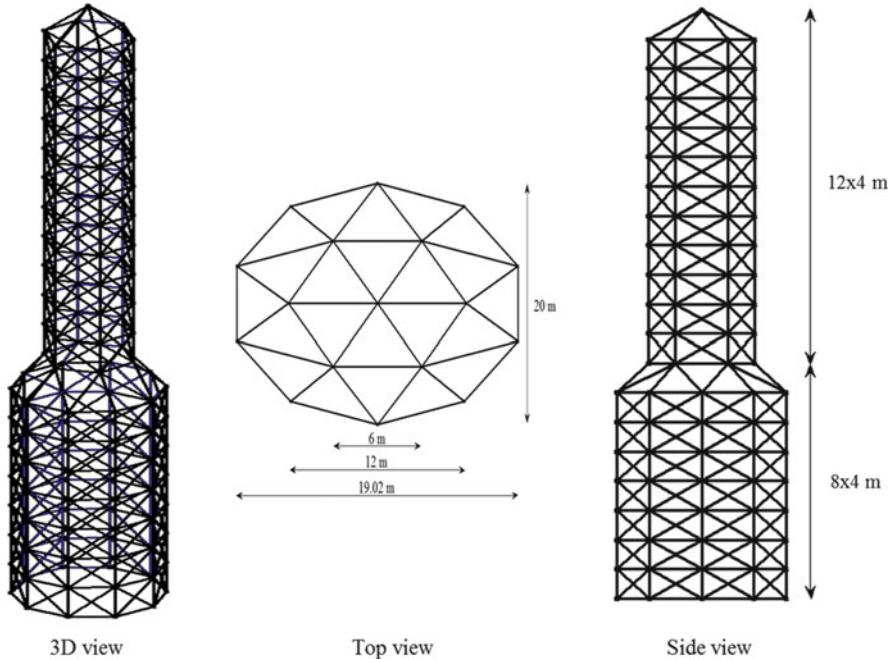


**Fig. 13.6** Convergence curves for the 72-bar space truss

(389.33 lb) although this is identical to the best design developed using IRO algorithm [12]. The best weights found by HPSO [14], HPSACO [15], PSOPC [13], ICA [13], and CBO are 933.09 lb, 393.380 lb, 941.82 lb, 392.84 lb, and 391.23 lb, respectively. All designs obtained by the algorithms are feasible. The CBO and ECBO algorithms achieve the best solutions after 4620 and 17,010 analyses, respectively. The mean weight of the solutions obtained by the ECBO is remarkably less than those of the IRO and CBO. Figure 13.6 shows the best and average of 20 runs of convergence history for the proposed algorithms.

### 13.5.3 A 582-Bar Tower Truss

Figure 13.7 shows the 582-bar tower truss as the second design example. A single load case is considered consisting of the lateral loads of 1.12 kips (5.0 kN) applied in both x and y directions and a vertical load of -6.74 kips (-30 kN) applied in the z direction at all nodes of the tower. The 582 members are categorized into 32 groups, because of symmetry. The tower is optimized for minimum volume with the cross-sectional areas of the members being the design variables. A discrete set of standard steel sections selected from W-shaped profile list based on area and radii of gyration properties is used to size the variables. The lower and upper bounds on size variables are taken as  $6.16 \text{ in}^2$  ( $39.74 \text{ cm}^2$ ) and  $215.0 \text{ in}^2$  ( $1387.09 \text{ cm}^2$ ), respectively. In this problem, CBs are allowed to select discrete values from this list (real numbers are rounded to the nearest integer). The stress and stability limitations of the members are imposed according to the provisions of ASD-AISC [16] as follows:



**Fig. 13.7** Schematic of a 582-bar tower truss

The allowable tensile stresses for tension members are calculated by:

$$\sigma_i^+ = 0.6F_y \quad (13.21)$$

where  $F_y$  stands for the yield strength.

The allowable stress limits for compression members are calculated depending on two possible failure modes of the members known as elastic and inelastic buckling. Thus:

$$\sigma_i^- = \begin{cases} \left[ \left( 1 - \frac{\lambda_i^2}{2C_c^2} \right) F_y \right] / \left[ \frac{5}{3} + \frac{3\lambda_i}{8C_c} - \frac{\lambda_i^3}{8C_c^3} \right] & \text{for } \lambda_i < C_c \\ \frac{12\pi^2 E}{23\lambda_i^2} & \text{for } \lambda_i \geq C_c \end{cases} \quad (13.22)$$

where  $E$  is the modulus of elasticity;  $\lambda_i$  is the slenderness ratio ( $\lambda_i = kl_i/r_i$ );  $C_c$  denotes the slenderness ratio dividing the elastic and inelastic buckling regions ( $C_c = \sqrt{2\pi^2 E/F_y}$ );  $k$  is the effective length factor ( $k$  is set to 1 for all truss members);  $L_i$  is the member length; and  $r_i$  is the minimum radii of gyration.

In ASD-AISC design code provisions [16], the maximum slenderness ratio is limited to 300 for tension members, and it is recommended to be 200 for compression members. Also in this example, the displacement limitations of  $\pm 3.15$  in ( $\pm 8.0$  cm) are imposed on all nodes in x, y, and z directions.

Table 13.6 presents the optimum designs obtained by DHPsaco [15], HBB-BC [17], CBO, and ECBO. It can be seen that the best answer is achieved using ECBO with 20 CBs, and this design is found after 19,700 analyses. CBO with 20 and 50 CBs and ECBO with 50 CBs required 6300, 17,700, and 19,800 analyses to converge to their best solutions, respectively.

The existing stress ratio and nodal displacements in all directions are demonstrated for the best design of ECBO with 20 CBs in Fig. 13.8. The maximum stress ratio is 99.86 %, and the maximum displacements in x and y directions are 3.1498 *in* and 2.9941 *in*, respectively.

Convergence history for each case is illustrated in Fig. 13.9. It is observed that although the numbers of structural analyses required by the ECBO for the best designs are more than those of CBO, the curves of ECBO are below that of CBO in all penalized volumes.

### 13.5.4 A 3-Bay 15-Story Frame

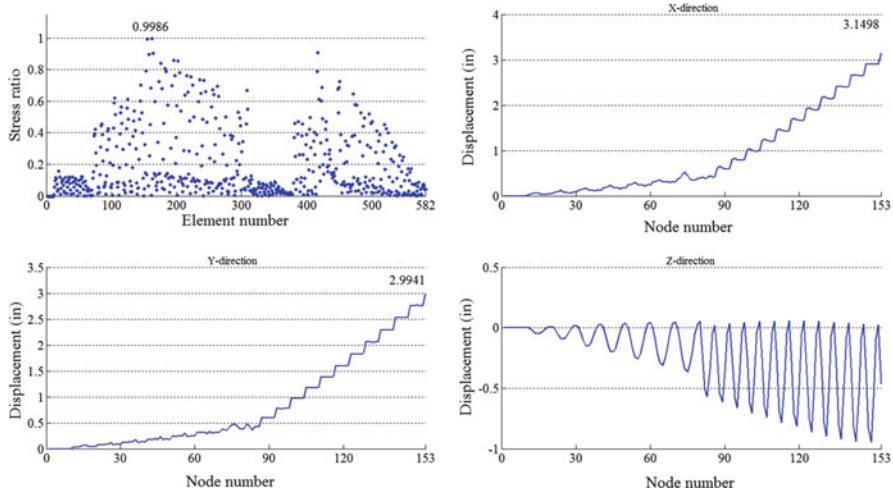
The fourth design example is a 15-story frame consisting of 64 joints and 105 members. The configuration, the applied loads, and the numbering of member groups for this problem are shown in Fig. 13.10 (Ref. [17]). The modulus of elasticity is 29,000 ksi (200 GPa), and the yield stress is 36 ksi (248.2 MPa) for all members. 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 non-braced length for each beam member is specified as one-fifth of the span length. The frame is designed following the LRFD specification and uses an inter-story drift displacement constraint [4]. Also, the sway of the top story is limited to 9.25 in (23.5 cm).

Table 13.7 shows the best solution vectors, the corresponding weights, the average weights, and the constraint violation tolerances for the present algorithms and some other metaheuristic algorithms. ECBO has obtained the lightest design compared to other methods. The best weight of the ECBO algorithm is 86,986 lb, while it is 95,850 lb for the HPSACO [18], 97,689 lb for the HBB-BC [17], 93,846 lb for the ICA [13], 92,723 lb for CSS [19], and 93,795 lb for the CBO. All optimum designs found by the algorithms satisfy the design constraints. The CBO and ECBO algorithms get the optimal solution after 9520 and 9000 analyses, respectively. Convergence history of the present algorithms for the best and average

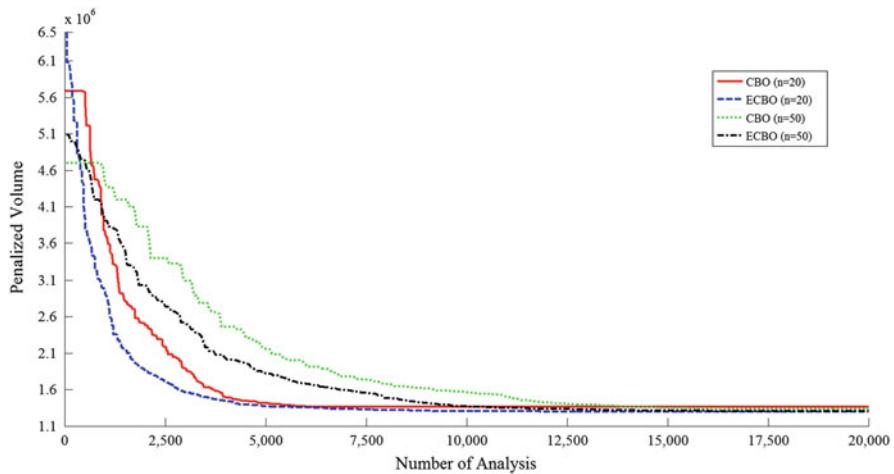
**Table 13.6** Comparison of optimized designs for the 582-bar tower truss

Element group	Optimal W-shaped sections		Present work [2]			
	DHP SACO [15]	HBB-BC [17]	n = 20		n = 50	
			CBO	ECBO	CBO	ECBO
			W8 × 24	W8 × 24	W8 × 21	W8 × 21
1			W8 × 24	W8 × 24	W8 × 21	W8 × 21
2			W12 × 72	W24 × 68	W16 × 100	W14 × 90
3			W8 × 28	W8 × 28	W8 × 31	W8 × 24
4			W12 × 58	W18 × 60	W10 × 54	W14 × 61
5			W8 × 24	W8 × 24	W12 × 26	W8 × 24
6			W8 × 24	W8 × 24	W10 × 22	W8 × 21
7			W10 × 49	W21 × 48	W10 × 60	W10 × 49
8			W8 × 24	W8 × 24	W6 × 25	W8 × 24
9			W8 × 24	W10 × 26	W8 × 21	W8 × 21
10			W12 × 40	W14 × 38	W21 × 62	W14 × 43
11			W12 × 30	W12 × 30	W8 × 24	W8 × 24
12			W12 × 72	W12 × 72	W12 × 45	W12 × 72
13			W18 × 76	W21 × 73	W12 × 72	W14 × 82
14			W10 × 49	W14 × 53	W10 × 45	W10 × 54
15			W14 × 82	W18 × 86	W18 × 76	W12 × 65
16			W8 × 31	W8 × 31	W12 × 40	W8 × 31
17			W14 × 61	W18 × 60	W12 × 58	W10 × 60
18			W8 × 24	W8 × 24	W6 × 25	W8 × 24
19			W8 × 21	W16 × 36	W10 × 22	W8 × 21
20			W12 × 40	W10 × 39	W10 × 45	W14 × 43
21			W8 × 24	W8 × 24	W12 × 26	W8 × 24
22			W14 × 22	W8 × 24	W8 × 21	W8 × 21
23			W8 × 31	W8 × 31	W14 × 30	W8 × 21
24			W8 × 28	W8 × 28	W6 × 25	W8 × 24
25			W8 × 21	W8 × 21	W8 × 21	W10 × 22
26			W8 × 21	W8 × 24	W8 × 21	W10 × 22
27			W8 × 24	W8 × 28	W6 × 25	W8 × 24
28			W8 × 28	W14 × 22	W8 × 21	W8 × 21
29			W16 × 36	W8 × 24	W8 × 21	W8 × 21
30			W8 × 24	W8 × 24	W6 × 25	W8 × 24
31			W8 × 21	W14 × 22	W8 × 24	W8 × 21
32			W8 × 24	W8 × 24	W8 × 24	W6 × 25
Volume (in <sup>3</sup> )	1,346,227	1,365,143	1,365,329	1,296,776	1,334,994	1,302,228
Mean volume (in <sup>3</sup> )	N/A	N/A	1,538,666	1,306,728	1,345,429	1,311,450

optimum designs is depicted in Fig. 13.11. It can be seen that the convergence rate of the ECBO algorithm is higher than the CBO. Figures 13.12 and 13.13 demonstrate the existing stress ratios and inter-story drifts for the best designs of CBO and ECBO.



**Fig. 13.8** The existing stress ratio and displacement in the x direction, y direction, and z direction for the best design of ECBO with 20 CBs of the 582-bar truss

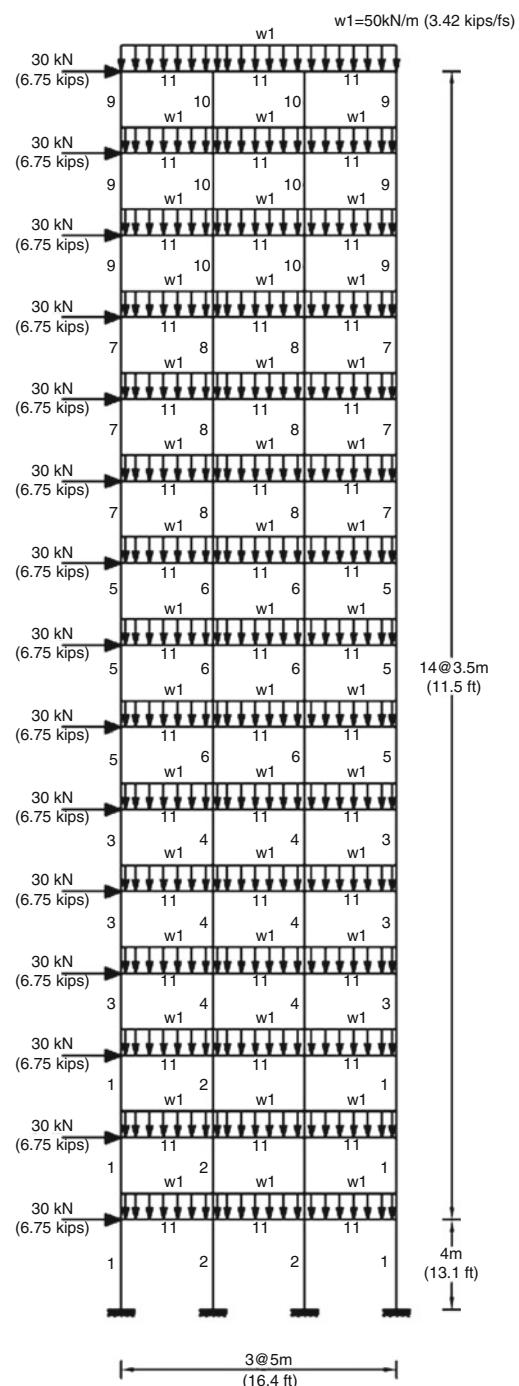


**Fig. 13.9** Convergence curves for the 582-bar tower truss

### 13.5.5 A 3-Bay 24-Story Frame

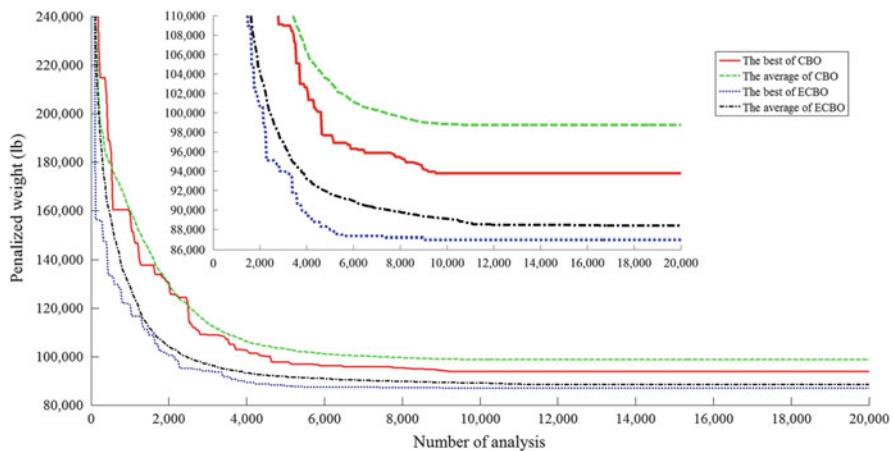
Figure 13.14 shows the configuration and applied loads of a 3-bay 24-story frame structure [18]. The frame consists of 168 members that are collected in 20 groups (16 column groups and 4 beam groups). Each of the four beam element groups is chosen from all 267 W shapes, while the 16 column element groups are limited to

**Fig. 13.10** Schematic of the 3-bay 15-story frame

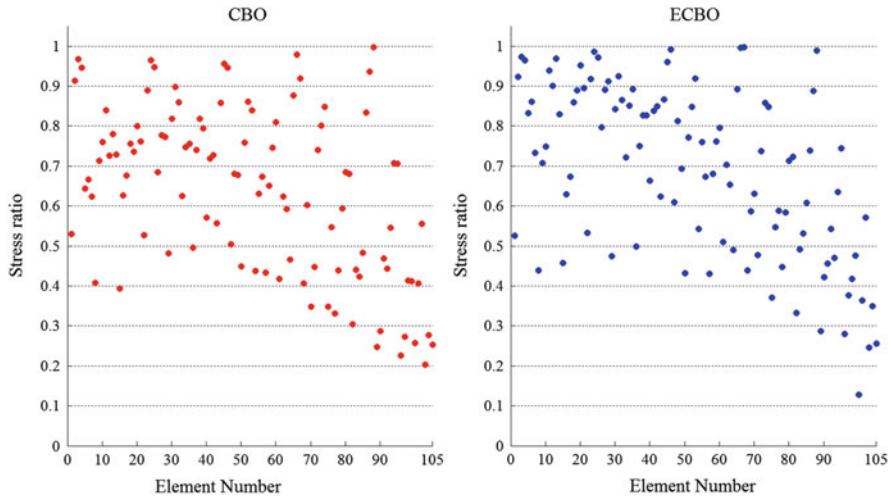


**Table 13.7** Optimal design obtained for the 3-bay 15-story frame

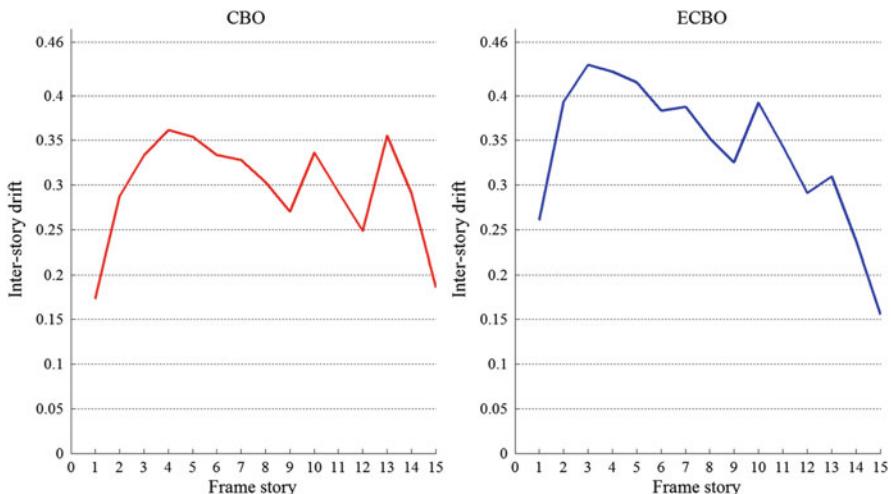
Element group	Optimal W-shaped sections					
	HPSACO [18]	HBB-BC [17]	ICA [13]	CSS [19]	Present work [1]	
					CBO	ECBO
1	W21 × 111	W24 × 117	W24 × 117	W21 × 147	W24 × 104	W14 × 99
2	W18 × 158	W21 × 132	W21 × 147	W18 × 143	W40 × 167	W27 × 161
3	W10 × 88	W12 × 95	W27 × 84	W12 × 87	W27 × 84	W27 × 84
4	W30 × 116	W18 × 119	W27 × 114	W30 × 108	W27 × 114	W24 × 104
5	W21 × 83	W21 × 93	W14 × 74	W18 × 76	W21 × 68	W14 × 61
6	W24 × 103	W18 × 97	W18 × 86	W24 × 103	W30 × 90	W30 × 90
7	W21 × 55	W18 × 76	W12 × 96	W21 × 68	W8 × 48	W14 × 48
8	W27 × 114	W18 × 65	W24 × 68	W14 × 61	W21 × 68	W14 × 61
9	W10 × 33	W18 × 60	W10 × 39	W18 × 35	W14 × 34	W14 × 30
10	W18 × 46	W10 × 39	W12 × 40	W10 × 33	W8 × 35	W12 × 40
11	W21 × 44	W21 × 48	W21 × 44	W21 × 44	W21 × 50	W21 × 44
Weight (lb)	95,850	97,689	93,846	92,723	93,795	86,986
Mean weight (lb)	N/A	N/A	N/A	N/A	98,738	88,410
Constraint tolerance (%)	None	None	None	None	None	None

**Fig. 13.11** Convergence curves for the 3-bay 15-story frame

W14 sections. The material has a modulus of elasticity equal to  $E = 29,732$  ksi (205 GPa) and a yield stress of  $f_y = 33.4$  ksi (230.3 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  $k_y = 1.0$ . All columns and beams



**Fig. 13.12** The stress ratios for the best designs of the 3-bay 15-story frame



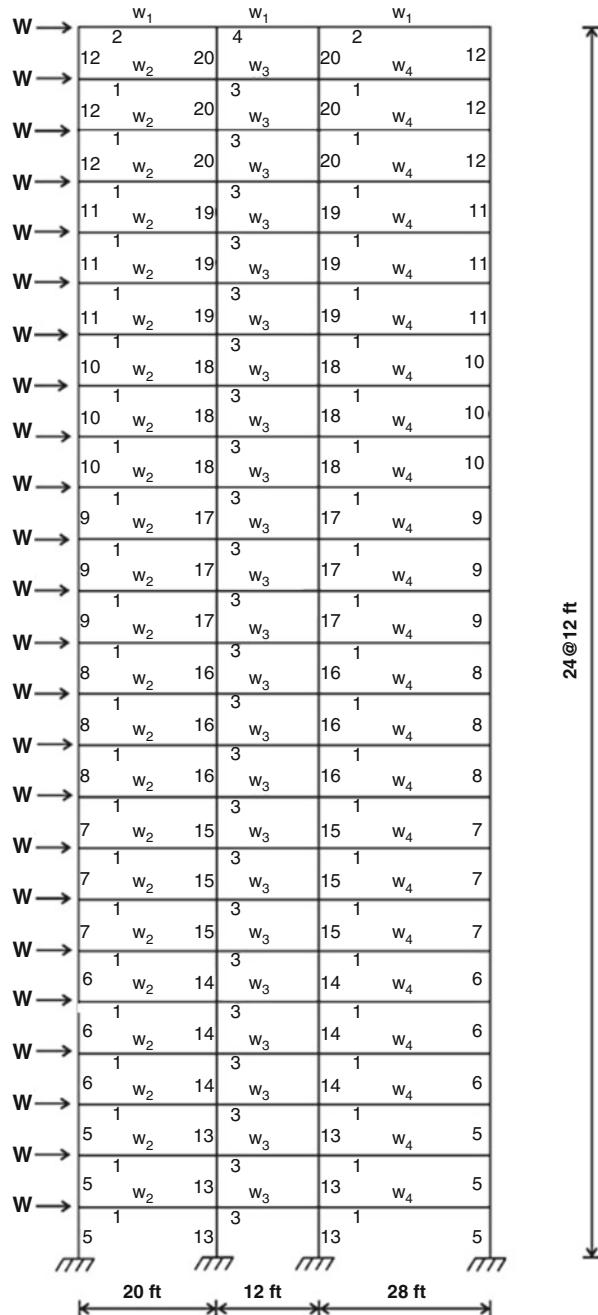
**Fig. 13.13** The inter-story drifts for the best designs of the 3-bay 15-story frame

are considered as non-braced along their lengths. The frame is designed following the LRFD specification and uses an inter-story drift displacement constraint [4].

Table 13.8 lists the optimal values of the 20 variables obtained by the ACO [21], HS [20], ICA [13], CSS [19], CBO, and ECBO. The result of ECBO method is lighter than the results found in literature (201,618 lb). The optimum designs for ACO, HS, ICA, CSS, and CBO have the weights of 220,465 lb, 214,860 lb,

**Fig. 13.14** Schematic of the 3-bay 24-story frame

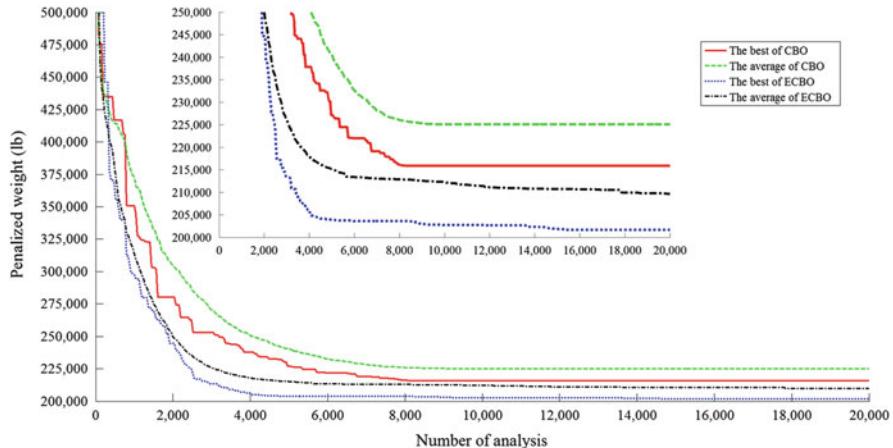
$W=5,761.85 \text{ lb}$ ,  $w_1=300 \text{ lb/ft}$ ,  $w_2=436 \text{ lb/ft}$   
 $w_3=474 \text{ lb/ft}$  and  $w_4=408 \text{ lb/ft}$



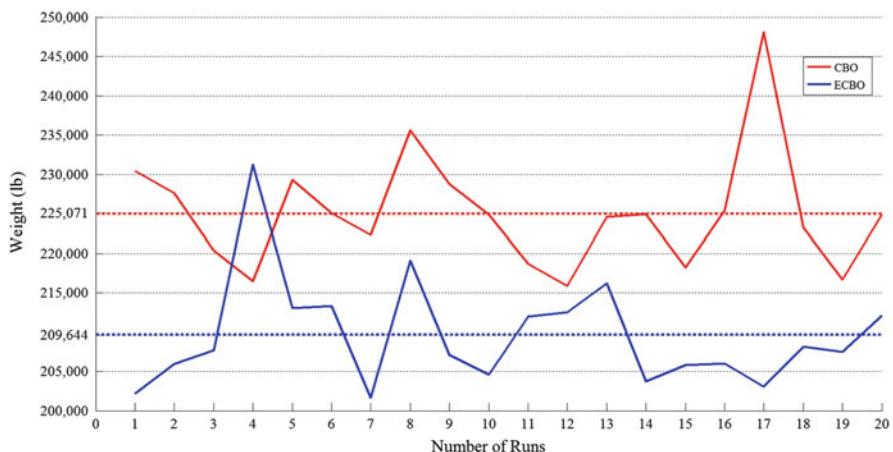
**Table 13.8** Optimal design obtained for the 3-bay 24-story frame

Element group	Optimal W-shaped sections					
	ACO [21]	HS [20]	ICA [13]	CSS [19]	Present work [1]	
					CBO	ECBO
1	W30 × 90	W30 × 90	W30 × 90	W30 × 90	W27 × 102	W30 × 90
2	W8 × 18	W10 × 22	W21 × 50	W21 × 50	W8 × 18	W6 × 15
3	W24 × 55	W18 × 40	W24 × 55	W21 × 48	W24 × 55	W24 × 55
4	W8 × 21	W12 × 16	W8 × 28	W12 × 19	W6 × 8.5	W6 × 8.5
5	W14 × 145	W14 × 176	W14 × 109	W14 × 176	W14 × 132	W14 × 145
6	W14 × 132	W14 × 176	W14 × 159	W14 × 145	W14 × 120	W14 × 132
7	W14 × 132	W14 × 132	W14 × 120	W14 × 109	W14 × 145	W14 × 99
8	W14 × 132	W14 × 109	W14 × 90	W14 × 90	W14 × 82	W14 × 90
9	W14 × 68	W14 × 82	W14 × 74	W14 × 74	W14 × 61	W14 × 74
10	W14 × 53	W14 × 74	W14 × 68	W14 × 61	W14 × 43	W14 × 38
11	W14 × 43	W14 × 34	W14 × 30	W14 × 34	W14 × 38	W14 × 38
12	W14 × 43	W14 × 22	W14 × 38	W14 × 34	W14 × 22	W14 × 22
13	W14 × 145	W14 × 145	W14 × 159	W14 × 145	W14 × 99	W14 × 99
14	W14 × 145	W14 × 132	W14 × 132	W14 × 132	W14 × 109	W14 × 99
15	W14 × 120	W14 × 109	W14 × 99	W14 × 109	W14 × 82	W14 × 99
16	W14 × 90	W14 × 82	W14 × 82	W14 × 82	W14 × 90	W14 × 82
17	W14 × 90	W14 × 61	W14 × 68	W14 × 68	W14 × 74	W14 × 68
18	W14 × 61	W14 × 48	W14 × 48	W14 × 43	W14 × 61	W14 × 61
19	W14 × 30	W14 × 30	W14 × 34	W14 × 34	W14 × 30	W14 × 30
20	W14 × 26	W14 × 22	W14 × 22	W14 × 22	W14 × 22	W14 × 22
Weight (lb)	220,465	214,860	212,640	212,364	215,874	201,618
Mean weight (lb)	229,555	222,620	N/A	215,226	225,071	209,644
Constraint tolerance (%)	None	None	None	None	None	None

212,640 lb, 212,364 lb, and 215,874 lb, respectively. There is no penalty for these results. The CBO found the optimum weight of after 8280 analyses, and ECBO obtained the best design after 15,360 analyses. It should be noted that the design found by ECBO at 8280th analysis is lighter than that found by standard CBO at the same analysis. Design history of number of analyses for the best and average optimal design with proposed methods is shown in Fig. 13.15. Figure 13.16 shows the final weights obtained by CBO and ECBO algorithms in 20 independent runs. It can be seen that the ECBO has better performance than the standard version.



**Fig. 13.15** Convergence curves for the 3-bay 24-story frame



**Fig. 13.16** The final weights of 20 independent runs for the 3-bay 24-story frame

## 13.6 Concluding Remarks

In this study, the CBO and ECBO algorithms are examined in the context of size optimization of skeletal structure designed for minimum weight. The CBO has simple structure, depends on no internal parameter, and does not use memory for saving the best-so-far solutions. In order to improve the exploration capabilities of the CBO and to prevent premature convergence, a stochastic approach is employed in ECBO that changes some components of CBs randomly. Colliding memory is also utilized to save a number of the best-so-far solutions to reduce the computational cost.

In order to indicate the similarities and differences between the characteristics of the CBO and ECBO algorithms, four benchmark structural examples comprising of trusses and frames are considered. The convergence speed of CBO is better than ECBO for truss problems; however, the reliability of search and solution accuracy of the ECBO are superior. In frame examples, the ECBO has remarkably better performance than CBO and other methods in terms of accuracy, reliability, and speed of convergence. Generally, comparison of the results with some other well-known metaheuristics shows the suitability and efficiency of the proposed algorithms.

## References

1. Kaveh A, Ilchi Ghazaan M (2015) A comparative study of CBO and ECBO for optimal design of skeletal structures. *Comput Struct* 153:137–147
2. Kaveh A, Ilchi Ghazaan M (2014) Enhanced colliding bodies optimization for design problems with continuous and discrete variables. *Adv Eng Softw* 77:66–75
3. Kaveh A, Mahdavi VR (2014) Colliding bodies optimization method for optimum design of truss structures with continuous variables. *Adv Eng Softw* 70:1–12
4. American Institute of Steel Construction (AISC) (2001) Manual of steel construction: load and resistance factor design. AISC, Chicago, IL
5. Dumonteil P (1992) Simple equations for effective length factors. *Eng J AISE* 29:111–115
6. Chen WN, Zhang J, Lin Y, Chen N, Zhan ZH, Chang H, Li Y, Shi YH (2013) Particle swarm optimization with an aging leader and challengers. *IEEE Trans Evol Comput* 17(2):241–258
7. Shi Y, Eberhart RC (1998) A modified particle swarm optimizer. *Proc IEEE Congr Evol Comput* 69–73
8. Liang JJ, Qin AK, Suganthan PN, Baskar S (2006) Comprehensive learning particles swarm optimization for global optimization of multimodal functions. *IEEE Trans Evol Comput* 10 (3):281–295
9. Camp CV, Bichon BJ (2004) Design of space trusses using ant colony optimization. *J Struct Eng ASCE* 130:741–751
10. Camp CV (2007) Design of space trusses using big bang-big crunch optimization. *J Struct Eng ASCE* 133:999–1008
11. Sonmez M (2011) Discrete optimum design of truss structures using artificial bee colony algorithm. *Struct Multidiscip Optim* 43:85–97
12. Kaveh A, Ilchi Ghazaan M, Bakhshpoori T (2013) An improved ray optimization algorithm for design of truss structures. *Period Polytech* 57:1–15
13. Kaveh A, Talatahari S (2010) Optimum design of skeletal structure using imperialist competitive algorithm. *Comput Struct* 88:1220–1229
14. Li LJ, Huang ZB, Liu F (2009) A heuristic particle swarm optimization method for truss structures with discrete variables. *Comput Struct* 87:435–443
15. Kaveh A, Talatahari S (2009) A particle swarm ant colony optimization for truss structures with discrete variables. *J Constr Steel Res* 65:1558–1568
16. American Institute of Steel Construction (AISC) (1989) Manual of steel construction—allowable stress design, 9th edn. AISC, Chicago, IL
17. 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
18. 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

19. Kaveh A, Talatahari S (2012) Charged system search for optimal design of planar frame structures. *Appl Soft Comput* 12:382–393
20. Degertekin SO (2008) Optimum design of steel frames using harmony search algorithm. *Struct Multidiscip Optim* 36:393–401
21. Camp CV, Bichon BJ, Stovall S (2005) Design of steel frames using ant colony optimization. *J Struct Eng ASCE* 131:369–379

# Chapter 14

## Global Sensitivity Analysis-Based Optimization Algorithm

### 14.1 Introduction

In this chapter a single-solution metaheuristic optimizer, namely, global sensitivity analysis-based (GSAB) algorithm [1], is presented that uses a basic set of mathematical techniques, namely, global sensitivity analysis. Sensitivity analysis (SA) studies the sensitivity of the model output with respect to its input parameters (Rahman [2]). This analysis is generally categorized as local SA and global SA techniques. While local SA studies the sensitivity of the model output about variations around a specific point, the global SA considers variations of the inputs within their entire feasibility space (Pianosi and Wagener [3], Zhai et al. [4]). One important feature of the GSA is factor prioritization (FP), which aims at ranking the inputs in terms of their relative contribution to output variability. The GSAB comprises of a single-solution optimization strategy and GSA-driven procedure, where the solution is guided by ranking the decision variables using the GSA approach, resulting in an efficient and rapid search. The proposed algorithm can be studied within the family of search algorithms such as the random search (RS) by Rastrigin [5], pattern search (PS) by Hooke and Jeeves [6], and vortex search (VS) by Dog and Ölmez [7] algorithms. In this method, similar to these algorithms, the search process is achieved in the specified boundaries. Contrary to these algorithms that use different functions for decreasing the search space, in the present method, the well-known GSA approach is employed to decrease the search boundaries. The minimization of an objective function is then performed by moving these search spaces into around the best global sample.

The present chapter is organized as follows: In Sect. 14.2, we describe the well-known variance-based sensitivity approach. In Sect. 14.3, the new method is presented. Two well-studied constrained optimization problems and three structural design examples are studied in Sect. 14.4. Conclusions are derived in Sect. 14.5.

## 14.2 Background Study

The single-solution search algorithm proposed in this chapter uses the SA theory. In this section the main strategies used for taking SA into account which are based on the works of Zhai et al. [4], Saltelli et al. [8], and Archer et al. [9] are described.

### 14.2.1 Variance-Based Sensitivity Indices

The variance-based sensitivity indices can be estimated by a numerical model,  $Y = g(X)$ , with  $X = [x_1, x_2, \dots, x_n]$  being the input vector,  $Y$  being the output scalar of the model, and  $g()$  being a deterministic mapping function. Here, the input  $X$  is a random variable. Because of the uncertainty of  $X$  propagating through  $g()$ ,  $Y$  is also a random variable. As the uncertainty of the output model is represented by its variance,  $V(Y)$ , to find the effect of an input  $X_i$  on the output, it is assumed that the true value of  $X_i$  can be determined by the variance reduction in the output, i.e.,  $V(Y) - V(Y|X_i = x_i^0)$ , where  $x_i^0$  is the true value of  $X_i$  and  $V(Y|X_i = x_i^0)$  is the conditional expected value of  $V(Y)$ . Since the true value is unknown, one can employ  $V(Y) - E_{X_i}(V(Y|X_i))$  to evaluate the expected variance reduction in the output (Zhai et al. [4]; Saltelli et al. [8]; Archer et al. [9]).

The variance of output model is calculated utilizing the following equation:

$$V(Y) = V_{X_i}(E(Y|X_i)) + E_{X_i}(V(Y|X_i)) \quad (14.1)$$

And the sensitivity indicator of  $X_i$  can be expressed as (Zhai et al. [4]):

$$SI_i = \frac{V(Y) - E_{X_i}(V(Y|X_i))}{V(Y)} = 1 - \frac{E_{X_i}(V(Y|X_i))}{V(Y)} = \frac{V_{X_i}(E(Y|X_i))}{V(Y)} \quad (14.2)$$

In sensitivity analysis,  $SI_i$  varies between 0 and 1. The lower value of  $SI_i$  corresponds to the less influential  $X_i$ , the higher value of  $SI_i$  corresponds to the much influential  $X_i$ , and for  $SI_i = 0$ , the  $X_i$  will have no influence on  $Y$ .

### 14.2.2 The Variance-Based Sensitivity Analysis Using Space-Partition Method

The most well-known methods for calculating the variance-based sensitivity indicators are the Monte Carlo simulations; however, they do not make full use of each output model evaluation. In order to calculate the variance-based sensitivity indicators from a given data, the scatterplot partitioning method can be utilized

(Zhai et al. [4]). For this method a single set of samples suffices to estimate all the sensitivity indicators. For estimating the variance-based sensitivity indices, a space-partition method is used in the following:

Suppose we have  $M$  points/samples  $\{X^1, \dots, X^M\}$  and  $M$  model output samples  $\{y^1, \dots, y^M\}$  obtained using the model  $y = g(X)$ . The variance of  $Y$  can be calculated by the sample variance  $\hat{V}(Y)$ . For the sample bounds of  $X_i$  as  $[b_1, b_2]$ , let it be decomposed into  $s$  successive, equal-probability, and nonoverlapping subintervals  $A_k = [a_{k-1}, a_k]$ , with  $k=1, \dots, s$ ,  $b_1 = a_0 < a_1 < \dots < a_k < \dots < a_s = b_2$  and  $\Pr(A_k) = 1/s$ . Decompose the output samples  $\{y^1, \dots, y^M\}$  into  $s$  subsets according to the decomposition of  $X_i$ , where  $B_k = \left\{ y^j \mid x_i^j \in A_k \right\}$ ,  $k = 1, \dots, s$ . The variance  $V(Y|x_i \in A_k)$  can then be estimated by:

$$\hat{V}(Y|x_i \in A_k) = V(B_k) \quad (14.3)$$

The expected conditional variance  $E_{x_i}(V(Y|x_i))$  can now be approximately estimated using the following relationship:

$$\hat{E}_{x_i}(V(Y|x_i)) \approx \frac{1}{s} \sum_{k=1}^s V(B_k) \quad (14.4)$$

And ultimately,  $SI_i$  is estimated by:

$$\hat{SI}_i = 1 - \frac{\hat{E}_{x_i}(V(Y|x_i))}{\hat{V}(Y)} \quad (14.5)$$

## 14.3 A Global Sensitivity Analysis-Based Algorithm

This section introduces a global sensitivity analysis-based (GSAB) algorithm, which is a single-solution metaheuristic method. The proposed algorithm is named a global sensitivity analysis (GSA) because of determining the sensitivity indicator ( $SI$ ) of decision variables.

Metaheuristic algorithms can be divided into two categories based on their search mechanism: population based and single solution (Kaveh and Mahdavi [10]). In the first group, a number of populations/agents are first generated, and then all agents updated iteratively until the termination condition is satisfied. On the other hand, single-solution metaheuristics are also known as trajectory methods, in which these algorithms produce single solution by exploring the search space efficiently while reducing the effective size of the search space. The GSAB algorithm consists of some samples for estimating the  $SI$  of decision variables. As these samples do not update iteratively and these are used only for calculating the

SI, the proposed GSBA is studied within the single-solution metaheuristic category. The feasibility space of samples in the GSAB algorithm updates for searching the optimal solution over several iterations. In each iteration, the feasibility space is updated using two values: the sensitivity indicators and the global best sample. It is assumed that the problem is a minimization problem in  $R^D$ . The notations used are as follows:

$S^t$ : The sample matrix in the  $t$ th iteration,  $S^t = [X_i^t | i = 1, 2, \dots, N]$

$X_i^t$ : The position of sample vector  $i$  in the  $t$ th iteration,  $X_i^t = \{x_{ij}^t | j = 1, 2, \dots, D\}$

$X_{min}$ : The minimum allowable value vector of variables,  
 $X_{min} = \{x_{min_j} | j = 1, 2, \dots, D\}$

$X_{max}$ : The maximum allowable value vector of variables,  
 $X_{max} = \{x_{max_j} | j = 1, 2, \dots, D\}$

$f(X_i)$ : The fitness of vector  $i$

$UB^t$ : The upper search boundary vector of variables in the  $t$ th iteration,

$$UB^t = \{ub_j^t | j = 1, 2, \dots, D\}$$

$LB^t$ : The lower search boundary vector of variables in the  $t$ th iteration,

$$LB^t = \{lb_j^t | j = 1, 2, \dots, D\}$$

$BW^t$ : The bandwidth of search space of variables in the  $t$ th iteration,

$$BW^t = \{bw_j^t | j = 1, 2, \dots, D\}$$

$SF^t$ : The scale factor of bandwidth of search space in the  $t$ th iteration,

$$SF^t = \{sf_j^t | j = 1, 2, \dots, D\}$$

$Sbest$ : The global best sample (i.e., with lower fitness),

$$Sbest = \{sbest_j | j = 1, 2, \dots, D\}$$

$R$ : A random vector within  $[0,1]$ .

### 14.3.1 Methodology

The following steps outline the main procedure in the implementation of the GSAB.

**Step 1: Initialization** The initial positions of samples are determined with random initialization in the search space:

$$X_i^0 = X_{min} + R(X_{max} - X_{min}), \quad i = 1, 2, \dots, N \quad (14.6)$$

where  $X_i^0$  determines the initial value vector of the  $i$ th sample and  $N$  is the number of samples. In the first step, some parameter settings must also be predefined for the proposed algorithm. There are two parameters: the number of samples,  $N$ , and the number of subintervals for estimating the sensitivity indices,  $s$ . The number of samples is considered according to the problem's complexity. More complex problems require a higher number of samples. The last parameter is used for GSA

as mentioned in Sect. 14.2.2. These values affect the estimation of  $SI$  [Eq. (14.6)]. A more detailed discussion of these parameters is given in the subsequent subsections.

**Step 2: Detection of the Most Sensitive Variable** In this step the output model, i.e., the objective function of optimization problem, is first calculated. The sensitivity analysis is performed next for the generated samples, and the sensitivity indicators ( $SI$ s) of variables are calculated in Eqs. (14.3)–(14.6). Once the  $SI$ s are known, the most sensitive variable (which it has the high  $SI$ ) and the amount of its  $SI$  are saved for the next step.

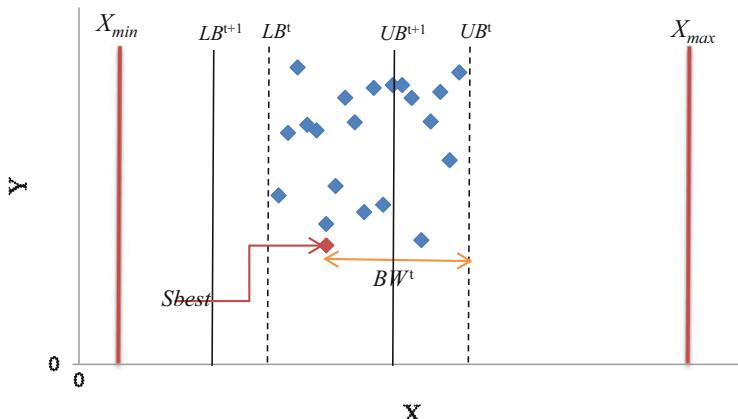
**Step 3: Defining the Search Boundaries** In the GSAB algorithm, the search boundaries are moved to the global best sample (which is updated and memorized in each iteration),  $S_{best}$ , to push the samples into a feasible search space. The search boundaries are also decreased based on the values of the most sensitive variable, which is evaluated in the previous step. Hence, the upper boundary and lower boundary of the search space of variables in the  $t+1$ th iteration can be computed by:

$$\begin{aligned} UB^{t+1} &= S_{best} + BW^t \times SF^t \leq X_{\max} \\ LB^{t+1} &= S_{best} - BW^t \times SF^t \geq X_{\min} \end{aligned} \quad (14.7)$$

where  $BW^t$  and  $SF^t$  are the bandwidth and scale factor of boundaries in the  $t$ th iteration (Fig. 14.1), respectively. Equation (14.8) ensures that the current search space is moved around  $S_{best}$  with the bandwidth  $BW^t$  in the D-dimensional space. The vector  $BW^t$  can be calculated as:

$$BW^t = \max(S_{best} - LB^t, UB^t - S_{best}) \quad (14.8)$$

For the algorithm to converge to a near-optimal solution, further exploitation (strong locality) is required to move the current solution toward to the optimal one. In the proposed GSAB algorithm, this is achieved by using a scale factor,  $SF$ . For this purpose, once  $SI$  values of variables are calculated, the most sensitive variable,



**Fig. 14.1** An illustrative sketch of the search process

i.e., variable with high  $SI$  value, for reducing the bandwidth is identified, and then the  $SF$  is calculated as:

$$SF_j = \begin{cases} 1 - si_j & \text{if } si_j = \max(SI) \\ 1 & \text{Otherwise} \end{cases}, \quad \forall j = 1, \dots, D \quad (14.9)$$

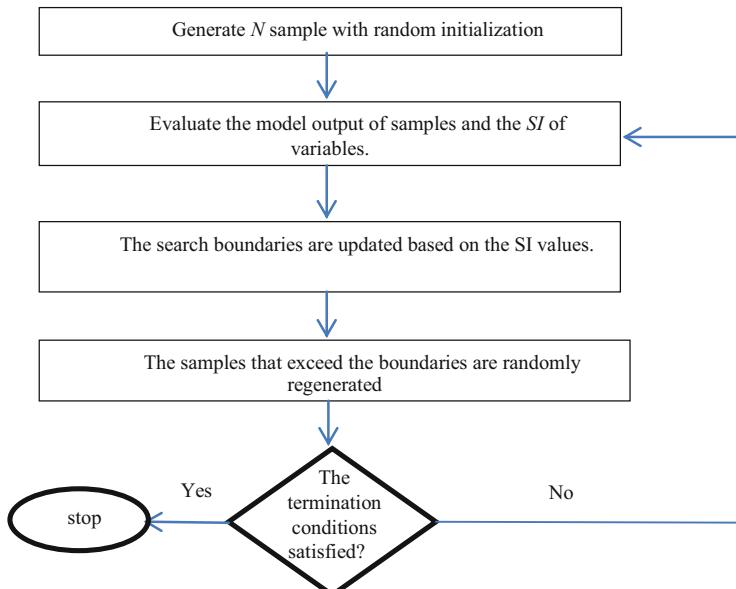
This equation shows that the bandwidth of the most sensitive variable is decreased, while other bandwidths are constant in the  $t$ th iteration.

**Step 4: Replacement of the Current Samples** In this step, the samples must be ensured to be inside the new search boundaries. For this purpose, the samples that exceed the boundaries are randomly regenerated into the new search boundaries, shown in Fig. 14.1, as:

$$X_i^{t+1} = \begin{cases} X_i^t, & LB^{t+1} \leq X_i^t \leq UB^{t+1} \\ LB^{t+1} + R(UB^{t+1} - LB^{t+1}), & \text{Otherwise} \end{cases} \quad (14.10)$$

where  $i=1, 2, \dots, n$  and  $t$  represents the iteration index.

**Step 5: Termination** The optimization process is repeated from Step 2 until a termination criterion, such as maximum iteration number or no improvement of the best sample, is satisfied. In the GSAB algorithm, if the maximum bandwidth of the search space,  $\max(W)$ , becomes smaller than 0.000001, the optimization process will be stopped. This is because the GSAB cannot change the search space of the agents. For the sake of clarity, the flowchart of optimization procedure using the proposed GSAB is shown in Fig. 14.2.



**Fig. 14.2** Flowchart of the GSAB

## 14.4 Numerical Examples

In this section the efficiency of the proposed algorithm, GSAB, is shown through two mathematical constrained functions and three well-studied truss structures under static loads taken from the optimization literature. These examples have been previously solved using a variety of other techniques and are good examples to show the validity and effectiveness of the proposed algorithm. Examples 1 and 2 show the applicable of GSAB for optimization of the constrained problems. In Example 4, a planar truss structure is studied for finding the optimal cross sections. Examples 4 and 5 are selected to show the importance of selection of optimization algorithm in reduction of the number of function evaluations.

In structural optimization problems, the main objective is to minimize the weight of the structures under some constraints. The optimization problem for a truss structure can be stated as follows:

$$\begin{aligned} \text{Find } X &= [x_1, x_2, x_3, \dots, x_n] \\ \text{to minimizes } W(X) &= \sum_{i=1}^{ne} \rho_i A_i l_i \\ \text{subjected to } g_j(X) &\leq 0, j = 1, 2, \dots, m \\ x_{l\min} &\leq x_l \leq x_{l\max} \end{aligned} \quad (14.11)$$

where  $X$  is the vector of all design variables with  $n$  unknowns;  $W$  is the weight of truss structure;  $\rho_i$ ,  $A_i$ , and  $l_i$  are the mass density, cross-sectional area, and length of the  $i$ th member, respectively;  $ne$  is the number of the structural elements;  $g_j$  is the  $j$ th constraint from  $m$  inequality constraints; and, also,  $x_{l\min}$  and  $x_{l\max}$  are the lower and upper bounds of design variable vector, respectively.

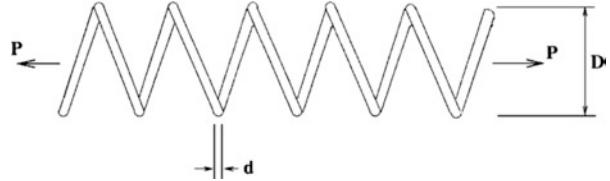
The employed constraint handling is the penalty function approach proposed by Deb [11]. It should be noted that the output model of SA method is the penalized objective function. For truss design and engineering design examples, the numbers of  $N=40$  and  $N=20$  samples are utilized, respectively. Also, all examples are independently optimized 20 times. The algorithm is coded in MATLAB. Structural analysis is performed with the direct stiffness method.

### 14.4.1 Design of a Tension/Compression Spring

This problem was first described by Belegundu [12] and Arora [13]. 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. 14.3.

The design variables are the mean coil diameter  $D(= x_1)$ , the wire diameter  $d(= x_2)$ , and the number of active coils  $N(= x_3)$ . The problem can be stated as follows:

**Fig. 14.3** Schematic of the tension/compression spring with indication of design variables



$$\text{Find } \{x_1, x_2, x_3\} \quad (14.12)$$

To minimize:

$$\cos t(x) = (x_3 + 2)x_2 x_1^2 \quad (14.13)$$

Subject to

$$\begin{aligned} g_1(x) &= 1 - \frac{x_2^3 x_3}{71785 x_1^4} \leq 0 \\ g_2(x) &= \frac{4x_2^2 - x_1 x_2}{12566(x_2 x_1^3 - x_1^4)} + \frac{1}{5108 x_1^2} - 1 \leq 0 \\ g_3(x) &= 1 - \frac{140.45 x_1}{x_2^2 x_3} \leq 0 \\ g_4(x) &= \frac{x_1 + x_2}{1.5} - 1 \leq 0 \end{aligned} \quad (14.14)$$

The bounds on the design variables are:

$$0.05 \leq x_1 \leq 2, \quad 0.25 \leq x_2 \leq 1.3, \quad 2 \leq x_3 \leq 15, \quad (14.15)$$

This problem has been solved by Belegundu [12] using eight different mathematical optimization techniques. Arora [13] solved this problem using a numerical optimization technique called a constraint correction at the constant cost. Coello [14] as well as Coello and Montes [15] solved this problem using GA-based method. Additionally, He and Wang [16] utilized a coevolutionary particle swarm optimization (CPSO). Recently, Montes and Coello [17], Kaveh and Talatahari [18], and Kaveh and Mahdavi [19] used the ES, CSS, and CBO to solve this problem, respectively.

Tables 14.1 and 14.2 compare the best results obtained in this chapter and those of the other researches. The GSAB found the best cost as 0.0126652 after 3729 fitness function evaluations. Although the best cost found is more than the standard CSS, it is the lowest fitness function evaluations among the existing literature results. It should be noted that the lighter design found by Kaveh and Talatahari [18] slightly violates the first two optimization constraints.

In order to show the performance of the GSA method in the GSAB algorithm, a study is focused on the influence of the SIs on the proposed algorithm result. As

**Table 14.1** Comparison of GSAB optimized designs with literature for the tension/compression spring problem

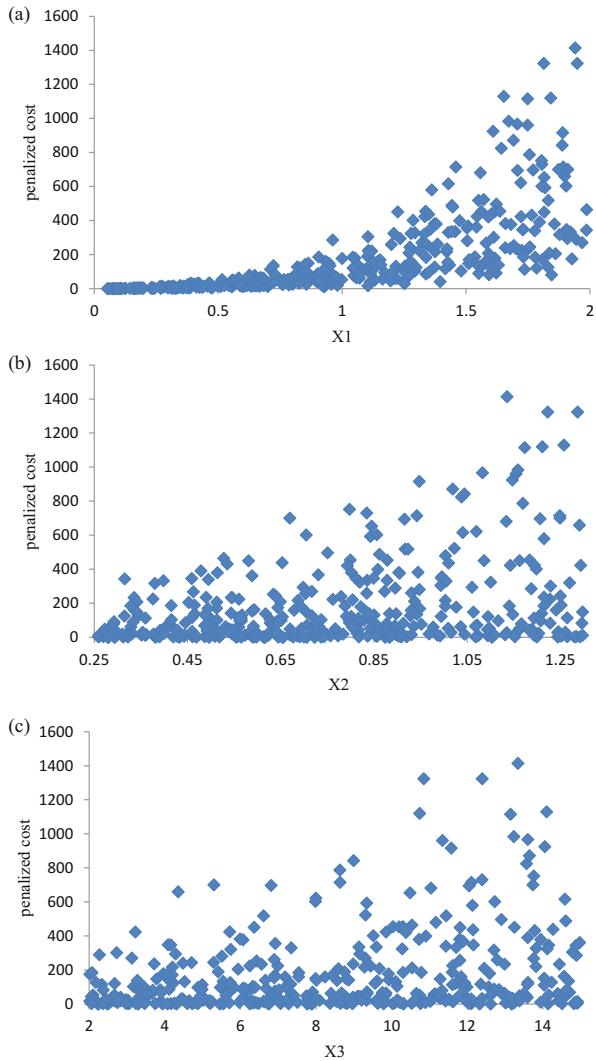
Methods	Optimal design variables			f(x)
	x <sub>1</sub> (d)	x <sub>2</sub> (D)	x <sub>3</sub> (N)	
Belegundu [12]	0.050000	0.315900	14.250000	0.0128334
Arora [13]	0.053396	0.399180	9.185400	0.0127303
Coello [14]	0.051480	0.351661	11.632201	0.0127048
Coello and Montes [15]	0.051989	0.363965	10.890522	0.0126810
He and Wang [16]	0.051728	0.357644	11.244543	0.0126747
Montes and Coello [17]	0.051643	0.355360	11.397926	0.012698
Kaveh and Talatahari [18]	0.051744	0.358532	11.165704	0.0126384
Kaveh and Mahdavi [19]	0.051894	0.3616740	11.007846	0.0126697
Present work [1]	0.05171604	0.3573671	11.2509979	0.0126652

**Table 14.2** Statistical results from different optimization methods for tension/compression string problem

Methods	Best result	Average optimized cost	Worst result	Std dev	Fitness function evaluations
Belegundu [12]	0.0128334	N/A	N/A	N/A	N/A
Arora [13]	0.0127303	N/A	N/A	N/A	N/A
Coello [14]	0.0127048	0.012769	0.012822	3.9390e-5	900,000
Coello and Montes [15]	0.0126810	0.0127420	0.012973	5.9000e-5	N/A
He and Wang [16]	0.0126747	0.012730	0.012924	5.1985e-5	200,000
Montes and Coello [17]	0.012698	0.013461	0.16485	9.6600e-4	25,000
Kaveh and Talatahari [18]	0.0126384	0.012852	0.013626	8.3564e-5	4000
Kaveh and Mahdavi [19]	0.0126697	0.01272964	0.0128808	5.00376e-5	4000
Present work [1]	0.0126652	0.012875334	0.01334400	2.31935e-4	3729

described in Sect. 14.3.1, the GSA method requires two predefined parameters: the number of samples,  $N$ , and the number of subintervals,  $s$ . A larger number of samples lead to an increase of the accuracy of the sensitivity indicators. On the other hand, because of generating the output model of the GSA method, the fitness function (or output) evaluations increase with the number of samples. The number of subintervals can also be affected to the SI values. As Zhai et al. [4] underline, the appropriate number of subintervals can be considered as  $s = \frac{N}{5}$ . The scatter plots of  $X_{i=1,2,3}$  and  $cost$  for  $N = 100$  samples are shown in Fig. 14.4. It can be noticed that (i)  $x_1$  seems to be the most influential input and (ii)  $x_2$  and  $x_3$  seem to be the low

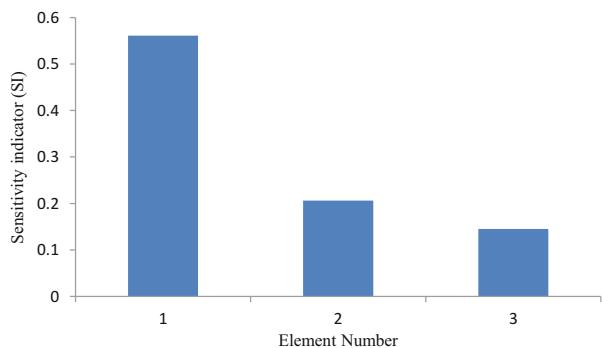
**Fig. 14.4** Scatter plots for variables: (a)  $X_1$ , (b)  $X_2$ , (c)  $X_3$  of the first example



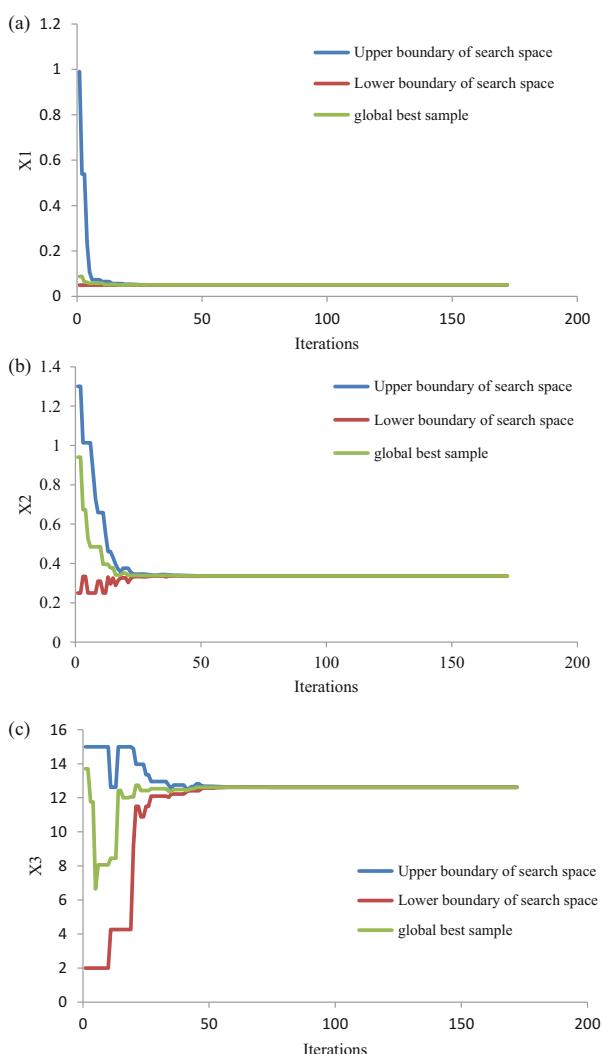
influential inputs, because the distribution of samples against the first variable,  $x_1$ , is more dense compared to other variables. This is confirmed by the GSA method. If we apply the space-partition variance-based sensitivity analysis approach, we obtained the sensitivity indicators,  $SI$ s, in Fig. 14.5. As it can be seen from this figure, the  $SI$  of the first variable is much than other variables; then, the most influential/sensitive variable is the first variable.

Figure 14.6 shows the convergence rates of the upper and lower boundary of the search space and the best ones in the optimization process. It should be noted that, as mentioned before, the number of samples is considered as  $N = 40$  in the optimization process. It can be seen, with respect to the second and third variables, that the

**Fig. 14.5** The obtained sensitivity indicator of variables for penalized cost function of the first example



**Fig. 14.6** The convergence history graphs of search space for variables: (a)  $X_1$ , (b)  $X_2$ , (c)  $X_3$



search space of the first variable is rapidly decreased in the early iterations because it has more sensitivity to output (i.e., objective function). Hence, despite the fewer samples, the proposed GSA approach could appropriately rank the variables based on these sensitivities.

The optimum variables found with different algorithms can also be used for comparing the *SI* of variables. As shown in Table 14.1, although the optimal objective functions found by different optimization algorithms have no significant difference, the values of the optimum second and third variables have significant difference compared to the first variable.

#### 14.4.2 A Constrained Function

This is a 10-variable problem which challenges the algorithm ability to deal with the problem of optimization. This problem also has eight nonlinear inequality constraints and it is defined as

Find

$$\{x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}\} \quad (14.16)$$

To minimize:

$$\begin{aligned} f(x) = & x_1^2 + x_2^2 + x_1x_2 - 14x_1 - 16x_2 + (x_3 - 10)^2 + 4(x_4 - 5)^2 + (x_5 - 3)^2 \\ & + 2(x_6 - 1)^2 + 5x_7^2 + 7(x_8 - 11)^2 + 2(x_9 - 10)^2 + (x_{10} - 7)^2 + 45 \end{aligned} \quad (14.17)$$

Subjected to:

$$\begin{aligned} g_1(x) &= 105 - 4x_1 - 5x_2 + 3x_7 - 9x_8 \geq 0, \\ g_2(x) &= -10x_1 + 8x_2 + 17x_7 - 2x_8 \geq 0, \\ g_3(x) &= 8x_1 - 2x_2 - 5x_9 + 2x_{10} + 12 \geq 0, \\ g_4(x) &= -3(x_1 - 2)^2 - 4(x_2 - 3)^2 - 2x_3^2 + 7x_4 + 120 \geq 0, \\ g_5(x) &= -5x_1^2 - 8x_2 - (x_3 - 6)^2 + 2x_4 + 40 \geq 0, \\ g_6(x) &= -x_1^2 - 2(x_2 - 2)^2 + 2x_1x_2 - 14x_5 + 6x_6 \geq 0, \\ g_7(x) &= -0.5(x_1 - 8)^2 - 2(x_2 - 4)^2 - 3x_5^2 + x_6 + 30 \geq 0, \\ g_8(x) &= 3x_1 - 6x_2 - 12(x_9 - 8)^2 + 7x_{10} \geq 0. \end{aligned} \quad (14.18)$$

The bounds on the design variables are:

$$-10 \leq x_i \leq 10 \quad (i = 1 - 10) \quad (14.19)$$

This problem has been solved by Deb [11] utilizing an efficient constraint handling method for the GA. Lee and Geem [20] and Kaveh and Mahdavi [21]

employed the harmony search and colliding bodies optimization algorithms, respectively.

Tables 14.3 and 14.4 compare the optimal variables, best cost, mean cost, and standard deviation of the results obtained using GSAB with the outcomes of other algorithms. As anticipated, GSAB led to a much better results in terms of best cost and also fitness function evaluations. Figure 14.7 provides the amount of SIs founded by the GSA method, and high sensitivity value of the ninth variable is shown. As it can be seen also in Table 14.3, the best objective function and the optimum value of the ninth variable, with respect to other variables, obtained by the HS algorithm, are similar to these outcomes of the CBO algorithm. However, the best objective function and the optimum value of the ninth variable obtained using the GSAB algorithm are much better than the other two algorithms.

#### 14.4.3 A Planar 17-Bar Truss Problem

A 17-bar planar truss is schematized in Fig. 14.8. The single vertical downward load of 100 kips at node 9 is considered, and there are 17 independent design

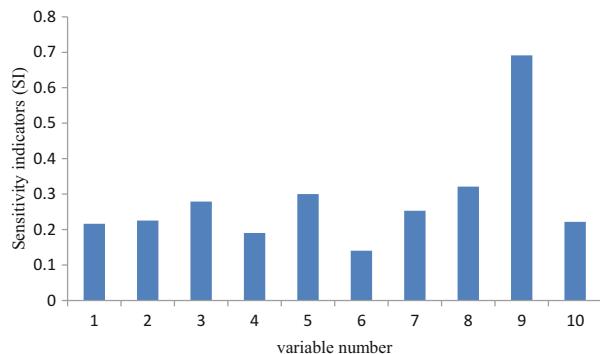
**Table 14.3** Optimal design variables obtained by different researchers for the constrained function

Optimal design variables ( $x$ )	Deb [11]	Lee and Geem [20]	Kaveh and Mahdavi [21]	Present study
$x_1$	Unavailable	2.155225	2.142755	2.193229
$x_2$		2.407687	2.441786	2.229117
$x_3$		8.778069	8.772559	8.747274
$x_4$		5.102078	5.089189	5.074095
$x_5$		0.967625	0.976804	1.011086
$x_6$		1.357685	1.36545	1.38219
$x_7$		1.287760	1.261765	1.347327
$x_8$		9.800438	9.778372	9.902594
$x_9$		8.187803	8.196755	8.308814
$x_{10}$		8.256297	8.362651	8.22824

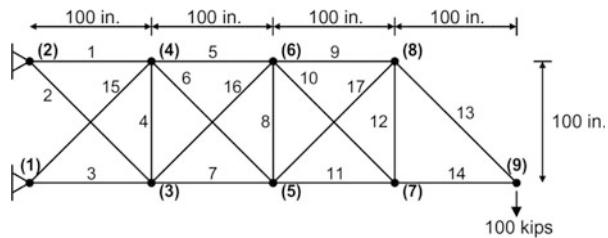
**Table 14.4** Statistical results from different optimization methods for the constrained function

Methods	Best objective function	Average objective function	Std dev	Fitness function evaluations
Deb [11]	24.37248	24.40940	N/A	350,070
Lee and Geem [20]	24.36679	N/A	N/A	230,000
Kaveh and Mahdavi [21]	24.38470	24.86188	0.580431	100,000
Present study	23.91122644	24.87359216	0.768176491	24,693

**Fig. 14.7** The obtained sensitivity indicator of variables for penalized cost function of the constrained function example



**Fig. 14.8** Schematic of the planar 17-bar truss problem



variables. The elastic modulus is 30,000 ksi and the material density is 0.268 lb/in<sup>3</sup> for all elements. The members are subjected to the stress limits of 50 ksi both in tension and compression. Displacement limitations of  $\pm 2.0$  in are imposed on all nodes in both directions (x and y). The allowable minimum cross-sectional area of all the elements is set to 0.1 in<sup>2</sup>.

Table 14.5 presents the optimum designs obtained by Khot and Berke [22], Adeli and Kumar [23], standard CBO, ECBO, Kaveh and Ilchi Ghazaan [24], and the proposed GSA algorithms. Although, the best design is obtained by the ECBO and the work of Khot and Berke [22], the average weight and standard deviation of independent runs obtained by the GSAB are the lowest. The optimization process of the best run of the GSAB is completed in 12,255 analyses. Standard CBO and ECBO required 15,560 and 14,180 analyses to converge to the optimum. The SI values of variables are shown in Fig. 14.9. It can be seen in Fig. 14.9 and Table 14.5 that the sensitivities of members 1, 3, 5, 7, 9, 11, and 13 are more than the remaining members, and the larger optimum designs obtained using optimization algorithms have the high value of SIs.

#### 14.4.4 A 72-Bar Spatial Truss Structure

Schematic topology and element numbering of a 72-bar space truss are shown in Fig. 14.10. The elements are classified into 16 design groups according to

**Table 14.5** Comparison of the optimized designs for the 17-bar planar truss

Element group	Khot and Berke [22]	Adeli and Kumar [23]	Optimal cross-sectional areas		Present work [1]	
			Kaveh and Ilchi [24]			
			CBO	ECBO		
A <sub>1</sub>	15.930	16.029	15.9674	15.9158	15.8916	
A <sub>2</sub>	0.100	0.107	0.1386	0.1001	0.10088	
A <sub>3</sub>	12.070	12.183	12.1735	12.0762	12.00129	
A <sub>4</sub>	0.100	0.110	0.1000	0.1000	0.100015	
A <sub>5</sub>	8.067	8.417	7.8524	8.0527	8.078015	
A <sub>6</sub>	5.562	5.715	5.5447	5.5611	5.571161	
A <sub>7</sub>	11.933	11.331	11.9648	11.9470	11.98603	
A <sub>8</sub>	0.100	0.105	0.1002	0.1000	0.100602	
A <sub>9</sub>	7.945	7.301	7.9385	7.9425	8.009118	
A <sub>10</sub>	0.100	0.115	0.1003	0.1000	0.100585	
A <sub>11</sub>	4.055	4.046	4.1146	4.0589	4.06476	
A <sub>12</sub>	0.100	0.101	0.1000	0.1000	0.100046	
A <sub>13</sub>	5.657	5.611	5.8134	5.6644	5.577003	
A <sub>14</sub>	4.000	4.046	4.0556	4.0057	4.004148	
A <sub>15</sub>	5.558	5.152	5.4973	5.5565	5.611166	
A <sub>16</sub>	0.100	0.107	0.1329	0.1000	0.104159	
A <sub>17</sub>	5.579	5.286	5.4043	5.5740	5.568715	
Best weight (lb)	2581.89	2594.42	2582.79	2581.89	2582.032	
Average weight (lb)	N/A	N/A	2631.07	2597.11	2585.62	
Std dev (lb)	N/A	N/A	49.45	22.43	9.248879	

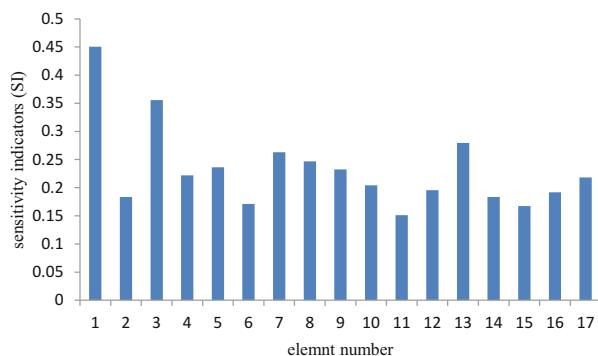
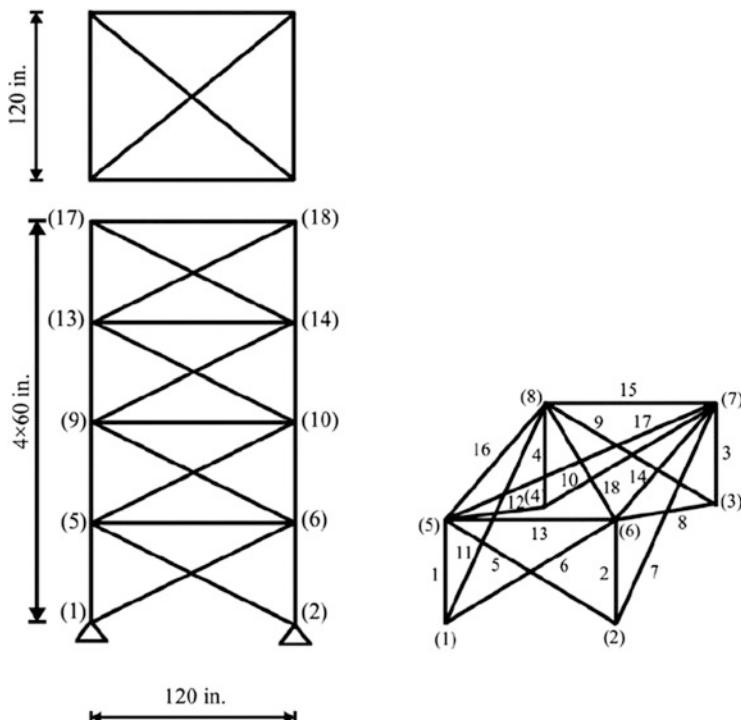
**Fig. 14.9** The obtained sensitivity indicator of variables for the penalized weight of the planar 17-bar truss problem

Table 14.6. The material density is  $0.1 \text{ lb/in}^3$  ( $2767.990 \text{ kg/m}^3$ ), and the modulus of elasticity is taken as 10,000 ksi ( $68,950 \text{ MPa}$ ). The members are subjected to the stress limits of  $\pm 25 \text{ ksi}$  ( $\pm 172.375 \text{ MPa}$ ). The uppermost nodes are subjected to the displacement limits of  $\pm 0.25 \text{ in}$  ( $\pm 0.635 \text{ cm}$ ) in both x and y directions. The



**Fig. 14.10** Schematic of the 72-bar spatial truss

minimum permitted cross-sectional area of each member is taken as  $0.10 \text{ in}^2$  ( $0.6452 \text{ cm}^2$ ), and the maximum cross-sectional area of each member is  $4.00 \text{ in}^2$  ( $25.81 \text{ cm}^2$ ). The loading conditions are considered as:

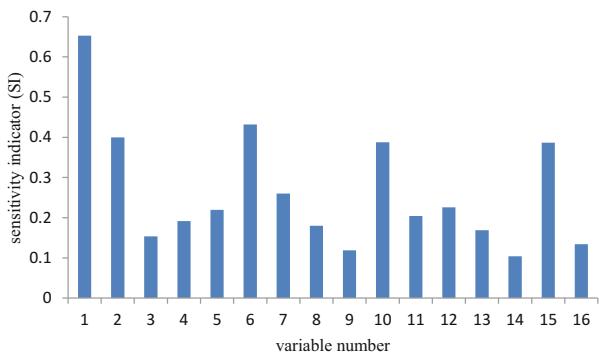
1. Loads 5, 5, and  $-5$  kips in the  $x$ ,  $y$ , and  $z$  directions at node 17, respectively
2. A load equal to  $-5$  kips in the  $z$  direction at nodes 17, 18, 19, and 20

Table 14.6 shows the optimum design variables using the GSAB algorithm, which is compared to the results of the other algorithms. The best result of the GSAB approach is 379.7689, while it is 385.76, 380.24, 381.91, 379.85, 380.458, 379.75, and 379.77 lb for the GA Erbatur et al. [25], ACO Camp and Bichon [26], PSO Perez and Behdinan [27], BB–BC Camp [28], RO Kaveh and Khayatazad [29], CBO and ECBO, and Kaveh and Ilchi Ghazaan [24] algorithms, respectively. Also, the number of analyses of the GSAB is 13,795, while it is 18,500, 19,621, 19,084, 16,000, and 18,000 for the ACO, BB–BC, RO, CBO, and ECBO algorithms, respectively. It is evident in Table 14.6 that although the statistical results of 20 independent runs for the CBO are less than the GSAB algorithm, the number of function evaluations for the GSAB algorithm is less than that of the CBO. Figure 14.11 shows the SI values of the variables for this example. Figure 14.12 shows the maximum stress ratios in truss group members obtained using the GSAB. As it

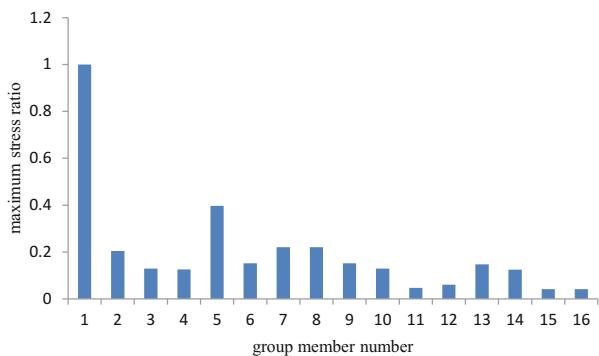
**Table 14.6** Comparison of GSAB optimized designs with those of literature for the 72-bar spatial truss ( $\text{in}^2$ )

Optimal cross-sectional areas ( $\text{in}^2$ )						
Element group	Erbatur et al. [25] GA	Camp et al. [26] ACO	Perez et al. [27] PSO	Camp [28] BB–BC	Kaveh and Khayat [29] RO	Kaveh and Mahdavi [24] CBO
1–4	1.755	1.948	1.7427	1.8577	1.8365	1.9170
5–12	0.505	0.508	0.5185	0.5059	0.5021	0.5031
13–16	0.105	0.101	0.1000	0.1000	0.1000	0.1000
17–18	0.155	0.102	0.1000	0.1000	0.1004	0.1001
19–22	1.155	1.303	1.3079	1.2476	1.2522	1.2721
23–30	0.585	0.511	0.5193	0.5269	0.5033	0.5050
31–34	0.100	0.101	0.1000	0.1000	0.1002	0.1000
35–36	0.100	0.100	0.1000	0.1012	0.1001	0.1000
37–40	0.460	0.561	0.5142	0.5209	0.5730	0.5184
41–48	0.530	0.492	0.5464	0.5172	0.5499	0.5362
49–52	0.120	0.1	0.1000	0.1004	0.1004	0.1000
53–54	0.165	0.107	0.1095	0.1005	0.1001	0.1000
55–58	0.155	0.156	0.1615	0.1565	0.1576	0.1569
59–66	0.535	0.550	0.5092	0.5507	0.5222	0.5374
67–70	0.480	0.390	0.4967	0.3922	0.4356	0.4062
71–72	0.520	0.592	0.5619	0.5922	0.5971	0.5741
Best weight (lb)	385.76	380.24	381.91	379.85	380.458	379.75
Average weight (lb)	N/A	383.16	N/A	382.08	382.553	380.03
Std dev	N/A	3.66	N/A	1.912	1.221	0.278
No. of analyses	N/A	18,500	N/A	19,621	19,084	16,000
						18,000
						13,795
						Present work [1]

**Fig. 14.11** The obtained sensitivity indicator of variables for the penalized weight of 72-bar spatial truss



**Fig. 14.12** The maximum stress ratio in the group elements of the 72-bar truss structure

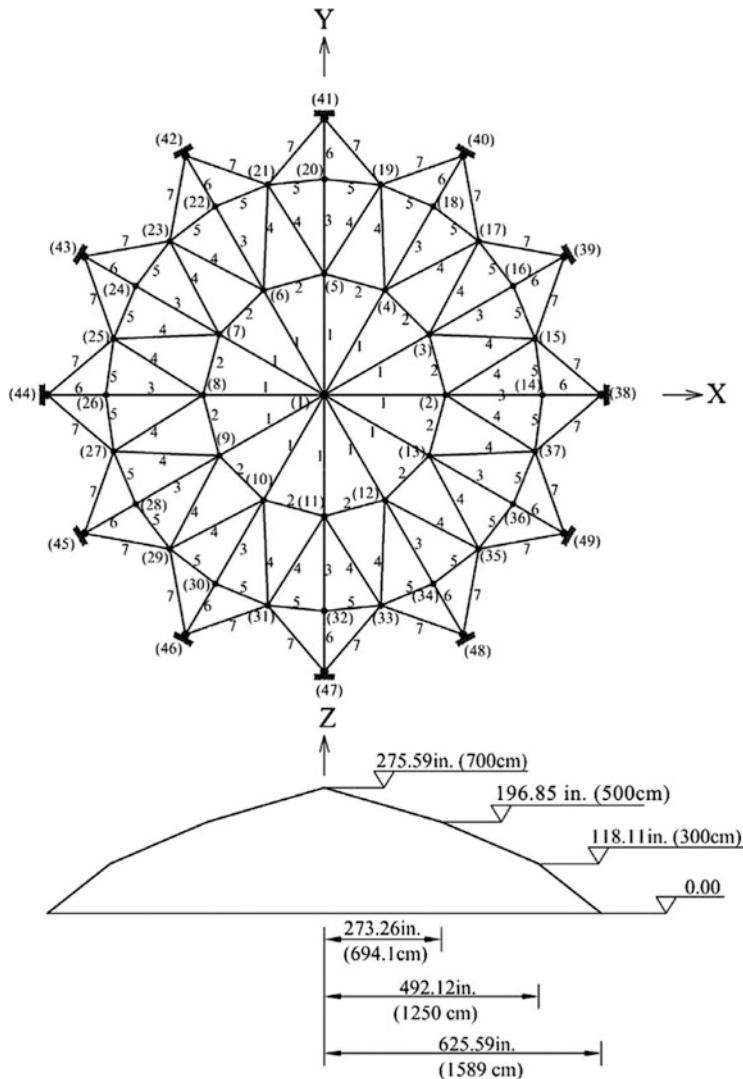


can be seen in Figs. 14.11 and 14.12 and Table 14.6, the first design variable, i.e., the first-story column area, is the most sensitive variable because of the high amount of axial force in the first-story columns. The design variables corresponding to the vertical braces area are also the sensitive variables, with respect to other truss group members, because these can affect the displacement constraints and can have high length in the shape of truss.

#### 14.4.5 A 120-Bar Truss Dome

The last test case solved in this study is the weight minimization problem of the 120-bar truss dome shown in Fig. 14.13. This test case was investigated by Soh and Yang [30] as a configuration optimization problem. It has been solved later as a sizing optimization problem by Kaveh and Talatahari [18], Kaveh and Khayatazad [29], and Kaveh and Mahdavi [19].

The allowable tensile and compressive stresses are set according to the ASD-AISC [31] code, as follows:



**Fig. 14.13** Schematic of the spatial 120-bar dome truss with indication of design variables and main geometric dimensions

$$\begin{cases} \sigma_i^+ = 0.6F_y & \text{for } \sigma_i \geq 0 \\ \sigma_i^- & \text{for } \sigma_i \leq 0 \end{cases} \quad (14.20)$$

where  $\sigma_i^-$  is calculated according to the slenderness ratio:

$$\sigma_i^- = \begin{cases} \left[ \left( 1 - \frac{\lambda_i^2}{2C_c^2} \right) F_y \right] / \left( \frac{5}{3} + \frac{3\lambda_i}{8C_c} - \frac{\lambda_i^3}{8C_c^3} \right) & \text{for } \lambda_i < C_c \\ \frac{12\pi^2 E}{23\lambda_i^2} & \text{for } \lambda_i \geq C_c \end{cases} \quad (14.21)$$

where  $E$  is the modulus of elasticity,  $F_y$  is the yield stress of steel,  $C_c$  is the slenderness ratio ( $\lambda_i$ ) dividing the elastic and inelastic buckling regions ( $C_c = \sqrt{2\pi^2 E/F_y}$ ),  $\lambda_i$  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 and the material density is 0.288 lb/in<sup>3</sup>. The yield stress of steel is taken as 58.0 ksi. On the other hand, the radius of gyration ( $r_i$ ) is expressed in terms of cross-sectional areas as  $r_i = aA_i^{b_i}$  (Saka [32]). 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. 14.7. The dome is considered to be subjected to vertical loads at all 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<sup>2</sup> (cm<sup>2</sup>). In this example, the constraints are considered: Stress constraints and displacement limitations of  $\pm 0.1969$  in are imposed on all nodes in all directions. The maximum cross-sectional area is also considered as 20.0 in<sup>2</sup>.

Table 14.7 summarizes the results obtained by the present work and those of the previously reported researches. As it can be seen, the best results obtained using the GSAB is better than those of the other methods (except for the HPSACO). The standard deviations of results are also better than the RO and CBO algorithms. In this example, the GSAB needs 5823 analyses to find the optimum result, while this number is 10,000, 125,000, 19,800, and 16,000 for the HPSACO, PSOPC, RO, and CBO algorithms as reported, respectively.

## 14.5 Concluding Remarks

In this chapter, a new single-solution global sensitivity analysis-based optimizer called GSAB is developed. Compared to other metaheuristic algorithms, the GSAB has several distinct features. Firstly, the population/agents in GSAB are directly represented by the samples, which are used to find the sensitivity values of the decision variables as well as the optimization search in sequence at each iteration. Hence, one can consider the proposed algorithm as a single-solution metaheuristic category. Secondly, the search boundaries are considered, and these are decreased

**Table 14.7** Comparison of the GSAB optimized designs with those of literature for the 120-bar dome problem

Element group	Optimal cross-sectional areas (in <sup>2</sup> )					
	Kaveh and Talat [18] PSO	Kaveh and Talat [18] PSOPC	Kaveh and Talat [18] HPSACO	Kaveh and Khayat [29] RO	Kaveh and Mahdavi [19] CBO	Present work [1]
1	12.802	3.040	3.095	3.030	3.0284	3.024214
2	11.765	13.149	14.405	14.806	14.9543	14.8525
3	5.654	5.646	5.020	5.440	5.4607	5.064194
4	6.333	3.143	3.352	3.124	3.1214	3.134918
5	6.963	8.759	8.631	8.021	8.0552	8.457656
6	6.492	3.758	3.432	3.614	3.3735	3.283562
7	4.988	2.502	2.499	2.487	2.4899	2.49657
Best weight (lb)	51986.2	33481.2	33248.9	33317.8	33286.3	33249.68
Average weight (lb)	–	–	–	–	33398.5	33253.32
Std dev (lb)	–	–	–	354.333	67.09	4.112399

based on the sensitivity values of the variables at each iteration. The sample, which is the best one, is also selected to push the search boundaries around this sample, and it is selected as solution of the GSAB algorithm. Then, the samples that exceed the search boundaries are randomly regenerated into the boundaries. Unlike the common metaheuristic algorithms where the agents of a population move to the new positions without considering any information about the sensitivity of variables, in this algorithm the search boundaries are decreased based on the sensitivity indices of the variables, and this accelerates the converge of the solution.

The GSAB algorithm is tested over five benchmark optimization problems consisting of mathematical and truss structure optimization problems with different dimensions. The results are compared to those of some population-based metaheuristics. This comparison reveals that besides its simplicity, the proposed GSAB algorithm is also competitive, especially from the number of function evaluations' point of view, when compared to the performance of the some other algorithms.

## References

1. Kaveh A, Mahdavi VR (2016) Optimal design of truss structures using a new metaheuristic algorithm based on global sensitivity analysis. *Struct Eng Mech Int J* 60(26)
2. Rahman S (2011) Global sensitivity analysis by polynomial dimensional decomposition. *Reliab Eng Syst Saf* 96:825–837

3. Pianosi F, Wagener T (2015) A simple and efficient method for global sensitivity analysis based on cumulative distribution functions. *Environ Model Softw* 67:1–11
4. Zhai Q, Yang J, Zhao Y (2014) Space-partition method for the variance-based sensitivity analysis: optimal partition scheme and comparative study. *Reliab Eng Syst Saf* 131:66–82
5. Rastrigin LA (1963) The convergence of the random search method in the extremal control of a many parameter system. *Autom Remote Control* 24(10):1337–1342
6. Hooke R, Jeeves TA (1961) Direct search solution of numerical and statistical problems. *J Assoc Comput Mach* 8(2):212–229
7. Dog B, Ölmez T (2015) A new meta-heuristic for numerical function optimization: Vortex Search algorithm. *Inform Sci* 293:125–145
8. Saltelli A, Annoni P, Azzini I, Campolongo F, Ratto M, Tarantola S (2010) Variance based sensitivity analysis of model output. Design and estimator for the total sensitivity index. *Comput Phys Commun* 181:259–270
9. Archer G, Saltelli A, Sobol I (1997) Sensitivity measures, ANOVA-like techniques and the use of bootstrap. *J Statist Comput Simul* 58:99–120
10. Kaveh A, Mahdavi VR (2015) Colliding bodies optimization; extensions and applications. Springer, Switzerland
11. Deb K (2000) An efficient constraint handling method for genetic algorithms. *Comput Meth Appl Mech Eng* 186(2–4):311–338
12. 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
13. Arora JS (1989) Introduction to optimum design. McGraw-Hill, New York, NY
14. Coello CAC (2000) Use of a self-adaptive penalty approach for engineering optimization problems. *Comput Indust Eng* 41:113–127
15. Coello CAC, Montes EM (2002) Constraint-handling in genetic algorithms through the use of dominance-based tournament. *IEEE Trans Reliab* 41:576–582
16. He Q, Wang L (2007) An effective co-evolutionary particle swarm optimization for constrained engineering design problem. *Eng Appl Artific Intell* 20:89–99
17. Montes EM, Coello CAC (2008) An empirical study about the usefulness of evolution strategies to solve constrained optimization problems. *Int J General Syst* 37:443–473
18. Kaveh A, Talatahari S (2010) A novel heuristic optimization method: charged system search. *Acta Mech* 213:267–289
19. Kaveh A, Mahdavi VR (2014) Colliding bodies optimization: a novel meta-heuristic method. *Comput Struct* 139:18–27
20. Lee KS, Geem ZW (2005) A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice. *Comput Meth Appl Mech Eng* 194 (36–38):3902–3933
21. Kaveh A, Mahdavi VR (2015) A hybrid CBO–PSO algorithm for optimal design of truss structures with dynamic constraints. *Appl Soft Comput* 34:260–273
22. Khot NS, Berke L (1984) Structural optimization using optimality criteria methods. In Atrek E, Gallagher H, Ragsdell KM, Zienkiewicz OC (eds). Wiley, New York
23. Adeli H, Kumar S (1995) Distributed genetic algorithm for structural optimization. *J Aerosp Eng* 8(3):156–163
24. Ilchi KA, Ghazaan M (2014) Enhanced colliding bodies optimization for design problems with continuous and discrete variables. *Adv Eng Softw* 77:66–75
25. Erbatur F, Hasançebi O, Tütüncü I, Kılıç H (2014) Optimal design of planar and space structures with genetic algorithms. *Comput Struct* 75:209–224
26. Camp CV, Bichon BJ (2004) Design of space trusses using ant colony optimization. *J Struct Eng* 130:741–751
27. Perez RE, Behdinan K (2007) Particle swarm approach for structural design optimization. *Comput Struct* 85:1579–1588

28. Camp CV (2007) Design of space trusses using Big Bang–Big Crunch optimization. *J Struct Eng* 133:999–1008
29. Kaveh A, Khayatiazad M (2012) A novel meta-heuristic method: ray optimization. *Comput Struct* 112–113:283–294
30. Soh CK, Yang J (1996) Fuzzy controlled genetic algorithm search for shape optimization. *J Comput Civil Eng* 10:143–150
31. American Institute of Steel Construction (AISC) (1989) Manual of steel construction allowable stress design, 9th edn. Chicago, IL, USA
32. Saka MP (1990) Optimum design of pin-jointed steel structures with practical applications. *J Struct Eng* 116:2599–2620

# Chapter 15

## Tug of War Optimization

### 15.1 Introduction

In this chapter, tug of war optimization (TWO) is presented as a newly developed nature-inspired, population-based metaheuristic algorithm. Utilizing a sport metaphor, the algorithm considers each candidate solution as a team participating in a series of rope-pulling competitions. The teams exert pulling forces on each other based on the quality of the solutions they represent. The competing teams move to their new positions according to the governing laws of motion from the Newtonian mechanics. Unlike many other metaheuristic methods, the algorithm is formulated in such a way that considers the qualities of both of the interacting teams. TWO is applicable to global optimization of discontinuous, multimodal, non-smooth, and non-convex functions.

This chapter consists of two parts. In the first part, the physical background and the basic rules of TWO are presented together with mathematical and engineering design problems in order to show the viability and efficiency of the algorithm [1].

In the second part, the algorithm is applied to different structural optimization problems including truss weight optimization with static constraints [1] and dynamic [2] constraints.

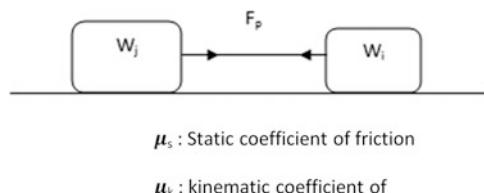
### 15.2 Tug of War Optimization Method

#### 15.2.1 Idealized Tug of War Framework

Tug of war or rope pulling is a strength contest in which two competing teams pull on the opposite ends of a rope in an attempt to bring the rope in their direction against the pulling force of the opposing team. The activity dates back to ancient times and has continued to exist in different forms ever since. There has been a wide



**Fig. 15.1** A competing team in a tug of war competition



**Fig. 15.2** An idealized tug of war framework

variety of rules and regulations for the game, but the essential part has remained almost unaltered. Naturally, as far as both teams sustain their grips of the rope, the movement of the rope corresponds to the displacement of the losing team. Figure 15.1 shows one of the competing teams in a tug of war competition.

Triumph in a real game of tug of war generally depends on many factors and could be difficult to analyze. However, an idealized framework is utilized here, where the two teams having weights  $W_i$  and  $W_j$  are considered as two objects lying on a smooth surface as shown in Fig. 15.2.

As a result of pulling the rope, the teams experience two equal and opposite forces ( $F_p$ ) according to Newton's third law. For object  $i$ , as far as the pulling force is smaller than the maximum static friction force ( $W_i\mu_s$ ), the object rests in its place. Otherwise, the nonzero resultant force can be calculated as:

$$F_r = F_p - W_i\mu_k \quad (15.1)$$

As a result, the object  $i$  accelerates toward the object  $j$  according to the Newton's second law:

$$a = \frac{F_r}{(W_i/g)} \quad (15.2)$$

Since the object  $i$  starts from zero velocity, its new position can be determined as:

$$X_i^{new} = \frac{1}{2}at^2 + X_i^{old} \quad (15.3)$$

### 15.2.2 Tug of War Optimization Algorithm

TWO is a population-based metaheuristic algorithm, which considers each candidate solution  $X_i = \{x_{i,j}\}$  as a team engaged in a series of tug of war competitions. The weight of the teams is determined based on the quality of the corresponding solutions, and the amount of pulling force that a team can exert on the rope is assumed to be proportional to its weight. Naturally, the opposing team will have to maintain at least the same amount of force in order to sustain its grip of the rope. The lighter team accelerates toward the heavier team, and this forms the convergence operator of the TWO. The algorithm improves the quality of the solutions iteratively by maintaining a proper exploration/exploitation balance using the described convergence operator. The steps of TWO can be stated as follows:

**Step 1: Initialization** A population of  $N$  initial solutions is generated randomly:

$$x_{ij}^0 = x_{j,\min} + rand(x_{j,\max} - x_{j,\min}) \quad j = 1, 2, \dots, n \quad (15.4)$$

where  $x_{ij}^0$  is the initial value of the  $j$ th variable of the  $i$ th candidate solution;  $x_{j,\max}$  and  $x_{j,\min}$  are the maximum and minimum permissible values for the  $j$ th variable, respectively;  $rand$  is a random number from a uniform distribution in the interval  $[0, 1]$ ; and  $n$  is the number of optimization variables.

**Step 2: Evaluation of Candidate Designs and Weight Assignment** The objective function values for the candidate solutions are evaluated. All of the initial solutions are sorted and recorded in a memory denoted as the league. Each solution is considered as a team with the following weight:

$$W_i = \left( \frac{fit(i) - fit_{worst}}{fit_{best} - fit_{worst}} \right) + 1 \quad i = 1, 2, \dots, N \quad (15.5)$$

where  $fit(i)$  is the fitness value for the  $i$ th particle, evaluated as the penalized objective function value for constrained problems and  $fit_{best}$  and  $fit_{worst}$  are the fitness values for the best and worst candidate solutions of the current iteration. According to Eq. (15.5), the weights of the teams range between 1 and 2.

**Step 3: Competition and Displacement** In TWO each of the teams of the league competes against all the others one at a time to move to its new position in each iteration. The pulling force exerted by a team is assumed to be equal to its static

friction force ( $W\mu_s$ ). Hence, the pulling force between teams  $i$  and  $j$  ( $F_{p,ij}$ ) can be determined as  $\max\{W_i\mu_s, W_j\mu_s\}$ . Such a definition keeps the position of the heavier team unaltered.

The resultant force affecting team  $i$  due to its interaction with heavier team  $j$  in the  $k$ th iteration can then be calculated as follows:

$$F_{r,ij}^k = F_{p,ij}^k - W_i^k \mu_k \quad (15.6)$$

where  $F_{p,ij}^k$  is the pulling force between teams  $i$  and  $j$  in the  $k$ th iteration and  $\mu_k$  is the coefficient of kinematic friction. Consequently, team  $i$  accelerates toward team  $j$ :

$$a_{ij}^k = \left( \frac{F_{r,ij}^k}{W_i^k \mu_k} \right) g_{ij}^k \quad (15.7)$$

where  $a_{ij}^k$  is the acceleration of team  $i$  toward team  $j$  in the  $k$ th iteration and  $g_{ij}^k$  is the gravitational acceleration constant defined as:

$$g_{ij}^k = X_j^k - X_i^k \quad (15.8)$$

where  $X_j^k$  and  $X_i^k$  are the position vectors for candidate solutions  $j$  and  $i$  in the  $k$ th iteration, respectively. Finally, the displacement of the team  $i$  after competing with team  $j$  can be derived as:

$$\Delta X_{ij}^k = \frac{1}{2} a_{ij}^k \Delta t^2 + \alpha^k \beta (X_{\max} - X_{\min}) \circ \text{randn}(1, n) \quad (15.9)$$

The second term of Eq. (15.9) introduces randomness into the algorithm. This term can be interpreted as the random portion of the search space traveled by team  $i$  before it stops after the applied force is removed. The role of  $\alpha^k$  is to gradually decrease the random portion of the team's movement. For most of the applications,  $\alpha$  could be considered as a constant chosen from the interval [0.9, 0.99]; bigger values of  $\alpha$  decrease the convergence speed of the algorithm and help the candidate solutions explore the search space more thoroughly.  $\beta$  is a scaling factor which can be chosen from the interval (0,1]. This parameter controls the steps of the candidate solutions when moving in the search space. When the search space is supposed to be searched more accurately with smaller steps, smaller values should be chosen for this parameter. For our numerical examples, values between 0.01 and 0.05 seem to be appropriate for this parameter;  $X_{\max}$  and  $X_{\min}$  are the vectors containing the upper and lower bounds of the permissible ranges of the design variables, respectively;  $\circ$  denotes element-by-element multiplication; and  $\text{randn}(1, n)$  is a vector of random numbers drawn from a standard normal distribution.

It should be noted that when team  $j$  is lighter than team  $i$ , the corresponding displacement of team  $i$  will be equal to zero (i.e.,  $\Delta X_{ij}^k$ ). Finally, the total displacement of team  $i$  in iteration  $k$  is equal to ( $i$  not equal  $j$ ):

$$\Delta X_i^k = \sum_{j=1}^N \Delta X_{ij}^k \quad (15.10)$$

The new position of the team  $i$  at the end of the  $k$ th iteration is then calculated as:

$$X_i^{k+1} = X_i^k + \Delta X_i^k \quad (15.11)$$

**Step 4: Updating the League** Once the teams of the league compete against each other for a complete round, the league should be updated. This is done by comparing the new candidate solutions (the new positions of the teams) to the current teams of the league. That is to say, if the new candidate solution  $i$  is better than the  $N$ th team of the league in terms of objective function value, the  $N$ th team is removed from the league, and the new solution takes its place.

**Step 5: Handling the Side Constraints** It is possible for the candidate solutions to leave the search space, and it is important to deal with such solutions properly. This is especially the case for the solutions corresponding to lighter teams for which the values of  $\Delta X$  are usually bigger. Different strategies might be used in order to solve this problem. In this chapter a new strategy is introduced and incorporated using the global best solution. The new value of the  $j$ th optimization variable of the  $i$ th team that violated side constraints in the  $k$ th iteration is defined as:

$$x_{ij}^k = GB_j + \left( \frac{randn}{k} \right) (GB_j - x_{ij}^{k-1}) \quad (15.12)$$

where  $GB_j$  is the  $j$ th variable of the global best solution (i.e., the best-so-far solution) and  $randn$  is a random number drawn from a standard normal distribution. There is a very slight possibility for the newly generated variable to be still outside the search space. In such cases a flyback strategy is used.

The abovementioned strategy is utilized with a certain probability (0.5 in this chapter). For the rest of the cases, the violated limit is taken as the new value of the  $j$ th optimization variable.

**Step 6: Termination** Steps 2 through 5 are repeated until a termination criterion is satisfied. The pseudo code of TWO is presented in Table 15.1.

**Table 15.1** Pseudo code of the TWO algorithm developed in this study

```

procedure Tug of War Optimization
begin
    Initialize parameters;
    Initialize a population of N random candidate solutions;
    Initialize the league by recording all random candidate solution;
    while (termination condition not met) do
        Evaluate the objective function values for the candidate solutions
        Sort the new solutions and update the league
        Define the weights of the teams of the league  $W_i$  based on  $\text{fit}(X_i)$ 
        for each team i
            for each team j
                if ( $W_i < W_j$ )
                    Move team i towards team j using Eq. (15.9);
                end if
            end for
            Determine the total displacement of team i using Eq. (15.10)
            Determine the final position of team i using Eq.(15.11)
            Use the side constraint handling technique to regenerate violating variables
        end for
    end while
end

```

### 15.3 Mathematical and Engineering Design Problems

In order to evaluate the efficiency of the proposed algorithm, some benchmark mathematical and engineering design test problems are considered from the literature. A set of unimodal and multimodal mathematical optimization problems are studied in Sect. 15.3.1. In addition, two well-studied engineering design problems are investigated in Sect. 15.3.2. A population of 20 agents and a maximum number of permitted iterations of 200 are used for all test problems. The coefficient of static friction ( $\mu_s$ ) is taken as unity, while the coefficient of kinematic friction ( $\mu_k$ ) varies linearly from 1 to 0.1. Since smaller values of  $\mu_k$  let the teams slide more easily toward each other and vice versa, such a parameter selection helps the algorithm's agents to explore the search space at early iterations without being severely affected by each other. As the optimization process proceeds, the values of  $\mu_k$  gradually decrease allowing for convergence. It was found that using the same value of kinematic friction coefficient for all teams yields the best optimization results for TWO.

### 15.3.1 Mathematical Optimization Problems

In this section the efficiency of TWO is evaluated by solving the mathematical benchmark problems summarized in Table 15.2. These benchmark problems are taken from [3], where some variants of GA were used as the optimization algorithm. The results obtained by TWO are presented in Table 15.3 along with those of GA variants. Each objective function is optimized 50 times independently starting from different initial populations, and the average number of function evaluations required by each algorithm is presented. The numbers in the parentheses indicate the ratio of the successful runs in which the algorithm has located the global minimum with predefined accuracy, which is taken as  $\epsilon = f_{\min} - f_{final} = 10^{-4}$ . The absence of the parentheses means that the algorithm has been successful in all independent runs.

As it can be seen from Table 15.3, TWO generally performs better than GA and its variants in the mathematical optimization problems considered in this study.

### 15.3.2 Engineering Design Problems

In order to further investigate the efficiency of the TWO, two engineering design problems are considered in this section. These problems have been previously studied using different optimization algorithms. The constrained optimization problems are turned into unconstrained ones using a penalty approach. If the constraints are satisfied, then the amount of penalty will be zero; otherwise, its value can be calculated as the ratio of violated constraint to the corresponding allowable limit.

#### 15.3.2.1 Design of a Tension/Compression Spring

Weight minimization of the tension/compression spring shown in Fig. 15.3, subject to constraints on shear stress, surge frequency, and minimum deflection, is considered as the first engineering design example. This problem was first described by Belegundu [4] and Arora [5]. The design variables are the mean coil diameter  $D(x_1)$ , the wire diameter  $d(x_2)$ , and the number of active coils  $N(x_3)$ .

The cost function can be stated as:

$$f_{\text{cost}}(\mathbf{X}) = (x_3 + 2)x_2 x_1^2 \quad (15.13)$$

while the constraints are:

**Table 15.2** Details of the benchmark mathematical problems solved in this study

Function name	Side constraints	Function	Global minimum
Aluffi-Pentini	$\mathbf{X} \in [-10, 10]^2$	$f(\mathbf{X}) = \frac{1}{4}x_1^4 - \frac{1}{2}x_1^2 + \frac{1}{10}x_1 + \frac{1}{2}x_2^2$	-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 Lagoo	$\mathbf{X} \in [-10, 10]^2$	$f(\mathbf{X}) = ( x_1  - 5)^2 + ( x_2  - 5)^2$	0.0
Brainin	$0 \leq x_2 \leq 15 - 5 \leq x_1 \leq 10$	$f(\mathbf{X}) = (x_2 - \frac{5}{4}\frac{x_1^2}{x_1^2 + \frac{5}{4}x_1})^2 + 10(1 - \frac{1}{8\pi}) \cos(x_1) + 10$	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	$\mathbf{X} \in [-5, 5]^2$	$f(\mathbf{X}) = 2x_1^2 - 1.05x_1^4 + \frac{1}{6}x_1^6 + x_1x_2 + x_2^2$	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
DeJong	$\mathbf{X} \in [-5, 12, 5, 12]^3$	$f(\mathbf{X}) = x_1^2 + x_2^2 + x_3^2$	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	$\mathbf{X} \in [-2, 2]^2$	$f(\mathbf{X}) = \frac{[1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)]}{[30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]^2}$	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 $p = \begin{bmatrix} 0.3689 & 0.1117 & 0.2673 \\ 0.4699 & 0.4387 & 0.747 \\ 0.1091 & 0.8732 & 0.5547 \\ 0.03815 & 0.5743 & 0.8828 \end{bmatrix}$	-3.862782

Hartman 6	$\mathbf{X} \in [0, 1]^6$	$f(\mathbf{X}) = -\sum_{i=1}^4 c_i \exp \left( -\sum_{j=1}^6 a_{ij} (x_j - p_{ij})^2 \right)$ $a = \begin{bmatrix} 10 & 3 & 17 & 3.5 & 1.7 & 8 \\ 0.05 & 10 & 17 & 0.1 & 8 & 14 \\ 3 & 3.5 & 17 & 10 & 17 & 8 \\ 17 & 8 & 0.05 & 10 & 0.1 & 14 \end{bmatrix}, c = \begin{bmatrix} 1 \\ 1.2 \\ 3 \\ 3.2 \end{bmatrix}$ $p = \begin{bmatrix} 0.1312 & 0.1696 & 0.5569 & 0.0124 & 0.8283 & 0.5886 \\ 0.2329 & 0.4135 & 0.8307 & 0.3736 & 0.1004 & 0.9991 \\ 0.2348 & 0.1451 & 0.3522 & 0.2883 & 0.3047 & 0.6650 \\ 0.4047 & 0.8828 & 0.8732 & 0.5743 & 0.1091 & 0.0381 \end{bmatrix}$	-3.322368
-----------	---------------------------	--	-----------

**Table 15.3** Performance comparison of TWO and GA variants in the mathematical optimization problems

Function name	GEN	GEN-S	GEN-S-M	GEN-S-M-LS	TWO [1]
AP	1360 (0.99)	1360	1277	1253	1092
Bf1	3992	3356	1640	1615	1404
Bf2	20,234	3373	1676	1636	1232
BL	19,596	2412	2439	1436	1216
Branin	1442	1418	1404	1257	1189
Camel	1358	1358	1336	1300	1212
Cb3	9771	2045	1163	1118	992
CM	2105	2105	1743	1539	1420
DeJoung	9900	3040	1462	1281	1346
Exp2	938	936	817	807	314
Exp4	3237	3237	2054	1496	815
Exp8	3237	3237	2054	1496	1257
Goldstein and Price	1478	1478	1408	1325	1690
Griewank	18,838 (0.91)	3111 (0.91)	1764	1652 (0.99)	1766
Hartman3	1350	1350	1332	1274	1026
Harmann6	2562 (0.54)	2562 (0.54)	2530 (0.67)	1865 (0.68)	1601 (0.68)

**Fig. 15.3** Schematic of the tension/compression spring

$$\begin{aligned}
 g_1(\mathbf{X}) &= 1 - \frac{x_2^3 x_3}{71785 x_1^4} \leq 0 \\
 g_2(\mathbf{X}) &= \frac{4x_2^2 - x_1 x_2}{12566(x_2 x_1^3 - x_1^4)} + \frac{1}{5108 x_1^2} - 1 \leq 0 \\
 g_3(\mathbf{X}) &= 1 - \frac{140.45 x_1}{x_2^2 x_3} \leq 0 \\
 g_4(\mathbf{X}) &= \frac{x_1 + x_2}{1.5} - 1 \leq 0
 \end{aligned} \tag{15.14}$$

The side constraints are defined as follows:

**Table 15.4** Comparison of the optimization results obtained in the tension/compression spring problem

Methods	Optimal design variables			
	$x_1(d)$	$x_2(D)$	$x_3(N)$	$f_{\text{cost}}$
Belegundu [4]	0.050000	0.315900	14.250000	0.0128334
Arora [5]	0.053396	0.399180	9.185400	0.0127303
Coello [6]	0.051480	0.351661	11.632201	0.0127048
Coello and Montes [7]	0.051989	0.363965	10.890522	0.0126810
He and Wang [8]	0.051728	0.357644	11.244543	0.0126747
Montes and Coello [9]	0.051643	0.355360	11.397926	0.012698
Kaveh and Talatahari [10]	0.051865	0.361500	11.000000	0.0126432
Kaveh and Talatahari [11]	0.051744	0.358532	11.165704	0.0126384
Kaveh and Mahdavi [12]	0.051894	0.3616740	11.007846	0.0126697
TWO [1]	0.051592	0.354379	11.428784	0.0126671

$$\begin{aligned} 0.05 &\leq x_1 \leq 2 \\ 0.25 &\leq x_2 \leq 1.3 \\ 2 &\leq x_3 \leq 15 \end{aligned} \quad (15.15)$$

This problem has been solved by Belegundu [4] using eight different mathematical optimization techniques. Arora [5] utilized a numerical optimization technique called a constraint correction at the constant cost to investigate the problem. Coello [6] and Coello and Montes [7] used a GA-based algorithm to solve the problem. He and Wang [8] used a coevolutionary particle swarm optimization (CPSO). Montes and Coello [9] used evolution strategies. Kaveh and Talatahari used improved ant colony optimization [10] and charged system search (CSS) [11]. Recently, the problem has been solved by Kaveh and Mahdavi [12] using colliding bodies optimization (CBO).

The optimization results obtained by TWO are presented in Table 15.4 along with those of other methods. It can be seen that TWO found the best design overall. It should be noted that some constraints are slightly violated by the designs in [10, 11]. Table 15.5 shows the statistical results obtained for 30 independent optimization runs.

### 15.3.2.2 Design of a Welded Beam

The second test problem regards the design optimization of the welded beam shown in Fig. 15.4. This problem has been used for testing different optimization methods. The aim is to minimize the total manufacturing cost subject to constraints on shear stress ( $\tau$ ), bending stress ( $\sigma$ ), buckling load ( $P_c$ ), and deflection ( $\delta$ ). The four design variables, namely,  $h(x_1)$ ,  $l(x_2)$ ,  $t(x_3)$ , and  $b(x_4)$ , are also shown in the figure.

The objective function can be mathematically stated as:

$$f_{\text{cost}}(\mathbf{X}) = 1.10471x_1^2x_2 + 0.04811x_3x_4(14.0 + x_2) \quad (15.16)$$

The optimization constraints are

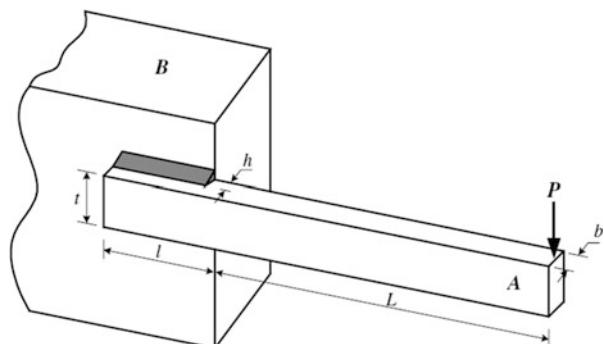
$$\begin{aligned} g_1(\mathbf{X}) &= \tau(\{x\}) - \tau_{\max} \leq 0 \\ g_2(\mathbf{X}) &= \sigma(\{x\}) - \sigma_{\max} \leq 0 \\ g_3(\mathbf{X}) &= x_1 - x_4 \leq 0 \\ g_4(\mathbf{X}) &= 0.10471x_1^2 + 0.04811x_3x_4(14.0 + x_2) - 5.0 \leq 0 \\ g_5(\mathbf{X}) &= 0.125 - x_1 \leq 0 \\ g_6(\mathbf{X}) &= \delta(\{x\}) - \delta_{\max} \leq 0 \\ g_7(\mathbf{X}) &= P - P_c(\{x\}) \leq 0 \end{aligned} \quad (15.17)$$

where

**Table 15.5** Comparison of statistical optimization results obtained in the tension/compression spring design problem

Methods	Best	Mean	Worst	Std dev
Belegundu [4]	0.0128334	N/A	N/A	N/A
Arora [5]	0.0127303	N/A	N/A	N/A
Coello [6]	0.0127048	0.012769	0.012822	3.9390e-5
Coello and Montes [7]	0.0126810	0.0127420	0.012973	5.9000e-5
He and Wang [8]	0.0126747	0.012730	0.012924	5.1985e-5
Montes and Coello [9]	0.012698	0.013461	0.016485	9.6600e-4
Kaveh and Talatahari [10]	0.0126432	0.012720	0.012884	3.4888e-5
Kaveh and Talatahari [11]	0.0126384	0.012852	0.013626	8.3564e-5
Kaveh and Mahdavi [12]	0.0126697	0.0127296	0.012881	5.00376e-5
TWO [1]	0.0126671	0.0129709	0.0135213	2.6125e-4

**Fig. 15.4** Schematic of the welded beam



$$\begin{aligned}
\tau(\mathbf{X}) &= \sqrt{(\tau')^2 + 2\tau'\tau'' \frac{x_2}{2R} + (\tau'')^2} \\
\tau' &= \frac{P}{\sqrt{2x_1x_2}}, \quad \tau'' = \frac{MR}{J} \\
M &= P \left( L + \frac{x_2}{2} \right), \quad R = \sqrt{\frac{x_2^2}{4} + \left( \frac{x_1 + x_3}{2} \right)^2} \\
J &= 2 \left\{ \sqrt{2x_1x_2} \left[ \frac{x_2^2}{12} + \left( \frac{x_1 + x_3}{2} \right)^2 \right] \right\} \\
\sigma(X) &= \frac{6PL}{x_4x_3^2}, \quad \delta(X) = \frac{4PL^3}{Ex_3^3x_4} \\
P_c(X) &= \frac{4.013E \sqrt{\frac{x_3^2x_4^6}{36}}}{L^2} \left( 1 - \frac{x_3}{2L} \sqrt{\frac{E}{4G}} \right) \\
P &= 6000lb, \quad L = 14in \\
E &= 30 \times 10^6 psi, \quad G = 12 \times 10^6 psi \\
\tau_{\max} &= 13.6 \times 10^3 psi, \quad \sigma_{\max} = 30 \times 10^3 psi \\
\delta_{\max} &= 0.25in
\end{aligned} \tag{15.18}$$

The side constraints can be stated as:

$$0.1 \leq x_1 \leq 2, \quad 0.1 \leq x_2 \leq 10, \quad 0.1 \leq x_3 \leq 10, \quad 0.1 \leq x_4 \leq 2 \tag{15.19}$$

Radgsdell and Phillips [13] utilized different optimization methods mainly based on mathematical programming to solve the problem and compared the results. GA-based methods are used by Deb [14], Coello [6], and Coello and Montes [7]. This test problem was also solved by He and Wang [8] using CPSO and Montes and Coello [9] using evolution strategies. Kaveh and Talatahari employed ant colony optimization [10] and charged system search [11]. Kaveh and Mahdavi [12] solved the problem using the colliding bodies optimization method.

Table 15.6 compares the best results obtained by the different optimization algorithms considered in this study. The statistical results for 30 independent runs are provided in Table 15.7. It can be seen from Table 15.6 that optimum design found by TWO is about 0.01 % heavier than that found by CBO which was the best design reported so far in literature.

## 15.4 Structural Optimization Problems

In this section the TWO is applied to different structural optimization problems including truss weight optimization with static [1] and dynamic constraints [2].

**Table 15.6** Comparison of the optimization results obtained in the welded beam problem

Methods	Optimal design variables				
	$x_1(h)$	$x_2(l)$	$x_3(t)$	$x_4(b)$	$f_{\text{cost}}$
Radgsdell and Phillips [13]	0.2444	6.2189	8.2915	0.2444	2.3815
Deb [14]	0.248900	6.173000	8.178900	0.253300	2.433116
Coello [6]	0.208800	3.420500	8.997500	0.210000	1.748309
Coello and Montes [7]	0.205986	3.471328	9.020224	0.206480	1.728226
He and Wang [8]	0.202369	3.544214	9.048210	0.205723	1.728024
Montes and Coello [9]	0.199742	3.612060	9.037500	0.206082	1.737300
Kaveh and Talatahari [10]	0.205700	3.471131	9.036683	0.205731	1.724918
Kaveh and Talatahari [11]	0.205820	3.468109	9.038024	0.205723	1.724866
Kaveh and Mahdavi [12]	0.205722	3.47041	9.037276	0.205735	1.724663
TWO [1]	0.205728	3.47052	9.036631	0.205730	1.724855

**Table 15.7** Comparison of statistical optimization results obtained in the welded beam problem

Methods	Best	Mean	Worst	Std Dev
Radgsdell and Phillips [13]	2.3815	N/A	N/A	N/A
Deb [14]	2.433116	N/A	N/A	N/A
Coello [6]	1.748309	1.771973	1.785835	0.011220
Coello and Montes [7]	1.728226	1.792654	1.993408	0.074713
He and Wang [8]	1.728024	1.748831	1.782143	0.012926
Montes and Coello [9]	1.737300	1.813290	1.994651	0.070500
Kaveh and Talatahari [10]	1.724918	1.729752	1.775961	0.009200
Kaveh and Talatahari [11]	1.724866	1.739654	1.759479	0.008064
Kaveh and Mahdavi [12]	1.724662	1.725707	1.725059	0.0002437
TWO [1]	1.724855	1.726016	1.729970	0.0009951

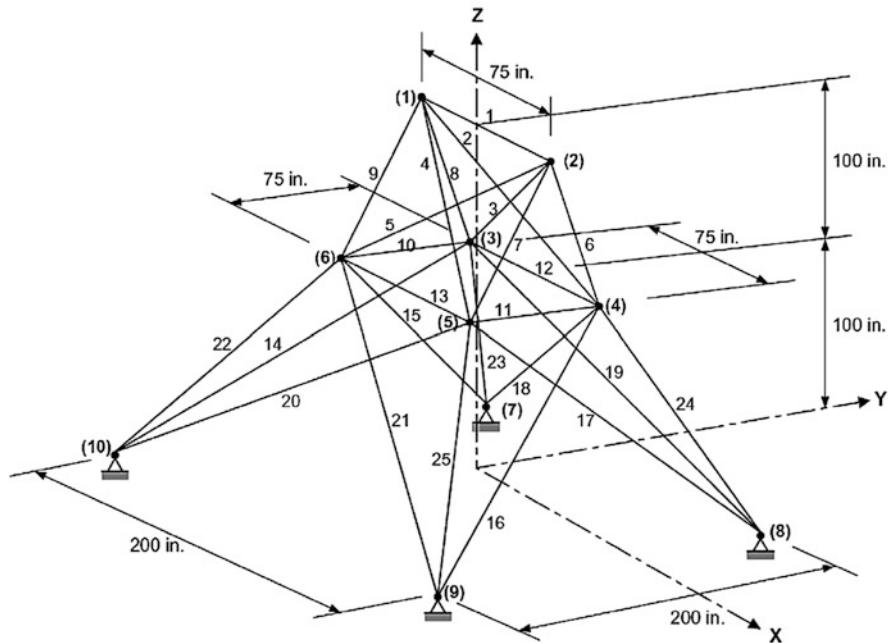
### 15.4.1 Truss Weight Optimization with Static Constraints

In this section, three truss weight optimization problems with constraints of stresses and displacements are presented. 30 agents are utilized for the first example, while 40 agents are used for the second and third examples. Maximum permissible iterations are considered as 400 for all examples.

#### 15.4.1.1 Design of a Spatial 25-Bar Truss Structure

The spatial 25-bar truss schematized in Fig. 15.5 is the first structural design example. The material density and modulus of elasticity are 0.1 lb/in<sup>3</sup> and 10,000 ksi, respectively. Table 15.8 shows the two independent loading conditions applied to the structure. The 25 bars of the truss are classified into eight groups as follows:

- (1) A<sub>1</sub>, (2) A<sub>2</sub>–A<sub>5</sub>, (3) A<sub>6</sub>–A<sub>9</sub>, (4) A<sub>10</sub>–A<sub>11</sub>, (5) A<sub>12</sub>–A<sub>13</sub>, (6) A<sub>14</sub>–A<sub>17</sub>, (7) A<sub>18</sub>–A<sub>21</sub>, and (8) A<sub>22</sub>–A<sub>25</sub>.



**Fig. 15.5** Schematic of the spatial 25-bar truss structure

**Table 15.8** Independent loading conditions acting on the spatial 25-bar truss

Node	Case 1			Case 2		
	P <sub>x</sub> kips	P <sub>y</sub> kips	P <sub>z</sub> kips	P <sub>x</sub> kips	P <sub>y</sub> kips	P <sub>z</sub> kips
1	0.0	20.0	-5.0	1.0	10.0	-5.0
2	0.0	-20.0	-5.0	0.0	10.0	-5.0
3	0.0	0.0	0.0	0.5	0.0	0.0
6	0.0	0.0	0.0	0.5	0.0	0.0

Maximum displacement limitations of 0.35 in are imposed on all nodes in all directions. The axial stress constraints, which are different for each group, are shown in Table 15.9. The cross-sectional areas vary continuously from 0.01 to 3.4 in<sup>2</sup> for all members.

Table 15.10 shows that the different optimization methods converged almost to the same structural weight. The best result obtained by TWO is 544.42 kg which is only slightly heavier than that of HS. However, it should be noted that according to our codes, the optimum design of HS slightly violates displacement constraints. Moreover, TWO requires 12,000 analyses to complete the optimization process, while HS has used 15000 analyses. The mean value and standard deviation of 30 independent optimization runs by TWO are 544.53 and 0.211 lb, respectively. Small differences in weight may be originated from the level of precision in the implementation of the optimization problem. For example, the displacement limit

**Table 15.9** Member stress limits for the 25-bar spatial truss

Element group	Compressive stress limits ksi (MPa)	Tensile stress limit 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 15.10** Comparison of the optimization results obtained in the spatial 25-bar truss problem

Element group		Optimal cross-sectional areas ( $\text{in}^2$ )			
		Rajeev and Krishnamoorthy, GA [15]	Schutte and Groenwold, PSO [16]	Lee and Geem, HS [17]	TWO [1]
1	A <sub>1</sub>	0.10	0.010	0.047	0.010
2	A <sub>2</sub> –A <sub>5</sub>	1.80	2.121	2.022	1.979
3	A <sub>6</sub> –A <sub>9</sub>	2.30	2.893	2.950	2.993
4	A <sub>10</sub> –A <sub>11</sub>	0.20	0.010	0.010	0.010
5	A <sub>12</sub> –A <sub>13</sub>	0.10	0.010	0.014	0.010
6	A <sub>14</sub> –A <sub>17</sub>	0.80	0.671	0.688	0.684
7	A <sub>18</sub> –A <sub>21</sub>	1.80	1.611	1.657	1.678
8	A <sub>22</sub> –A <sub>25</sub>	3.0	2.717	2.663	2.656
Best weight (lb)	546	545.21	544.38	544.42	
Average weight (lb)	N/A	546.84	N/A	544.53	
Std dev (lb)	N/A	1.478	N/A	0.211	
No. of analyses	N/A	9596	15,000	12,000	

of 0.350 in set for TWO resulted in feasible optimized design reported in Table 15.10, yielding a maximum displacement of 0.3504 in which can be rounded to the abovementioned limit.

#### 15.4.1.2 Design of a Spatial 72-Bar Truss Structure

A spatial 72-bar truss structure shown in Fig. 15.6 is considered as the second truss example. The two loading conditions acting on the structure are summarized in Table 15.11. The elements are grouped to form 16 design variables according to Table 15.12. The material density and the modulus of elasticity are taken as 0.1 lb/in<sup>3</sup> and 10,000 ksi, respectively. All members are subjected to a stress limitation of

$\pm 25$  ksi. The displacements of the uppermost nodes along x- and y-axes are limited to  $\pm 0.25$  in. Cross-sectional areas of bars can vary between 0.10 and 4.00 in $^2$ , respectively.

This problem has been studied by Erbatur et al. [19] using genetic algorithms, Camp and Bichon [19] using ant colony optimization, Perez and Behdinan [20] using particle swarm optimization, Camp [21] using Big Bang–Big Crunch algorithm, and Kaveh and Khayatazad [22] using ray optimization, among others.

Table 15.12 compares the results obtained by the TWO to those previously reported in the literature. The weight of the best result obtained by TWO is 379.846 lb which is the best among the compared methods. Moreover, the mean weight of the results for 20 independent optimization runs of TWO is 381.98 lb which is less than all other methods. Also, TWO requires only 16,000 structural analyses while ACO, BB–BC, and RO require 18,500, 19,621, and 19,084, respectively.

#### 15.4.1.3 Design of a 120-Bar Dome Truss

The third test problem in this section is the weight minimization of a 120-bar dome truss shown in Fig. 15.7. This structure was considered by Soh and Yang [23] as a configuration optimization problem. It has been solved by Lee and Geem [17], Kaveh and Talatahari [23], Kaveh et al. [24], Kaveh and Khayatazad [22], and Kaveh and Mahdavi [12] as a sizing optimization problem. The members of the structure are divided into seven groups as shown in Fig. 15.7.

The allowable tensile and compressive stresses are set according to the ASD-AISC (1989) provisions as follows:

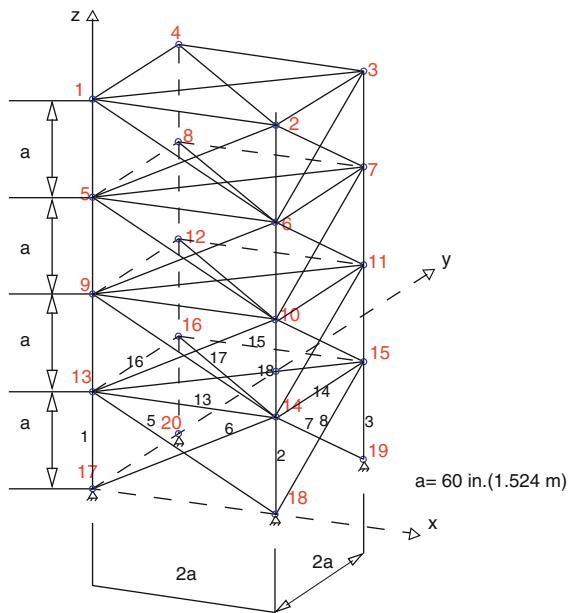
$$\begin{cases} \sigma_i^+ = 0.6F_y & \text{for } \sigma_i \geq 0 \\ \sigma_i^- & \text{for } \sigma_i \leq 0 \end{cases} \quad (15.20)$$

where  $\sigma_i^-$  is the compressive allowable stress and depends on the elements slenderness ratio:

$$\sigma_i^- = \begin{cases} \left[ \left( 1 - \frac{\lambda_i^2}{2C_c^2} \right) F_y \right] / \left( \frac{5}{3} + \frac{3\lambda_i}{8C_c} - \frac{\lambda_i^3}{8C_c^3} \right) & \text{for } \lambda_i \leq C_c \\ \frac{12\pi^2 E}{23\lambda_i^2} & \text{for } \lambda_i \geq C_c \end{cases} \quad (15.21)$$

where E is the modulus of elasticity,  $F_y$  is the material's yield stress,  $\lambda_i$  is the slenderness ratio ( $\lambda_i = \frac{K_i L_i}{r_i}$ ),  $K_i$  is the effective length factor,  $L_i$  is the member length,  $r_i$  is the radius of gyration, and  $C_c$  is the critical slenderness ratio separating elastic and inelastic buckling regions ( $C_c = \sqrt{2\pi^2 E/F_y}$ ).

**Fig. 15.6** Schematic of the spatial 72-bar truss structure



**Table 15.11** Independent loading conditions acting on the spatial 72-bar truss

Node	Case 1			Case 2		
	P <sub>x</sub> kips (kN)	P <sub>y</sub> kips (kN)	P <sub>z</sub> kips (kN)	P <sub>x</sub> kips(kN)	P <sub>y</sub> kips(kN)	P <sub>z</sub> kips (kN)
1	5	5	-5	-	-	-5
2	-	-	-	-	-	-5
3	-	-	-	-	-	-5
4	-	-	-	-	-	-5

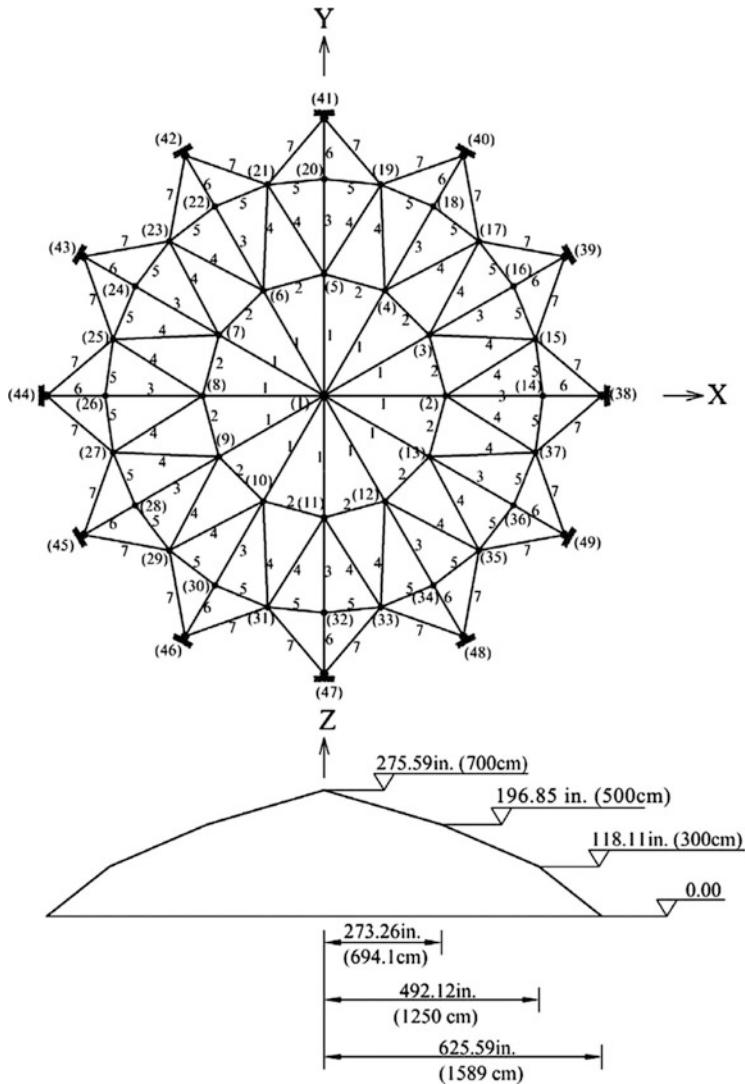
The modulus of elasticity and the material density are taken as 30,450 ksi (210 GPa) and 0.288 lb/in<sup>3</sup>, respectively. The yield stress is taken as 58.0 ksi (400 MPa). The radius of gyration is expressed in terms of cross-sectional areas of the members as  $r_i = aA_i^b$ . Constants  $a$  and  $b$  depend on the types of sections adopted for the members such as pipes, angles, etc. In this example pipe sections are used for the bars for which  $a = 0.4993$  and  $b = 0.6777$ . The dome is considered to be subjected to vertical loads at all unsupported nodes. These vertical 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 other nodes. Four different problem variants are considered for this structure: with stress constraints and no displacement constraints (Case 1), with stress constraints and displacement limitations of  $\pm 0.1969$  in (5 mm) imposed on all nodes in  $x$  and  $y$  directions (Case 2), no stress constraints and displacement limitations of  $\pm 0.1969$  in (5 mm) imposed on all nodes in  $z$  direction

**Table 15.12** Comparison of the optimization results obtained in the spatial 72-bar truss problem

Element group	Optimal cross-sectional areas ( $\text{in}^2$ )					
	Erbatur et al. [18]	Camp and Bichon [19]	Perez and Behdinan [20]	Camp [21]	Kaveh and Khayatazad [22]	TWO [1]
1–4	1.755	1.948	1.7427	1.8577	1.8365	1.9961
5–12	0.505	0.508	0.5185	0.5059	0.5021	0.5100
13–16	0.105	0.101	0.1000	0.1000	0.1000	0.1000
17–18	0.155	0.102	0.1000	0.1000	0.1004	0.1000
19–22	1.155	1.303	1.3079	1.2476	1.2522	1.2434
23–30	0.585	0.511	0.5193	0.5269	0.5033	0.5184
31–34	0.100	0.101	0.1000	0.1000	0.1002	0.1000
35–36	0.100	0.100	0.1000	0.1012	0.1001	0.1001
37–40	0.460	0.561	0.5142	0.5209	0.5730	0.5211
41–48	0.530	0.492	0.5464	0.5172	0.5499	0.5098
49–52	0.120	0.100	0.1000	0.1004	0.1004	0.1000
53–54	0.165	0.107	0.1095	0.1005	0.1001	0.1000
55–58	0.155	0.156	0.1615	0.1565	0.1576	0.1569
59–66	0.535	0.550	0.5092	0.5507	0.5222	0.5346
67–70	0.480	0.390	0.4967	0.3922	0.4356	0.3959
71–72	0.520	0.592	0.5619	0.5922	0.5971	0.5821
Best weight (lb)	385.76	380.24	381.91	379.85	380.458	379.846
Mean weight (lb)	N/A	383.16	N/A	382.08	382.553	381.976
Std dev (lb)	N/A	3.66	N/A	1.912	1.221	3.161
No. of analyses	N/A	18,500	N/A	19,621	19,084	16,000

(Case 3), and all the abovementioned constraints imposed together (Case 4). For Cases 1 and 2, the maximum cross-sectional area is taken as  $5.0 \text{ in}^2$  ( $32.26 \text{ cm}^2$ ), while for Cases 3 and 4, it is taken as  $20 \text{ in}^2$  ( $129.03 \text{ cm}^2$ ). The minimum cross-sectional area is taken as  $0.775 \text{ in}^2$  ( $5 \text{ cm}^2$ ) for all cases.

Table 15.13 compares the results obtained by different optimization techniques for this example. It can be seen that the results found by TWO are comparable to those of other methods. In Case 1 the best result obtained by TWO is the same as that of CBO which is the best result so far. In Cases 3 and 4, the results obtained by TWO are better than those of RO, IRO, and CBO and are only slightly heavier than that of HPSACO (0.01 and 0.004 % in Cases 3 and 4, respectively). The average number of structural analyses required by the RO and IRO algorithms was reported as 19,900 and 18,300, respectively, while TWO uses 40 candidate solutions and



**Fig. 15.7** Schematic of the 120-bar dome truss structure

400 iterations like CBO resulting in a maximum number of analyses of 16,000. HPSACO and HS, respectively, performed 10,000 and 35,000 structural analyses to obtain their optimal results. It should be noted that HPSACO is a hybrid method which combines good features of PSO, ACO, and HS.

**Table 15.13** Comparison of the optimization results obtained in the 120-bar dome problem

Element group	Optimal cross-sectional areas (in <sup>2</sup> )	Case 1				Case 2			
		HPSACO (Kaveh and Talatahari [23])	RO (Kaveh and Khayatazzad [22])	CBO (Kaveh and Mahdavi [12])	TWO [1]	HS (Lee and Geem [17])	Geem [17])	HPSACO (Kaveh and Talatahari [23])	RO (Kaveh and Khayatazzad [22])
1	3.295	3.311	3.128	3.1229	3.1229	3.296	3.779	3.084	3.0832
2	3.396	3.438	3.357	3.3538	3.3538	2.789	3.377	3.360	3.3526
3	3.874	4.147	4.114	4.1120	4.1120	3.872	4.125	4.093	4.0928
4	2.571	2.831	2.783	2.7822	2.7822	2.570	2.734	2.762	2.7613
5	1.150	0.775	0.775	0.7750	0.7750	1.149	1.609	1.593	1.5918
6	3.331	3.474	3.302	3.3005	3.3005	3.331	3.533	3.294	3.2927
7	2.784	2.551	2.453	2.4458	2.4458	2.781	2.539	2.434	2.4336
Best weight (lb)	19707.77	19491.3	19476.193	19454.7	19454.67	19893.34	20078.0	20071.9	20064.5
Average weight (lb)	—	—	—	19466.0	19454.98	—	—	—	20098.3
Std dev (lb)	—	—	33.966	7.02	1.17	—	—	112.135	26.17
									116.15

(continued)

Table 15.13 (continued)

Element group	Optimal cross-sectional areas (in <sup>2</sup> )	Case 4							
		HPSACO (Kaveh and Tatatahri [23])	RO (Kaveh and Khayatazzad [22])	CBO (Kaveh and Mahdavi [12])	Present work [1]	HPSACO (Kaveh and Tatatahri [23])	RO (Kaveh and Khayatazzad [22])	IRO (Kaveh et al. [24])	CBO (Kaveh and Mahdavi [12])
1	2.034	2.044	2.0660	1.9667	3.095	3.030	3.0252	3.0273	3.0247
2	15.151	15.665	15.9200	15.3920	14.405	14.806	14.8354	15.1724	14.7261
3	5.901	5.848	5.6785	5.7127	5.020	5.440	5.1139	5.2342	5.1338
4	2.254	2.290	2.2987	2.1960	3.352	3.124	3.1305	3.119	3.1369
5	9.369	9.001	9.0581	9.5439	8.631	8.021	8.4037	8.1038	8.4545
6	3.744	3.673	3.6365	3.6688	3.432	3.614	3.3315	3.4166	3.2946
7	2.104	1.971	1.9320	1.9351	2.499	2.487	2.4968	2.4918	2.4956
Best weight (lb)	31670.0	31733.2	31724.1	31673.62	33248.9	33317.8	33256.48	33286.3	33250.31
Average weight (lb)	—	—	32162.4	31680.34	—	—	—	33398.5	33282.64
Std dev (lb)	—	274.991	240.22	6.15	—	354.333	—	67.09	25.38

### 15.4.2 Truss Weight Optimization with Dynamic Constraints

In this section weight minimization of truss structures with dynamic constraints using TWO is presented, where some of the natural frequencies of the structure are upper/lower bounded. Four numerical examples are provided in this section in order to examine the performance of TWO on frequency constraint weight minimization of truss structures. The results are compared to those of some other optimization techniques reported in the literature. A total population of 20 particles is considered for all of the examples except for the second example where 30 agents are used.

#### 15.4.2.1 A 10-Bar Truss

Size optimization of a 10-bar truss structure shown in Fig. 15.8 is considered as the first dynamic constraint example. This is a well-known benchmark problem in the field of structural optimization subjected to frequency constraints. Cross-sectional areas of all ten members are assumed to be independent variables. A nonstructural mass of 454.0 kg is attached to all free nodes. Table 15.14 summarizes the material properties, variable bounds, and frequency constraints for this example.

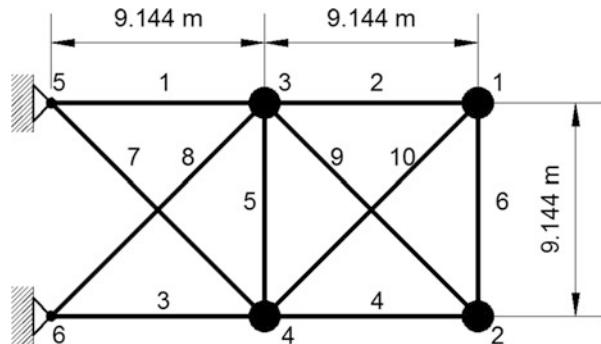
This problem is addressed by different researchers using a wide variety of methods: Grandhi and Venkayya [25] using an optimality algorithm, Sedaghati et al. [26] utilizing a sequential quadratic programming and finite element force method, Wang et al. [27] using an evolutionary node shift method, Lingyun et al. [28] utilizing a niche hybrid genetic algorithm, Gomes employing particle swarm optimization algorithm [29], and Kaveh and Zolghadr employing standard and enhanced CSS [30], hybridized CSS–BBC with trap recognition capability [31], democratic particle swarm optimization (DPSO) [32], and a hybridized PSRO [33].

Optimal structures found by different methods and the corresponding masses are summarized in Table 15.15. The optimal structure found by TWO is better than other methods and is only 0.05 % heavier than that of DPSO. It should be noted that the structures found by standard PSO [29] and CSS [30] are obtained using a modulus of elasticity of  $E = 6.98 \times 10^{10}$  N/m<sup>2</sup>, which generally results in lighter structures. Table 15.16 presents the natural frequencies of the optimized structures obtained by different methods. It can be seen that all constraints are satisfied. The mean value and the standard deviation of 50 independent runs of TWO are 539.28 kg and 3.28, respectively. Figure 15.9 presents the convergence curve of the best run of TWO for the 10-bar planar truss.

#### 15.4.2.2 A 72-Bar Spatial Truss

A 72-bar spatial truss as depicted in Fig. 15.10 is presented as the second example. Four nonstructural masses of 2270 kg are attached to the uppermost four nodes. The shape of the structure is kept unchanged during the optimization process, and the

**Fig. 15.8** Schematic of a 10-bar planar truss structure



**Table 15.14** Material properties, variable bounds, and frequency constraints for the 10-bar truss structure

Property/unit	Value
E (modulus of elasticity)/N/m <sup>2</sup>	$6.89 \times 10^{10}$
$\rho$ (material density)/kg/m <sup>3</sup>	2770.0
Added mass/kg	454.0
Design variable lower bound/m <sup>2</sup>	$0.645 \times 10^{-4}$
Design variable upper bound/m <sup>2</sup>	$50 \times 10^{-4}$
L (main bar's dimension)/m	9.144
Constraints on the first three frequencies/Hz	$\omega_1 \geq 7, \omega_2 \geq 15, \omega_3 \geq 20$

design variables only include cross-sectional areas of the members, which are grouped into 16 groups. Material properties, variable bounds, frequency constraints, and added masses are listed in Table 15.17.

Optimized designs obtained by different optimization methods are summarized in Table 15.18. It should be mentioned that a modulus of elasticity of  $E = 6.98 \times 10^{10}$  N/m<sup>2</sup> is used by Gomes [29] and Kaveh and Zolghadr [30, 31]. This generally results in lighter structures. The mean value and the standard deviation of 50 independent runs of TWO are 336.1 kg and 5.8, respectively. Table 15.19 presents the first five natural frequencies for the optimized structures found by different methods. It could be seen that all of the frequency constraints are satisfied. Figure 15.11 presents the convergence curve for the best run of TWO.

#### 15.4.2.3 A Simply Supported 37-Bar Planar Truss

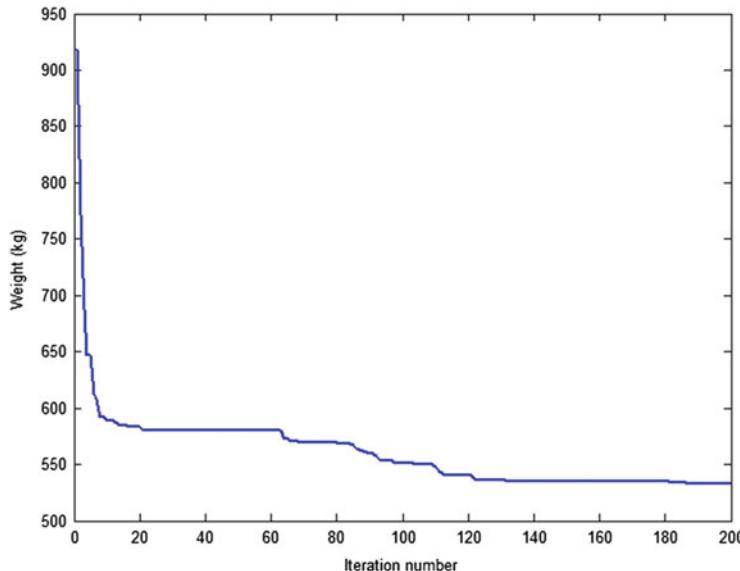
Shape and size optimization of a simply supported 37-bar planar truss as shown in Fig. 15.12 is studied as the third example. The elements of the lower chord are modeled as bar elements with constant rectangular cross-sectional areas of  $4 \times 10^{-3}$  m<sup>2</sup>. The rest of the members, which are modeled as bar elements, are grouped considering the symmetry. The y-coordinate of all the nodes on the upper chord can

**Table 15.15** Optimal structures ( $\text{cm}^2$ ) found by different methods for the planar 10-bar planar truss problem (the optimized weight does not include the added masses)

Element number	Grandhi and Venkayya [25]	Sedghati et al. [26]	Wang et al. [27]	Lingyun et al. [28]	Gomes [29]	Kaveh and Zolghadr		
						Standard CSS [30]	DPSO [32]	PSRO [33]
1	36.584	38.245	32.456	42.23	37.712	38.811	35.944	37.075
2	24.658	9.916	16.577	18.555	9.959	9.0307	15.530	15.334
3	36.584	38.619	32.456	38.851	40.265	37.099	35.285	33.665
4	24.658	18.232	16.577	11.222	16.788	18.479	15.385	14.849
5	4.167	4.419	2.115	4.783	11.576	4.479	0.648	0.645
6	2.070	4.419	4.467	4.451	3.955	4.205	4.583	4.643
7	27.032	20.097	22.810	21.049	25.308	20.842	23.610	24.528
8	27.032	24.097	22.810	20.949	21.613	23.023	23.599	23.188
9	10.346	13.890	17.490	10.257	11.576	13.763	13.135	12.436
10	10.346	11.452	17.490	14.342	11.186	11.414	12.357	13.500
Weight (kg)	594.0	537.01	553.8	542.75	537.98	531.95	532.39	532.85

**Table 15.16** Natural frequencies (Hz) of the optimized designs for the 10-bar planar truss

Frequency number	Grandhi and Venkayya [25]	Sedaghati et al. [26]	Wang et al. [27]	Lingyun et al. [28]	Gomes [29]	Kavvah and Zolghadr Standard CSS [30]	DPSO [31]	PSRO [32]	TWO [2]
1	7.059	6.992	7.011	7.008	7.000	7.000	7.000	7.000	7.001
2	15.895	17.599	17.302	18.148	17.786	17.442	16.187	16.143	16.185
3	20.425	19.973	20.001	20.000	20.000	20.031	20.000	20.000	20.001
4	21.528	19.977	20.100	20.508	20.063	20.208	20.021	20.032	20.018
5	28.978	28.173	30.869	27.797	27.776	28.261	28.470	28.469	28.572
6	30.189	31.029	32.666	31.281	30.939	31.139	29.243	29.485	29.265
7	54.286	47.628	48.282	48.304	47.297	47.704	48.769	48.440	48.560
8	56.546	52.292	52.306	53.306	52.286	52.420	51.389	51.157	51.290



**Fig. 15.9** Convergence curve of the best run of TWO for the 10-bar planar truss

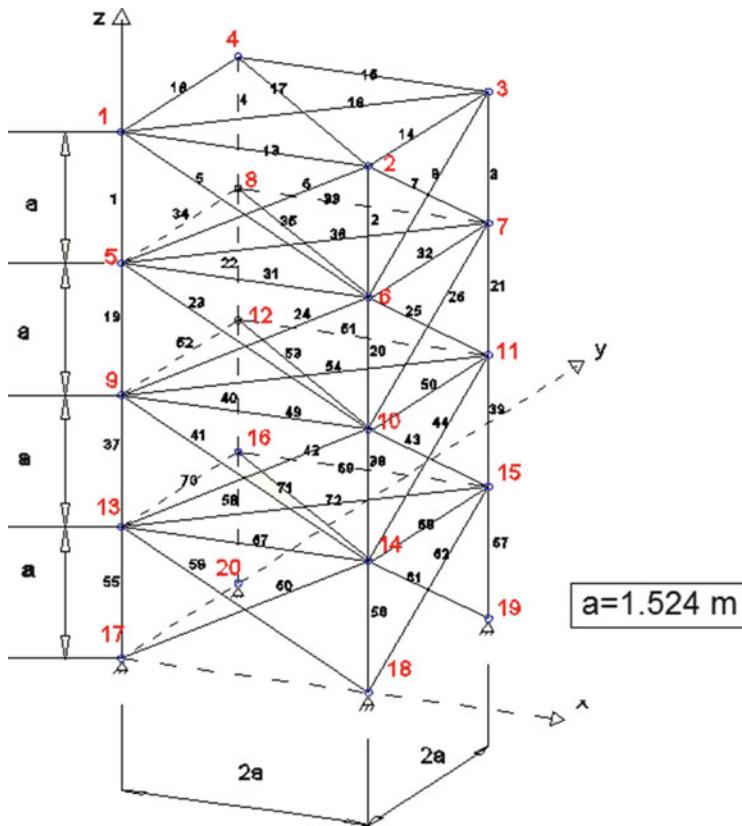
vary in a symmetrical manner to form the shape variables. A nonstructural mass of 10 kg is attached to all free nodes of the lower chord. Constraints are imposed on the first three natural frequencies of the structure.

This example has been investigated by different researchers including Wang et al. [27] using an evolutionary node shift method, Lingyun et al. [28] using a niche hybrid genetic algorithm, Gomes [29] utilizing particle swarm algorithm, and Kaveh and Zolghadr using CSS [30], democratic PSO [32], and a hybridized PSRO algorithm [33]. Material properties, frequency constraints, added masses, and variable bounds for this example are listed in Table 15.20. Final cross-sectional areas and node coordinates obtained by different methods together with the corresponding weights are shown in Table 15.21.

Table 15.21 shows that TWO has obtained the best result among the algorithms. The mean weight and the standard deviation of 50 independent runs of TWO are 363.75 kg and 2.48 kg, respectively. Table 15.22 presents the first five natural frequencies of the optimized structures. Figure 15.13 presents the convergence curve of the best run of TWO for the simply supported 37-bar planar truss.

#### 15.4.2.4 A 52-Bar Dome-Like Truss

As the last example, simultaneous shape and size optimization of a 52-bar dome-like truss is considered. The initial layout of the structure is depicted in Fig. 15.14. Nonstructural masses of 50 kg are attached to all free nodes. Material properties, frequency constraints, and variable bounds for this example are summarized in



**Fig. 15.10** The 72-bar spatial truss

**Table 15.17** Material properties and frequency constraints for the 72-bar spatial truss

Property/unit	Value
E (modulus of elasticity)/N/m <sup>2</sup>	$6.89 \times 10^{10}$
$\rho$ (material density)/kg/m <sup>3</sup>	2770.0
Added mass/kg	2270
Design variable lower bound/m <sup>2</sup>	$0.645 \times 10^{-4}$
Constraints on the first three frequencies/Hz	$\omega_1 = 4.0, \omega_3 \geq 6$

Table 15.23. The elements of the structure are categorized in eight groups according to Table 15.24. All free nodes are permitted to move  $\pm 2$  m from their initial position in a symmetrical manner.

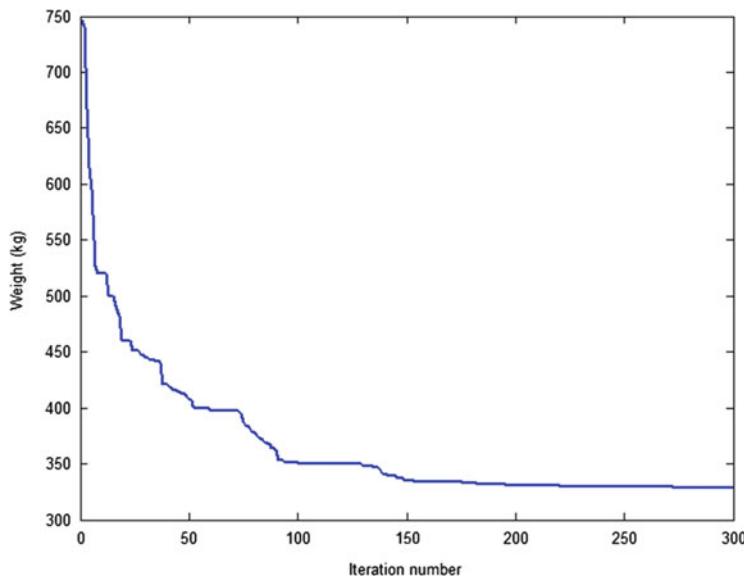
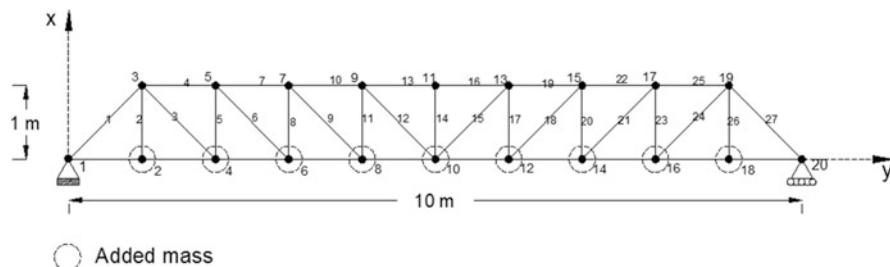
This example has been solved by Lin et al. [34] using a mathematical programming technique and Lingyun et al. [28] using a niche hybrid genetic algorithm. Gomes [29] has studied the problem using particle swarm optimization algorithm. The authors have studied the problem using CSS [30], hybridized CSS–BBC with

Table 15.18 Optimal cross-sectional areas for the 72-bar space truss ( $\text{cm}^2$ )

Group number	Elements	Sedaghati [26]	Gomes [29]	Kaveh and Zolghadr				
				Standard CSS[30]	Enhanced CSS[31]	CSS-BBBC[32]	PSRO[33]	TWO[2]
1	1–4	3.499	2.987	2.528	2.252	2.854	3.840	3.380
2	5–12	7.932	7.849	8.704	9.109	8.301	8.360	8.086
3	13–16	0.645	0.645	0.645	0.648	0.645	0.645	0.647
4	17–18	0.645	0.645	0.645	0.645	0.645	0.699	0.646
5	19–22	8.056	8.765	8.283	7.946	8.202	8.817	8.890
6	23–30	8.011	8.153	7.888	7.703	7.043	7.697	8.136
7	31–34	0.645	0.645	0.645	0.647	0.645	0.645	0.654
8	35–36	0.645	0.645	0.645	0.646	0.645	0.651	0.647
9	37–40	12.812	13.450	14.666	13.465	16.328	12.136	13.097
10	41–48	8.061	8.073	6.793	8.250	8.299	8.839	8.101
11	49–52	0.645	0.645	0.645	0.645	0.645	0.645	0.663
12	53–54	0.645	0.645	0.645	0.646	0.645	0.645	0.646
13	55–58	17.279	16.684	16.464	18.368	15.048	17.059	16.483
14	59–66	8.088	8.159	8.809	7.053	8.268	7.427	7.873
15	67–70	0.645	0.645	0.645	0.645	0.645	0.646	0.651
16	71–72	0.645	0.645	0.645	0.646	0.645	0.645	0.657
	Weight (kg)	327.605	328.823	328.814	328.393	327.507	329.80	328.83

**Table 15.19** Natural frequencies (Hz) obtained by different methods for the 72-bar space truss

Frequency number	Sedaghati [26]	Gomes [29]	Kaveh and Zolghadr				
			Standard CSS [30]	Enhanced CSS [31]	CSS–BBBC [32]	PSRO [33]	TWO [2]
1	4.000	4.000	4.000	4.000	4.000	4.000	4.000
2	4.000	4.000	4.000	4.000	4.000	4.000	4.000
3	6.000	6.000	6.006	6.004	6.004	6.000	6.000
4	6.247	6.219	6.210	6.155	6.2491	6.418	6.259
5	9.074	8.976	8.684	8.390	8.9726	9.143	9.082

**Fig. 15.11** Convergence curve of the best run of TWO for the 72-bar planar truss**Fig. 15.12** A simply supported planar 37-bar truss

**Table 15.20** Material properties, frequency constraints, and variable bounds for the simply supported 37-bar planar truss

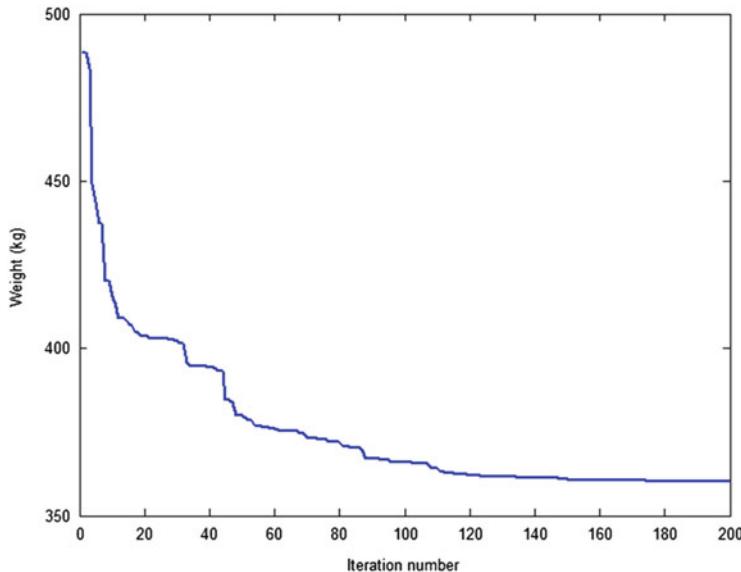
Property/unit	Value
E (modulus of elasticity)/N/m <sup>2</sup>	$2.1 \times 10^{11}$
$\rho$ (material density)/kg/m <sup>3</sup>	7800
Design variable lower bound/m <sup>2</sup>	$1 \times 10^{-4}$
Design variable upper bound/m <sup>2</sup>	$10 \times 10^{-4}$
Added mass/kg	10
Constraints on the first three frequencies/Hz	$\omega_1 \geq 20, \omega_2 \geq 40, \omega_3 \geq 60$

**Table 15.21** Optimized designs obtained for the planar 37-bar truss problem

Variable	Wang et al. [27]	Lingyun et al. [28]	Gomes [29]	Kaveh and Zolghadr			
				Standard CSS [30]	DPSO [32]	PSRO [33]	TWO [2]
Y3, Y19 (m)	1.2086	1.1998	0.9637	0.8726	0.9482	1.0087	1.0039
Y5, Y17 (m)	1.5788	1.6553	1.3978	1.2129	1.3439	1.3985	1.3531
Y7, Y15 (m)	1.6719	1.9652	1.5929	1.3826	1.5043	1.5344	1.5339
Y9, Y13 (m)	1.7703	2.0737	1.8812	1.4706	1.6350	1.6684	1.6768
Y11 (m)	1.8502	2.3050	2.0856	1.5683	1.7182	1.7137	1.7728
A1, A27 (cm <sup>2</sup> )	3.2508	2.8932	2.6797	2.9082	2.6208	2.6368	2.8892
A2, A26 (cm <sup>2</sup> )	1.2364	1.1201	1.1568	1.0212	1.0397	1.3034	1.0949
A3, A24 (cm <sup>2</sup> )	1.0000	1.0000	2.3476	1.0363	1.0464	1.0029	1.0213
A4, A25 (cm <sup>2</sup> )	2.5386	1.8655	1.7182	3.9147	2.7163	2.3325	2.6776
A5, A23 (cm <sup>2</sup> )	1.3714	1.5962	1.2751	1.0025	1.0252	1.2868	1.1981
A6, A21 (cm <sup>2</sup> )	1.3681	1.2642	1.4819	1.2167	1.5081	1.0704	1.1387
A7, A22 (cm <sup>2</sup> )	2.4290	1.8254	4.6850	2.7146	2.3750	2.4442	2.6537
A8, A20 (cm <sup>2</sup> )	1.6522	2.0009	1.1246	1.2663	1.4498	1.3416	1.4171
A9, A18 (cm <sup>2</sup> )	1.8257	1.9526	2.1214	1.8006	1.4499	1.5724	1.3934
A10, A19 (cm <sup>2</sup> )	2.3022	1.9705	3.8600	4.0274	2.5327	3.1202	2.7741
A11, A17 (cm <sup>2</sup> )	1.3103	1.8294	2.9817	1.3364	1.2358	1.2143	1.2759
A12, A15 (cm <sup>2</sup> )	1.4067	1.2358	1.2021	1.0548	1.3528	1.2954	1.2776
A13, A16 (cm <sup>2</sup> )	2.1896	1.4049	1.2563	2.8116	2.9144	2.7997	2.1666
A14 (cm <sup>2</sup> )	1.0000	1.0000	3.3276	1.1702	1.0085	1.0063	1.0099
Weight (kg)	366.50	368.84	377.20	362.84	360.40	360.97	360.27

**Table 15.22** Natural frequencies (Hz) evaluated at the optimized designs for the planar 37-bar truss

Frequency number	Wang et al. [27]	Lingyun et al. [28]	Gomes [29]	Kaveh and Zolghadr			
				Standard CSS [30]	DPSO [32]	PSRO [33]	TWO [2]
1	20.0850	20.0013	20.0001	20.0000	20.0194	20.1023	20.0279
2	42.0743	40.0305	40.0003	40.0693	40.0113	40.0804	40.0146
3	62.9383	60.0000	60.0001	60.6982	60.0082	60.0516	60.0946
4	74.4539	73.0444	73.0440	75.7339	76.9896	75.8918	76.5062
5	90.0576	89.8244	89.8240	97.6137	97.2222	97.2470	96.5840



**Fig. 15.13** Convergence curve of the best run of TWO for the simply supported 37-bar planar truss

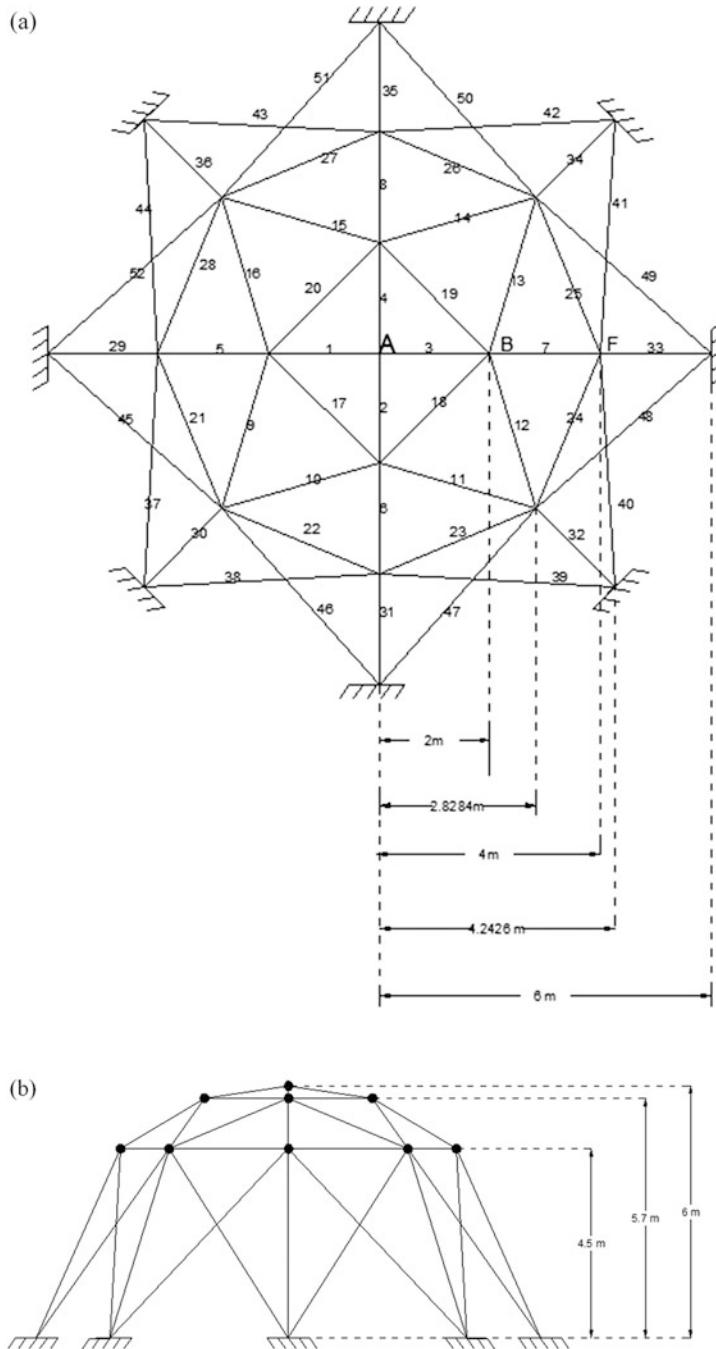
a trap recognition capability [31], democratic PSO [32], and a hybridized PSRO algorithm [33].

Table 15.25 summarizes the best results obtained by different methods for this example. It can be seen that the structure found by TWO is lighter than those of other methods. The mean weight and the standard deviation of 50 independent runs of TWO are 214.25 kg and 12.64 kg, respectively. Table 15.26 shows the first five natural frequencies of the final structures found by various methods for the 52-bar dome-like space truss. The convergence curve of the best run of TWO for this problem is shown in Fig. 15.15.

## 15.5 Concluding Remarks

A newly proposed metaheuristic algorithm named tug of war optimization (TWO) [1] is presented and utilized for engineering optimization in this chapter. The algorithm considers each of the candidate solutions as a team competing in a series of rope-pulling competitions.

An idealized framework is presented in order to simplify the physical nature of a game of tug of war, in which the teams are considered as two bodies lying on a smooth surface. It is then assumed that the pulling force that a team can exert is proportional to its weight, and the two teams sustain their grip of the rope during the



**Fig. 15.14** Initial layout of the spatial 52-bar truss. (a) Top view. (b) Side view

**Table 15.23** Material properties, frequency constraints, and variable bounds for the 52-bar space truss

Property/unit	Value
E (modulus of elasticity)/N/m <sup>2</sup>	$2.1 \times 10^{11}$
$\rho$ (material density)/kg/m <sup>3</sup>	7800
Added mass/kg	50
Allowable range for cross sections/m <sup>2</sup>	$0.0001 \leq A \leq 0.001$
Constraints on the first three frequencies/Hz	$\omega_1 \leq 15.916$ $\omega_2 \geq 28.648$

**Table 15.24** Element grouping for the spatial 52-bar truss

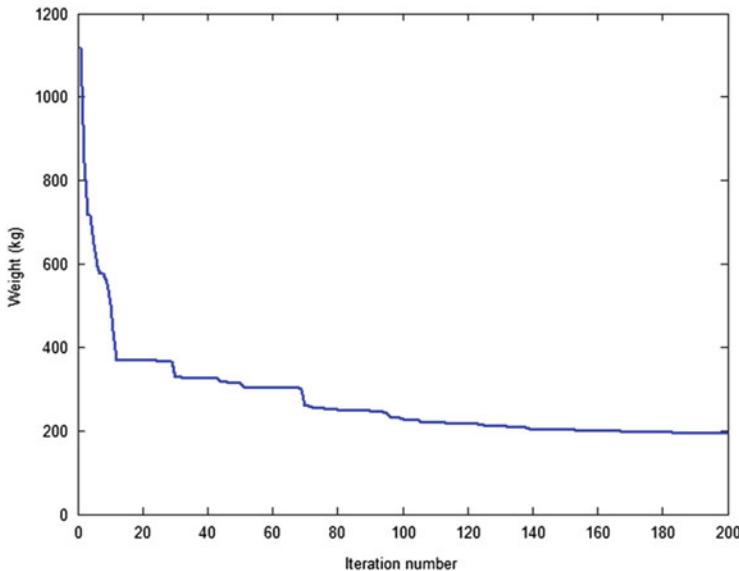
Group number	Elements
1	1–4
2	5–8
3	9–16
4	17–20
5	21–28
6	29–36
7	37–44
8	45–52

**Table 15.25** Optimized designs obtained for the spatial 52-bar truss problem

Variable	Lin et al. [34]	Lingyun et al. [28]	Gomes [29]	Kaveh and Zolghadr			
				Standard CSS [30]	DPSO [32]	PSRO [33]	TWO [2]
Z <sub>A</sub> (m)	4.3201	5.8851	5.5344	5.2716	6.1123	6.252	6.012
X <sub>B</sub> (m)	1.3153	1.7623	2.0885	1.5909	2.2343	2.456	1.598
Z <sub>B</sub> (m)	4.1740	4.4091	3.9283	3.7093	3.8321	3.826	4.287
X <sub>F</sub> (m)	2.9169	3.4406	4.0255	3.5595	4.0316	4.179	3.641
Z <sub>F</sub> (m)	3.2676	3.1874	2.4575	2.5757	2.5036	2.501	2.888
A1 (cm <sup>2</sup> )	1.00	1.0000	0.3696	1.0464	1.0001	1.0007	2.1245
A2 (cm <sup>2</sup> )	1.33	2.1417	4.1912	1.7295	1.1397	1.0312	1.1341
A3 (cm <sup>2</sup> )	1.58	1.4858	1.5123	1.6507	1.2263	1.2403	1.1870
A4 (cm <sup>2</sup> )	1.00	1.4018	1.5620	1.5059	1.3335	1.3355	1.3180
A5 (cm <sup>2</sup> )	1.71	1.911	1.9154	1.7210	1.4161	1.5713	1.3637
A6 (cm <sup>2</sup> )	1.54	1.0109	1.1315	1.0020	1.0001	1.0021	1.0299
A7 (cm <sup>2</sup> )	2.65	1.4693	1.8233	1.7415	1.5750	1.3267	1.3479
A8 (cm <sup>2</sup> )	2.87	2.1411	1.0904	1.2555	1.4357	1.5653	1.4446
Weight (kg)	298.0	236.046	228.381	205.237	195.351	197.186	194.25

**Table 15.26** Natural frequencies of the optimized designs obtained for the spatial 52-bar truss problem

Frequency number	Lin et al. [36]	Lingyun et al. [28]	Gomes [29]	Kaveh and Zolghadr			
				Standard CSS [30]	DPSO [32]	PSRO [33]	TWO [2]
1	15.22	12.81	12.751	9.246	11.315	12.311	9.265
2	29.28	28.65	28.649	28.648	28.648	28.648	28.667
3	29.28	28.65	28.649	28.699	28.648	28.649	28.667
4	31.68	29.54	28.803	28.735	28.650	28.715	28.686
5	33.15	30.24	29.230	29.223	28.688	28.744	29.734

**Fig. 15.15** Convergence curve of the best run of TWO for the 52-bar truss problem

contest. The weights of the teams are determined based on the quality of the solutions they represent.

Different numerical examples from various fields of structural optimization are investigated in order to show the viability and efficiency of TWO. Examples include truss weight minimization with static and dynamic constraints. Numerical results demonstrate that the proposed algorithm is a competent one, and its behavior is comparable to other state-of-the-art metaheuristic algorithms.

## References

1. Kaveh A, Zolghadr A (2016) Tug of War Optimization: a new metaheuristic algorithm. *Int J Optim Civil Eng* 6(4):469–493
2. Kaveh A, Zolghadr A (2016) Truss shape and size optimization with frequency constraints using Tug of war optimization
3. Tsoulos IG (2008) Modifications of real code genetic algorithm for global optimization. *Appl Math Comput* 203:598–607
4. 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
5. Arora JS (1989) Introduction to optimum design. McGraw-Hill, New York, NY
6. Coello CAC (2000) Use of a self-adaptive penalty approach for engineering optimization problems. *Comput Indust* 41:113–127
7. Coello CAC, Montes EM (2002) Constraint-handling in genetic algorithms through the use of dominance-based tournament selection. *Adv Eng Inform* 16:193–203
8. He Q, Wang L (2007) An effective co-evolutionary particle swarm optimization for constrained engineering design problems. *Eng Appl Artific Intell* 20:89–99
9. Montes EM, Coello CAC (2008) An empirical study about the usefulness of evolution strategies to solve constrained optimization problems. *Int J General Sys* 37(4):443–473
10. Kaveh A, Talatahari S (2010) An improved ant colony optimization for constrained engineering design problems. *Eng Comput* 27(1):155–182
11. Kaveh A, Talatahari S (2010) A novel heuristic optimization method: charged system search. *Acta Mech* 213:267–289
12. Kaveh A, Mahdavi VR (2014) Colliding bodies optimization: a novel metaheuristic method. *Comput Struct* 139:18–27
13. Ragsdell KM, Phillips DT (1976) Optimal design of a class of welded structures using geometric programming. *ASME J Eng Indust Ser B* 98(3):1021–1025
14. Deb K (1991) Optimal design of a welded beam via genetic algorithms. *AIAA J* 29 (11):2013–2015
15. Rajeev S, Krishnamoorthy CS (1992) Discrete optimization of structures using genetic algorithms. *ASCE J Struct Eng* 118:1233–1250
16. Schutte JJ, Groenwold AA (2003) Sizing design of truss structures using particle swarms. *Struct Multidisc Optim* 25:261–269
17. Lee KS, Geem ZW (2004) A new structural optimization method based on the harmony search algorithm. *Comput Struct* 82:781–798
18. Erbatur F, Hasançebi O, Tütüncü I, Kılıç H (2000) Optimal design of planar and space structures with genetic algorithms. *Comput Struct* 75:209–224
19. Camp CV, Bichon J (2004) Design of space trusses using ant colony optimization. *ASCE J Struct Eng* 130:741–751
20. Perez RE, Behdinan K (2007) Particle swarm approach for structural design optimization. *Comput Struct* 85:1579–1588
21. Camp CV (2007) Design of space trusses using Big Bang–Big Crunch optimization. *ASCE J Struct Eng* 133:999–1008
22. Kaveh A, Khayatazad M (2012) A novel metaheuristic method: ray optimization. *Comput Struct* 112–113:283–294
23. 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
24. Kaveh A, Ilchi Ghazaan M, Bakhspoori T (2013) An improved ray optimization algorithm for design of truss structures. *Period Polytech Civil Eng* 57:97–112
25. Grandhi RV, Venkayya VB (1988) Structural optimization with frequency constraints. *AIAA J* 26(7):858–866

26. Sedaghati R, Suleman A, Tabarrok B (2002) Structural optimization with frequency constraints using finite element force method. *AIAA J* 40(2):382–388
27. Wang D, Zha WH, Jiang JS (2004) Truss optimization on shape and sizing with frequency constraints. *AIAA J* 42:1452–1456
28. 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* 35(5):361–368
29. Gomes MH (2011) Truss optimization with dynamic constraints using a particle swarm algorithm. *Exp Syst Appl* 38(1):957–968
30. 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
31. 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
32. Kaveh A, Zolghadr A (2014) Democratic PSO for truss layout and size optimization with frequency constraints. *Comput Struct* 130:10–21
33. Kaveh A, Zolghadr A (2014) A new PSRO algorithm for frequency constraint truss shape and size optimization. *Struct Eng Mech* 52:445–468
34. Lin JH, Chen WY, Yu YS (1982) Structural optimization on geometrical configuration and element sizing with static and dynamic constraints. *Comput Struct* 15(5):507–515

# Chapter 16

## Water Evaporation Optimization Algorithm

### 16.1 Introduction

Efficient metaheuristic optimization algorithms are developed to overcome the drawbacks of some traditional methods in highly nonlinear engineering optimization problems with high complexity, high dimension, and multimodal design spaces gaining increasing popularity nowadays [1]. Performance assessment of a metaheuristic algorithm may be used by solution quality, computational effort, and robustness [2] directly affected by its two contradictory criteria: exploration of the search space (diversification) and exploitation of the best solutions found (intensification).

A novel metaheuristic algorithm, imitating the evaporation process of a tiny amount of water molecules adhered on a solid surface with different wettability, has been developed very recently and named water evaporation optimization (WEO) [3]. WEO was successfully utilized in real parameter optimization [3] and continuous structural optimization problems [4] presenting: (1) competitive behavior with other algorithms in terms of accuracy and robustness, (2) advantages over other algorithms in the aspect of parameter tuning (except population size, other parameters are set based on the molecular dynamics), and (3) imposing a rational number of physical rules that lead to significantly good convergence behavior and simple algorithmic structure. In structural optimization problems, high computational cost of the algorithm has been found to be the only drawback of the algorithm. In view of this, the present study will propose an accelerated version of WEO potentially able to solve multidisciplinary and engineering optimization problems regardless of the type of optimization problem at hand.

The basic WEO is initialized with a population of random designs named water molecules that are updated through an iterative process to search the optimum. Updating is governed by evaporation rules based on molecular dynamic simulation results obtained for evaporation process of water molecules from a solid surface with different wettability. The candidate solutions are updated globally and locally

in two independent sequential phases: monolayer and droplet evaporation phases. Rules driving each phase are in a good agreement with the local and global search ability of the algorithm. In this study, it is shown that the WEO can be improved to reduce computational cost by simultaneous utilization of both phases.

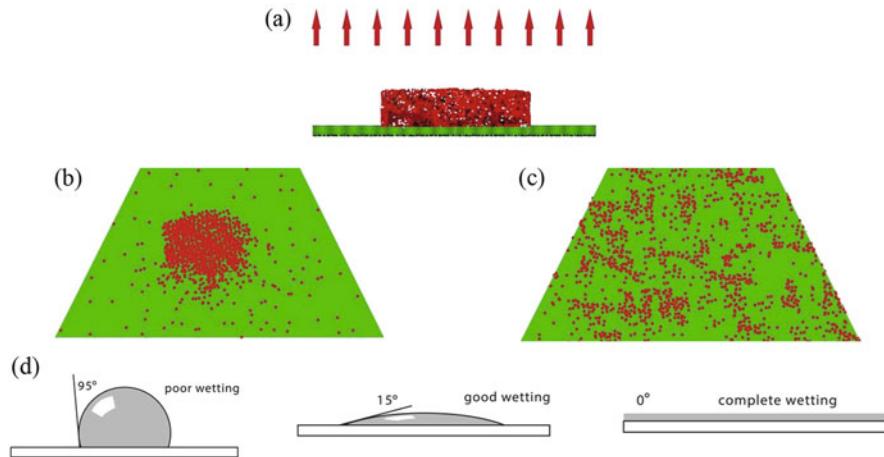
The improvement to WEO hence involves a simultaneous use of both phases. Two scenarios are developed for considering simultaneous evaporation phases based on (1) water molecules' distances and (2) objective function values of molecules. The effects of these enhancements are tested in very classical discrete structural optimization problems. The first scenario which is developed from the physics of water evaporation phenomena is efficient in enhancing the convergence rate. Although the second scenario (which is developed greedily) also is efficient, the first scenario is accepted and developed in this study in view of its physical interpretation and higher performance.

This chapter is based on the accelerated version of the water evaporation optimization algorithm developed by Kaveh and Bakhshpoori [5], and it is organized as follows: Section 16.2 outlines the basic WEO algorithm. Section 16.3 deals with the development of accelerated version of WEO with mixed phases. Section 16.4 discusses the performance of the new formulation of WEO in discrete structural optimization problems. Finally, conclusions are given in Sect. 16.5.

## 16.2 Basic Water Evaporation Optimization Algorithm

Evaporation of water restricted on the surface of solid materials is the inspiration basis of WEO which is different from the water evaporation of bulk surface. This type of water evaporation is essential in the macroscopic world such as the water loss through the surface of soil [6]. Wang et al. [7] presented molecular dynamics (MD) simulations on the evaporation of water from a solid substrate with different surface wettability. MD simulations were carried out by adhering nanoscale water aggregation in a neutral substrate which is chargeable. By varying the value of charge ( $0 \text{ e} \leq q \leq 0.7 \text{ e}$ ), a substrate with tunable surface wettability can be obtained. It is found that as the surface changed from hydrophobicity ( $q < 0.4 \text{ e}$ ) to hydrophilicity ( $q \geq 0.4 \text{ e}$ ), the evaporation speed did not show a monotonic decrease from intuition, but increased first and then decreased after reaching a maximum value. Figure 16.1 depicts the MD simulation method: (a) side view of the initial system (the upward arrow denotes the accelerating region), (b) snapshot of water on the substrate with low wettability (the water molecules accumulate into the form of a sessile spherical cap with a contact angle  $\theta$  to the surface), (c) snapshot of water on the substrate with high wettability (the adhered water forms a flat single-layer molecule sheet), and (d) theoretical topology of water molecules with respect to substrate wettability used for MD simulations.

Considering MD simulation results from end to beginning, a fine analogy can be found between this type of water evaporation phenomena and a population-based



**Fig. 16.1** (a) Side view of the initial system. (b) snapshot of water on the substrate with low wettability ( $q = 0$  e). (c) snapshot of water on the substrate with high wettability ( $q = 0.7$  e). (d) theoretical topology of water molecules with respect to substrate wettability used for MD simulations

Water molecules	~	Algorithm individuals
Substrate with decreasing wettability	~	Search space with different objective function values
Water aggregation reforms from a monolayer to a sessile droplet with decreasing the surface wettability	~	Change mutual positions of individuals as the algorithm progresses
Decreasing $q$ from 0.7 e to 0.0	~	Decreasing the objective function value
$q = 0.4$ e	~	The algorithm reaches the middle of the optimization process
Evaporation in two phases (Monolayer and droplet) with different evaporation flux	~	Global and local search ability of the algorithm

**Fig. 16.2** Analogy between water evaporation from a solid surface and a population-based metaheuristic algorithm

metaheuristic algorithm. This analogy led us to developing the basic WEO algorithm [4, 5] depicted in Fig. 16.2.

Water molecules and substrate with decreasing wettability are considered as algorithm individuals and search space, respectively. Decreasing the surface wettability reforms the water aggregation from a monolayer to a sessile droplet. Similarly, mutual positions of individuals in the design space change as the search process progresses. Decreasing  $q$  from 0.7 e to 0.0 e can represent the reduction of the objective function for a minimization problem. Evaporation flux variation is considered as the most appropriate measure for updating the algorithm individuals which is in a good agreement with the local and global search ability.

Evaporation flux reaches its maximum around  $q=0.4$  e. This situation is considered in the basic WEO [4] until the algorithm reaches the middle of the optimization process. In other words, basic WEO updates the individuals in two independent sequential phases: monolayer and droplet evaporation phases. These two phases were explained in detail in [4]. In each phase the corresponding rules are in a good agreement with the local and global search ability of the algorithm. The steps entailed by the basic WEO are now outlined:

### Step 1: Initialization

Algorithm parameters are set in the first step. These parameters are the number of water molecules ( $nVM$ ), maximum number of algorithm iterations ( $t_{max}$ ), minimum ( $MEP_{min}$ ) and maximum ( $MEP_{max}$ ) values of monolayer evaporation probability, and minimum ( $DEP_{min}$ ) and maximum ( $DEP_{max}$ ) values of droplet evaporation probability. Evaporation probability parameters are determined efficiently for WEO based on the MD simulation results ( $MEP_{min}=0.03$  and  $MEP_{max}=0.6$ ;  $DEP_{min}=0.6$  and  $DEP_{max}=1$ ). The initial positions of all water molecules are generated randomly within the  $n$ -dimensional search space ( $WM^{(0)}$ ) as follows and are evaluated based on the objective function of the problem at hand:

$$WM_{i,j}^{(0)} = \text{Round}\left(x_{j,min} + rand_{i,j} \cdot (x_{j,max} - x_{j,min})\right) \quad (16.1)$$

where  $WM_{i,j}^{(0)}$  determines the initial value of the  $j$ th variable for the  $i$ th water molecule;  $x_{j,min}$  and  $x_{j,max}$  are the minimum and maximum allowable values for the  $j$ th variable; and  $rand_{i,j}$  is a random number in the interval [0, 1]. The rounding function is used for discrete problems and rounds the value of design variable to the nearest discrete available value. The best water molecule as the output of the algorithm will be returned in this step.

### Step 2: Generating Water Evaporation Matrix

Every water molecule follows the evaporation probability rules specified for each phase of the algorithm.

For  $t \leq t_{max}/2$  or in the monolayer evaporation phase, in each iteration the objective function of individuals  $Fit_i^t$  is scaled to the interval  $[-3.5, -0.5]$  and represents the corresponding  $E_{sub}(i)^t$  inserted to each individual (substrate energy vector) via the following scaling function:

$$E_{sub}(i)^t = \frac{(E_{max} - E_{min}) \times (Fit_i^t - \text{Min}(Fit))}{(\text{Max}(Fit) - \text{Min}(Fit))} + E_{min} \quad (16.2)$$

$\text{Min}$  and  $\text{Max}$  are the minimum and maximum functions, respectively. After generating the corresponding substrate energy vector, water molecules are globally evaporated based on the monolayer evaporation probability ( $MEP$ ) rule:

$$MEP_{ij}^t = \begin{cases} 1 & \text{if } rand_{i,j} < \exp(E_{sub}(i)^t) \\ 0 & \text{if } rand_{i,j} \geq \exp(E_{sub}(i)^t) \end{cases} \quad (16.3)$$

where  $MEP_{ij}^t$  is the updating probability for the  $j$ th variable of the  $i$ th individual or water molecule in the  $t$ th optimization iteration and  $rand_{i,j}$  is a random number generated by the uniform distribution in the interval  $[0, 1]$ . In this way, in the monolayer evaporation phase, the best and worst candidate solutions will be updated by the probability equal to  $\exp(-3.5) = 0.03$  and  $\exp(-0.5) = 0.6$ , respectively. These values can be considered as minimum ( $MEP_{min}$ ) and maximum ( $MEP_{max}$ ) values of monolayer evaporation probability. Algorithm performance evaluations show that considering  $MEP_{min} = 0.03$  and  $MEP_{max} = 0.6$  based on the MD simulation results is logical.

For  $t > t_{max}/2$  or in the droplet evaporation phase, the objective function of individuals  $Fit_i^t$  is scaled to the interval  $[-50^\circ, -20^\circ]$  via the following scaling function which represents the corresponding contact angle  $\theta(i)^t$  (contact angle vector):

$$\theta(i)^t = \frac{(\theta_{max} - \theta_{min}) \times (Fit_i^t - \text{Min}(Fit))}{(\text{Max}(Fit) - \text{Min}(Fit))} + \theta_{min} \quad (16.4)$$

After generating contact angle vector, evaporation occurs based on the droplet evaporation probability ( $DEP$ ) matrix:

$$DEP_{ij}^t = \begin{cases} 1 & \text{if } rand_{i,j} < J\left(\theta_i^{(t)}\right) \\ 0 & \text{if } rand_{i,j} \geq J\left(\theta_i^{(t)}\right) \end{cases} \quad (16.5)$$

$$J(\theta) = J_0 P_0 \left( \frac{2}{3} + \frac{\cos^3 \theta}{3} - \cos \theta \right)^{-2/3} (1 - \cos \theta), \quad J_0 P_0 = \frac{1}{24}$$

where  $DEP_{ij}^t$  is the updating probability for the  $j$ th variable of the  $i$ th water molecule in the  $t$ th optimization iteration. In this way, minimum ( $DEP_{min}$ ) and maximum ( $DEP_{max}$ ) values of droplet evaporation probability are obtained equal to 0.6 and 1, respectively. Algorithm performance evaluation results show that these values are suitable.

### Step 3: Generating Random Permutation Based Step Size Matrix

A random permutation based step size matrix is generated according to:

$$S = rand \cdot \left( WM^{(t)}[\text{permute1}(i)(j)] - WM^{(t)}[\text{permute2}(i)(j)] \right) \quad (16.6)$$

where  $rand$  is a random number in the  $[0, 1]$  range,  $\text{permute1}$  and  $\text{permute2}$  are different row permutation functions,  $i$  is the number of water molecule, and  $j$  is the number of design variables of the problem at hand. It should be noted that random permutation based step size is used for two purposes. In the first phase, water

molecules are more far from each other than in the second. In this way, the generated permutation based step size will guarantee global and local search capability in each phase. The random part will guarantee the algorithm to be sufficiently dynamic. These two aspects are emphasized by considering two specific evaporation probability mechanisms for each phase.

#### **Step 4: Generating Evaporated Water Molecules and Updating the Matrix of Water Molecules**

The evaporated set of water molecules  $WM^{(t+1)}$  is generated by adding the product of step size matrix and evaporation probability matrix to the current set of molecules  $WM^{(t)}$  according to:

$$WM^{(t+1)} = \text{Round} \left( WM^{(t)} + S \times \begin{cases} MEP^{(t)} & t \leq t_{max}/2 \\ DEP^{(t)} & t > t_{max}/2 \end{cases} \right) \quad (16.7)$$

The rounding function is used for discrete problems and rounds the value of design variables to the nearest discrete available value. These water molecules are evaluated based on the objective function. For the molecule  $i$  ( $i = 1, 2, \dots, nWM$ ), if the newly generated molecule  $i$  ( $i = 1, 2, \dots, nWM$ ) is better than the old one, it will replace it. The best water molecule (best-WM) is returned.

#### **Step 5: Terminating Condition**

If the number of iterations of the algorithm ( $t$ ) becomes larger than the maximum number of iterations ( $t_{max}$ ), the algorithm terminates. Otherwise go to Step 2.

### **16.3 Water Evaporation Optimization with Mixed Phases**

The basic WEO algorithm updates candidate designs in two independent sequential phases: monolayer and droplet evaporation phases. Design updating strategies selected in each phase are consistent with the local and global search ability of the algorithm. The accelerated version of WEO involving the simultaneous use of both phases is now described.

Two scenarios are developed for considering simultaneous evaporation phases based on (1) water molecules' distances and (2) objective function values of molecules. First scenario is physically based and interpreted from water evaporation phenomena. The second one is considered in a biased way without any physics-based interpretation to investigate the possible improvements of the algorithm. It should be noted that the accelerated WEO is presented here for minimization problems. If the first scenario is adopted, after each individual has been evaluated, water molecules are sorted in ascending order based on their distance ( $dist_i$ ) from the worst current water molecule (*worst-WM*) by the following equation:

$$dist_i = |worstWM - WM_i|, \quad i = 1, 2, \dots, nWM \quad (16.8)$$

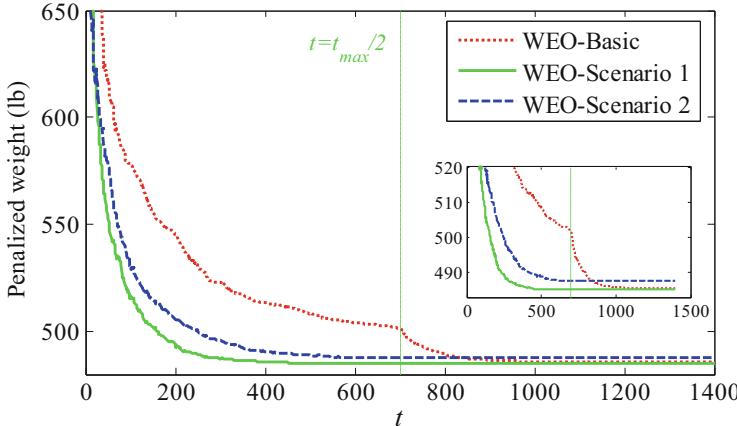
If the second scenario is adopted, water molecules are sorted in ascending order based on the objective function value. For both scenarios, water molecules are divided in two groups with the same number of molecules ( $nWM/2$ ) based on the distance from the worst current molecule and objective function value, respectively. In the case of the first scenario, the first half of water molecules in each optimization iteration will be updated by the droplet evaporation probability, and the second half are evaporated based on the monolayer evaporation probability. The second scenario would be the contrary of first one: the groups of molecules will be updated by monolayer and droplet evaporation probability, respectively.

Our simulation results show that the first scenario is effective and successful in enhancing the convergence rate, considering three variants (basic WEO, WEO with mixed phases using the first scenario, and WEO with mixed phases using the second scenario). Table 16.1 presents the statistical results obtained for 20 independent optimization runs carried out from different initial populations randomly generated by the three variants of WEO for the first test problem (a spatial 25-bar truss) solved in this study. Average penalized weight optimized histories obtained by these three variants for 20 independent optimization runs starting from a different population randomly generated are depicted in Fig. 16.3 for this test case. It appears that the accelerated WEO using the first scenario is efficient in enriching the WEO to work with less number of structural analyses.

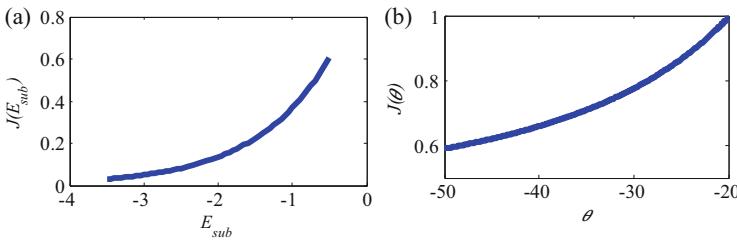
According to the previous paragraph and because of the physical interpretation of the first scenario, this strategy is developed here. Figures 16.4(a) and (b) show monolayer evaporation flux variation with different substrate energy and droplet evaporation probability variation with different contact angles, respectively. The rationale behind the first scenario is that the worst water molecule and molecules nearby are updated globally ( $0.6 \leq J(\theta) \leq 1.0$ ) by the droplet evaporation probability (*DEP*) and the molecules far from the worst molecule are updated locally ( $0.03 \leq exp(E_{sub}) \leq 0.6$ ) with the monolayer evaporation flux (*MEP*). The second scenario also is inspired by this rationale but greedily divides the molecules for local and global search using the criteria of objective function value in a way that better molecules are updated using *MEP*, and others are updated via *DEP*. Molecules get closer together with the progress of algorithm. Consequently, random

**Table 16.1** Comparison of relative efficiency of three WEO variants for the 25-bar problem

Algorithm	Weight (lb)			Difference best–average solution (%)	Difference best–worst solution (%)	SD
	Best	Average	Worst			
WEO-basic	484.854	485.598	489.158	0.15	0.73	1.149
WEO-scenario 1	484.854	485.252	487.989	0.08	0.56	0.715
WEO-scenario 2	484.854	487.584	509.351	0.56	4.27	6.119



**Fig. 16.3** Convergence curves recorded for the spatial 25-bar truss using three variants of WEO



**Fig. 16.4** (a) Monolayer evaporation flux with different substrate energy and (b) droplet evaporation flux with different contact angles [3, 4]

permutation based step size (Eq. (16.6)) used for generating evaporated molecules, along with different evaporation mechanisms, will enhance global and local search capability. It should be noted that in the basic WEO, utilizing the random permutation based step size (Eq. (16.6)), all molecules are updated globally using MEP and locally using DEP in the first and second half of the optimization process, respectively.

The steps involved in the first scenario of the accelerated WEO are as follows:

### Step 1: Initialization

This step is the same as the one described in the basic WEO. Additionally the worst water molecule (*worst-WM*) will be monitored in this step.

### Step 2: Generating Water Evaporation Matrix

Calculate the distance vector between all water molecules and the worst current one using Eq. (16.8). Sort molecules based on their distance values in an ascending order. Generate the *DEP* matrix for updating the first half of the molecules (*droplet-WM*) using the droplet evaporation rule (Eq. (16.5)). Generate the *MEP* matrix for

---

```

for i=1:nWM
    Dist(i)=norm(WM(i,:)-worst-WM);
end
[a,b]=sort(Dist);

for i=1:nWM/2
    droplet-WM(i,:)=WM(b(i),:);
end
Generate the corresponding  $\theta$  vector and  $DEP$  matrix using Eqs. (16.4) and (16.5), respectively.

for i=1: nWM/2
    monolayer-WM(i,:)=WM(b(size(nWM/2+i),:));
end
Generate the corresponding  $E_{sub}$  vector and  $MEP$  matrix using Eqs. (16.2) and (16.3), respectively.

for i=1:size(WM,1)
    if i<= nWM/2
        MDEP(b(i),:)=DEP(i,:);
    else
        MDEP (b(i),:)=MEP(i-size(WM,1)/2,:);
    end
end

```

---

**Fig. 16.5** Pseudo code for generating the mixed evaporation matrix ( $MDEP$ )

updating the next half of the molecules (*monolayer-WM*) based on the monolayer evaporation probability (Eq. (16.3)). In order to generate droplet and evaporation probability matrices, it is also necessary to define the corresponding contact angle vector (Eq. (16.4)) and substrate energy vector (Eq. (16.2)), respectively. It should be noted that in the accelerated version of WEO droplet and evaporation probability matrices and corresponding substrate energy and contact angle vectors include  $nWM/2$  rows. The mixed evaporation matrix (*MDEP*) including evaporation vectors of each molecule is assembled from *MEP* and *DEP* matrices. The pseudo code shown in Fig. 16.5 can be used for generating the mixed evaporation matrix (*MDEP*).

### Step 3: Generating Random Permutation Based Step Size Matrix

A random permutation based step size matrix is generated using Eq. (16.6).

### Step 4: Generating Evaporated Water Molecules and Updating the Matrix of Water Molecules

The evaporated set of water molecules  $WM^{(t+1)}$  is generated by adding the product of step size matrix and mixed evaporation probability matrix (*MDEP*) to the current set of molecules  $WM^{(t)}$  according to:

$$WM^{(t+1)} = \text{Round} \left( WM^{(t)} + S \times MDEP^{(t)} \right) \quad (16.9)$$

The rounding function is used for discrete problems and rounds the value of design variables to the nearest discrete available value. These water molecules are evaluated based on the objective function. For the molecule  $i$  ( $i = 1, 2, \dots, nWM$ ), if the newly generated molecule is better than the current one, it should be replaced. The best water molecule is returned in this step.

### Step 5: Terminating Condition

If the number of iterations of the algorithm ( $t$ ) becomes larger than the maximum number of iterations ( $t_{max}$ ), the algorithm terminates. Otherwise go to Step 2.

## 16.4 Test Problems and Optimization Results

Finding optimum design of the skeletal structures is an effective way to test new optimization algorithms as these problems may include many design variables and constraints as well as large search spaces. The developed accelerated version of WEO algorithm is tested on four well-known discrete weight minimization problems: 25-bar spatial truss, 72-bar spatial truss, 3-bay 15-story frame, and 3-bay 24-story frame.

These test cases included 8, 16, 11, and 20 sizing variables, respectively. In truss problems with discrete variables, search space is defined using a series of discrete values between two lower and upper bounds. The steel members used for the design of steel frames consist of 267 W-shaped sections from the LRFD–AISC database, from  $W44 \times 335$  to  $W4 \times 13$ . These sections with their properties are used to prepare a design pool. The sequence numbers assigned to this pool that are sorted with respect to the 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. Truss problems are designed under stress and nodal displacement constraints. Statement of the truss weight minimization problem can be found in [8]. Strength constraints of AISC load and resistance factor design (LRFD) specification [9] and displacement constraints are imposed on frames. Statement of the planar frame weight minimization problem can be found in [10]. In order to handle optimization constraints, a penalty approach was utilized in this study by introducing the following pseudo cost function:

$$f_{cost}(\{X\}) = (1 + \varepsilon_1 \cdot \nu)^{\varepsilon_2} \times W(\{X\}) \quad (16.10)$$

where  $\{X\}$  is the set of design variables;  $W(\{X\})$  is the weight of the structure; and  $\nu$  is the total constraint violation. Constants  $\varepsilon_1$  and  $\varepsilon_2$  must be selected considering the exploration and the exploitation rate of the search space. In this study,  $\varepsilon_1$  was set equal to one while  $\varepsilon_2$  was selected so as to decrease the total penalty yet reducing

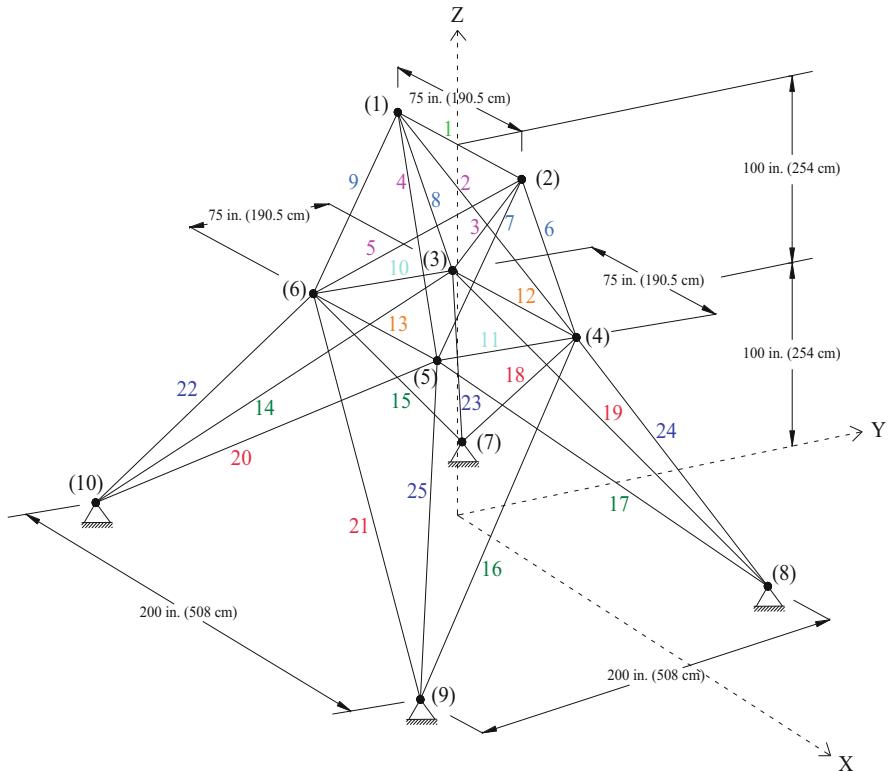
cross-sectional areas. Thus,  $\varepsilon_2$  increased from the value of 1.5 set in the first steps of the search process to the value of 3 set toward the end of the optimization process. It should be noted that in the discrete structural optimization problems the method selected for constraints handling affects the algorithms in the aspect of accuracy. Here we used the penalty approach as one of the simplest methods and with fixed constants for all test problems according to previous studies.

WEO algorithm is compared in two variants: basic WEO and WEO with mixed phases. In order to better evaluate performance of the developed accelerated version, the most effective available state-of-the-art metaheuristic optimization methods based on the authors knowledge are used here as basis of comparison. Since the search process is governed by random rules, each test problem was solved by carrying out 20 independent optimization runs considering different initial populations to obtain statistically significant results. During each run, the maximum number of structural analyses ( $NSA$ ) of  $14 \times 10^3$  is used. Other parameters are as follows [4, 5]: the number of water molecules,  $nWM = 10$ ; minimum and maximum value of monolayer evaporation probabilities,  $MEP_{min} = 0.03$  and  $MEP_{max} = 0.6$ ; and minimum and maximum value of droplet evaporation probabilities,  $DEP_{min} = 0.6$  and  $DEP_{max} = 1$ . The WEO is coded in the MATLAB software environment. Structural analyses entailed by the optimization process were performed by means of the direct stiffness method.

### 16.4.1 A Spatial 25-Bar Tower Truss

The 25-bar transmission tower is used widely in structural optimization to verify various metaheuristic algorithms. Truss geometry including node and element numbering is shown in Fig. 16.6. The material density is 0.1 lb/in<sup>3</sup> and the modulus of elasticity is 10<sup>7</sup> psi. Twenty-five members are categorized into eight groups as follows: (1) A<sub>1</sub>, (2) A<sub>2</sub>–A<sub>5</sub>, (3) A<sub>6</sub>–A<sub>9</sub>, (4) A<sub>10</sub>–A<sub>11</sub>, (5) A<sub>12</sub>–A<sub>13</sub>, (6) A<sub>14</sub>–A<sub>17</sub>, (7) A<sub>18</sub>–A<sub>21</sub>, and (8) A<sub>22</sub>–A<sub>25</sub>. The problem is described in detail in [8].

Table 16.2 tabulates the best optimized designs found by basic WEO and accelerated WEO with mixed phases based on the first scenario. All optimized designs are fully feasible. Selected algorithms to evaluate the accelerated WEO are artificial bee colony (ABC) [11], mine blast algorithm (MBA) [12], cuckoo search (CS) [8], and colliding bodies optimization (CBO) [13]. As it is clear, the accelerated WEO requires only half of the NSA of the basic WEO. Accelerated WEO needs twice the NSA needed for MBA and CS and needs less NSA compared to the CBO. The most interesting point is that the accelerated WEO not only is modified significantly in terms of accuracy but also shows better performance in terms of the computational time. The mean optimized weight found by the accelerated WEO is lower than those found by the basic WEO. However, the basic WEO can reach the same average results by considering higher NSA ( $20 \times 10^3$ ).



**Fig. 16.6** Schematic of the spatial 25-bar truss

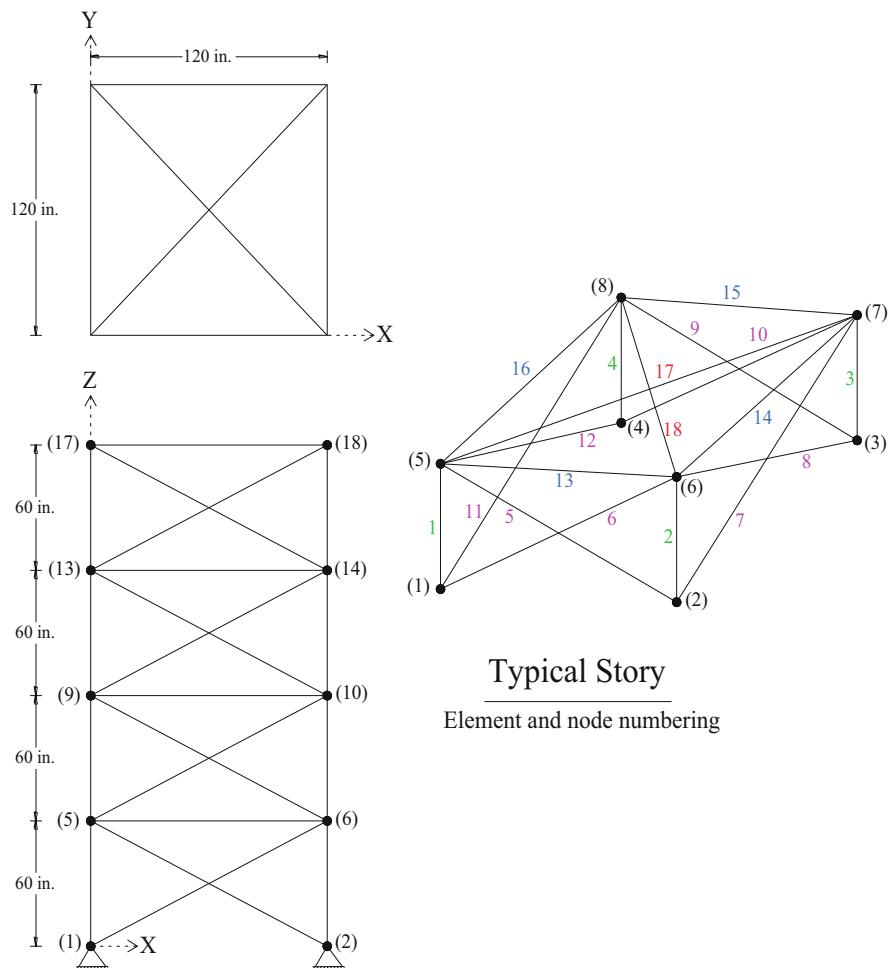
**Table 16.2** Comparison of optimization results obtained by WEO and some other metaheuristic algorithms for the 25-bar tower problem

Element group	Optimal cross-sectional areas ( $\text{in}^2$ )					
	ABC [11]	MBA [12]	CS [8]	CBO [13]	Basic WEO	WEO-scenario 1
1	0.1	0.1	0.1	0.1	0.1	0.1
2	0.3	0.3	0.30	0.3	0.3	0.3
3	3.4	3.4	3.4	3.4	3.4	3.4
4	0.1	0.1	0.1	0.1	0.1	0.1
5	2.1	2.1	2.1	2.1	2.1	2.1
6	1.0	1.0	1.0	1.0	1.0	1.0
7	0.5	0.5	0.5	0.5	0.5	0.5
8	3.4	3.4	3.4	3.4	3.4	3.4
Best weight (lb)	484.85	484.85	484.85	484.85	484.85	484.85
Mean weight (lb)	485.05	484.89	485.01	486.87	485.598	485.252
NSA	NA	2150	2000	7050	9610	5060

### 16.4.2 A Spatial 72-Bar Truss

For the spatial 72-bar truss structure shown in Fig. 16.7, the material density is 0.1 lb/in.<sup>3</sup> and the modulus of elasticity is 10<sup>7</sup> psi. The 72 bars are categorized into 16 groups using symmetry: (1) A<sub>1</sub>–A<sub>4</sub>, (2) A<sub>5</sub>–A<sub>12</sub>, (3) A<sub>13</sub>–A<sub>16</sub>, (4) A<sub>17</sub>–A<sub>18</sub>, (5) A<sub>19</sub>–A<sub>22</sub>, (6) A<sub>23</sub>–A<sub>30</sub>, (7) A<sub>31</sub>–A<sub>34</sub>, (8) A<sub>35</sub>–A<sub>36</sub>, (9) A<sub>37</sub>–A<sub>40</sub>, (10) A<sub>41</sub>–A<sub>48</sub>, (11) A<sub>49</sub>–A<sub>52</sub>, (12) A<sub>53</sub>–A<sub>54</sub>, (13) A<sub>55</sub>–A<sub>58</sub>, (14) A<sub>59</sub>–A<sub>66</sub>, (15) A<sub>67</sub>–A<sub>70</sub>, and (16) A<sub>71</sub>–A<sub>72</sub>. The structure must be designed for multiple loading conditions. The problem is described in detail in [8].

Table 16.3 presents the statistical results obtained for 20 independent runs carried out from different initial populations randomly generated by the three



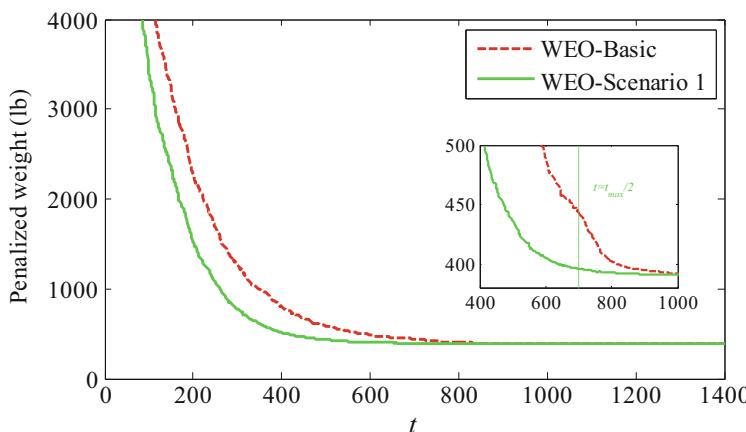
**Fig. 16.7** Schematic of the spatial 72-bar truss

**Table 16.3** Robustness and reliability analyses of WEO in 72-bar truss problem

Algorithm	Weight (lb)			Difference best–average solution (%)	Difference best–worst solution (%)	SD
	Best	Average	Worst			
WEO-basic	389.33	391.20	395.84	0.48	1.17	1.81
WEO-scenario 1	389.33	390.94	397.74	0.41	1.71	1.93
WEO-scenario 2	389.46	391.75	399.34	0.58	1.90	2.18

variants of WEO. Average penalized weight histories obtained by the basic and accelerated WEO for the 20 independent optimization runs are shown in Fig. 16.8. It appears that accelerated WEO with the first scenario allows to reduce the computational cost of optimization search compared to the basic WEO.

Table 16.4 lists the best optimized designs found by basic WEO and accelerated WEO with mixed phases based on the first scenario. All optimized designs are fully feasible. Both variants of WEO are compared with imperialist competitive algorithm (ICA) [14], CS [8], and CBO [15]. Accelerated WEO found the lightest design overall but needs 7500 NSA, nearly twice the number of analyses needed by other algorithms. It should be noted that the NSA required by accelerated WEO to find intermediate designs better than the optimized designs found by ICA, CS, and CBO (391.26 lb) is equal to 6200. Basic WEO also can find the lightest design, but it is clear that considering mixing phases formulation enables WEO to work with less computational cost (30 %). The most interesting point is that WEO with mixed phases results in averagely lightest optimum design than the basic WEO. It should be noted that the basic WEO can yield the same average results considering more NSA ( $20 \times 10^3$ ) as the stopping criteria.

**Fig. 16.8** Convergence curves recorded for the 72-bar truss problem

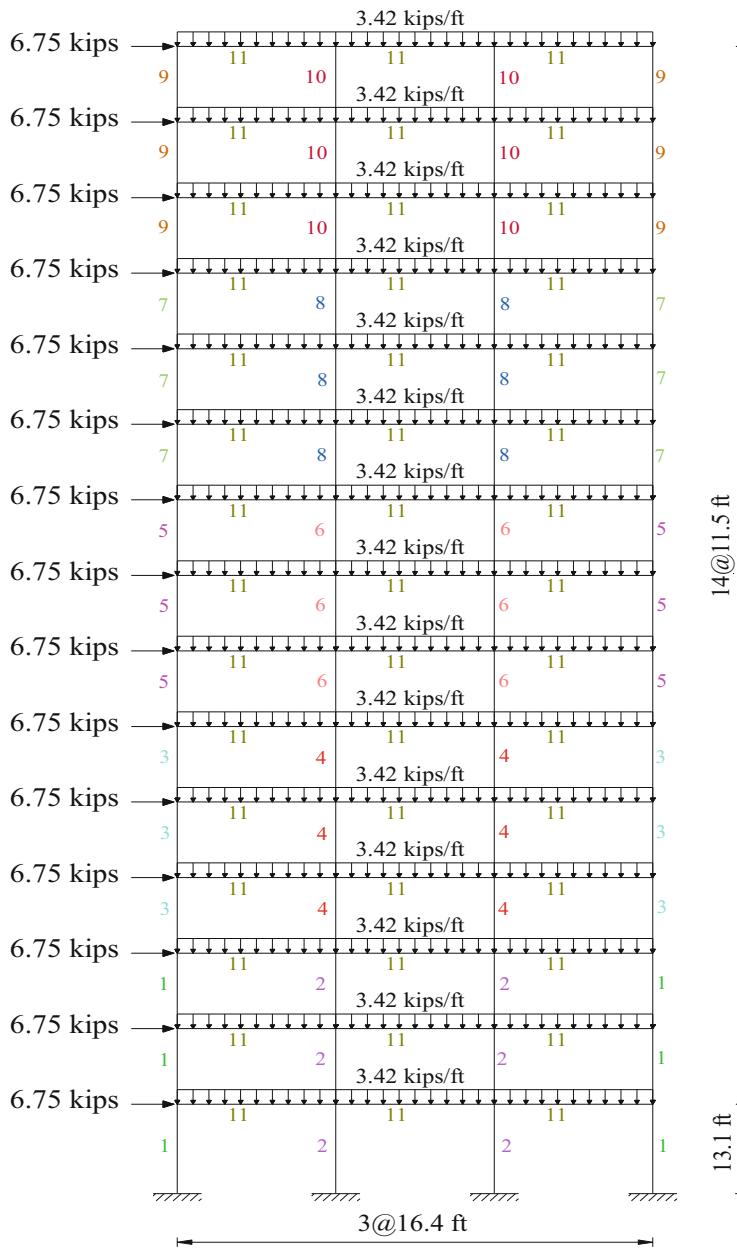
**Table 16.4** Comparison of optimization results obtained by WEO and some other metaheuristic algorithms for the 72-bar truss problem

Element group	Optimal cross-sectional areas (in <sup>2</sup> )				
	ICA [14]	CS [8]	CBO [15]	Basic WEO	Accelerated WEO
	Present work [5]				
1	A <sub>1</sub> ~ A <sub>4</sub>	1.99	1.800	1.62	1.99
2	A <sub>5</sub> ~ A <sub>12</sub>	0.442	0.563	0.563	0.563
3	A <sub>13</sub> ~ A <sub>16</sub>	0.111	0.111	0.111	0.111
4	A <sub>17</sub> ~ A <sub>18</sub>	0.141	0.111	0.111	0.111
5	A <sub>19</sub> ~ A <sub>22</sub>	1.228	1.266	1.457	1.228
6	A <sub>23</sub> ~ A <sub>30</sub>	0.602	0.563	0.442	0.442
7	A <sub>31</sub> ~ A <sub>34</sub>	0.111	0.111	0.111	0.111
8	A <sub>35</sub> ~ A <sub>36</sub>	0.141	0.111	0.111	0.111
9	A <sub>37</sub> ~ A <sub>40</sub>	0.563	0.563	0.602	0.563
10	A <sub>41</sub> ~ A <sub>48</sub>	0.563	0.442	0.563	0.563
11	A <sub>49</sub> ~ A <sub>52</sub>	0.111	0.111	0.111	0.111
12	A <sub>53</sub> ~ A <sub>54</sub>	0.111	0.111	0.111	0.111
13	A <sub>55</sub> ~ A <sub>58</sub>	0.196	0.196	0.196	0.196
14	A <sub>59</sub> ~ A <sub>66</sub>	0.563	0.602	0.602	0.563
15	A <sub>67</sub> ~ A <sub>70</sub>	0.307	0.391	0.391	0.391
16	A <sub>71</sub> ~ A <sub>72</sub>	0.602	0.563	0.563	0.563
Best weight (lb)		392.84	389.87	391.07	389.33
Mean weight (lb)		N/A	N/A	403.71	391.20
NSA		4500	5450	4500	10,510
					7490

### 16.4.3 A 3-Bay 15-Story Frame

The configuration, service loading conditions, and numbering of member groups for the 3-bay 15-story frame are shown in Fig. 16.9. All 64 columns are grouped into nine groups while 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 available in [10].

Table 16.5 compares the accelerated WEO with basic WEO and other state-of-the-art metaheuristic algorithms which efficiently solved this test problem: ICA [14], charged system search (CSS) [16], and enhanced version of CBO (ECBO) [13]. All optimized designs are fully feasible. The lightest design is obtained by ECBO algorithm which needs nearly the same NSA as accelerated WEO. Accelerated WEO is better than basic WEO considering the best run and average of 20 independent runs starting from different populations randomly



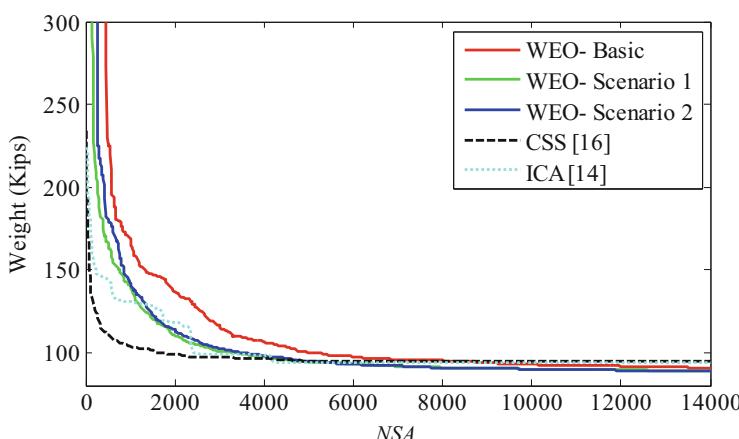
**Fig. 16.9** Schematic of the 3-bay 15-story frame

generated, and computational cost. Accelerated WEO leads to lighter design than ICA and CSS but needs nearly twice NSA.

**Table 16.5** Comparison of optimization results obtained by WEO and some other metaheuristic algorithms for the 3-bay 15-story frame problem

Element group	Optimal cross-sectional areas (W types)				
	ICA [14]	CSS [16]	ECBO [13]	Basic WEO	Accelerated WEO
	Present study [5]				
1	W24 × 117	W21 × 147	W14 × 99	W14 × 90	W14 × 99
2	W21 × 147	W18 × 143	W27 × 161	W36 × 170	W27 × 161
3	W27 × 84	W12 × 87	W27 × 84	W30 × 90	W27 × 84
4	W27 × 114	W30 × 108	W24 × 104	W24 × 104	W24 × 104
5	W14 × 74	W18 × 76	W14 × 61	W24 × 68	W14 × 61
6	W18 × 86	W24 × 103	W30 × 90	W12 × 87	W30 × 90
7	W12 × 96	W21 × 68	W14 × 48	W8 × 48	W16 × 50
8	W24 × 68	W14 × 61	W14 × 61	W14 × 68	W21 × 68
9	W10 × 39	W18 × 35	W14 × 30	W10 × 33	W14 × 34
10	W12 × 40	W10 × 33	W12 × 40	W16 × 45	W8 × 35
11	W21 × 44	W21 × 44	W21 × 44	W21 × 44	W21 × 44
Best weight (lb)	93,846	92,723	86,986	88,710.97	87,537.96
Mean weight (lb)	N/A	N/A	88,410	90,649.49	88,893.09
NSA	6000	5000	9000	13,580	10,670

Figure 16.10 compares average convergence histories for different formulations of the WEO, CSS [16], and ICA [14] algorithms. It should be noted that the weight optimization convergence history is monitored because of the available results in the literature. For more clarity, the y-axis is in units of kips and its upper bound is limited to 300. Best convergence rate is obtained by CSS algorithm. As it is clear, the new formulation based on the scenario 1 makes the WEO competitive with other algorithms also in terms of convergence rate.

**Fig. 16.10** Average convergence histories of algorithms for the 3-bay 15-story frame problem

### 16.4.4 A 3-Bay 24-Story Frame

The 3-bay 24-story frame optimized in the last test problem is shown in Fig. 16.11. The structure is comprised of 168 members that are collected in 20 groups (16 column groups and four beam groups). 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). Additional information about the problem can be found in [13].

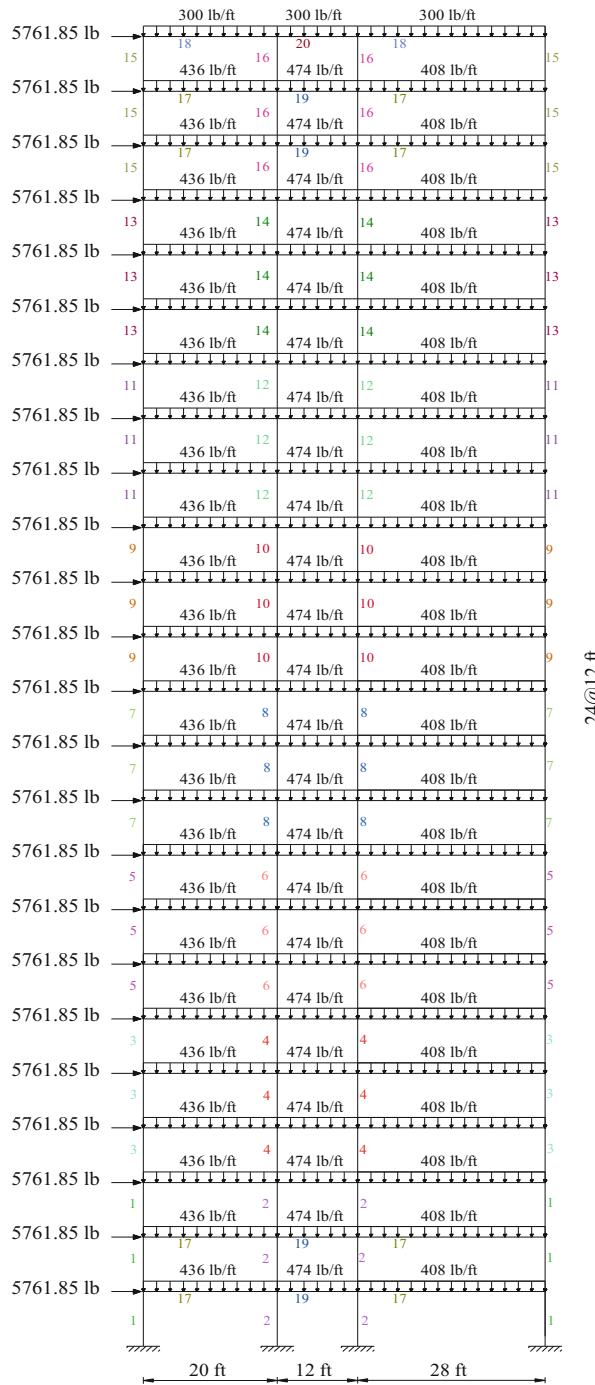
Table 16.6 compares the accelerated WEO with basic WEO and other state-of-the art metaheuristic algorithms which efficiently solved this test problem: harmony search (HS) [17], CSS [16], and ECBO [13]. All optimized designs are fully feasible. ECBO shows the best performance. Accelerated WEO performs better than the basic WEO and HS, in relation to both accuracy and computational cost, and leads to a lighter design than that of the CSS. However, the present algorithm requires more structural analyses to complete the optimization process.

Figure 16.12 compares average convergence histories for different formulations of the WEO, CSS [16], and HS [17] algorithms. For more clarity, the y-axis is in the units of kips and its upper bound is limited to 700. Best convergence rate is again obtained by CSS algorithm. The new formulation based on the scenario 1 makes WEO enough competitive in terms of convergence rate.

## 16.5 Concluding Remarks

This chapter presented an accelerated version of water evaporation optimization algorithm (WEO). WEO is a recently developed population physics-based metaheuristic algorithm which follows the well-known rules governing the evaporation process of water molecules from a solid surface with different wettability. In the basic WEO, molecules are updated globally and locally, respectively, in two independent sequential phases: monolayer and droplet evaporation phases. It was shown that computational cost of the WEO optimizations can be reduced by simultaneously using the two phases. For that purpose, we developed two scenarios based on the (1) water molecules' distance and (2) objective function value of molecules.

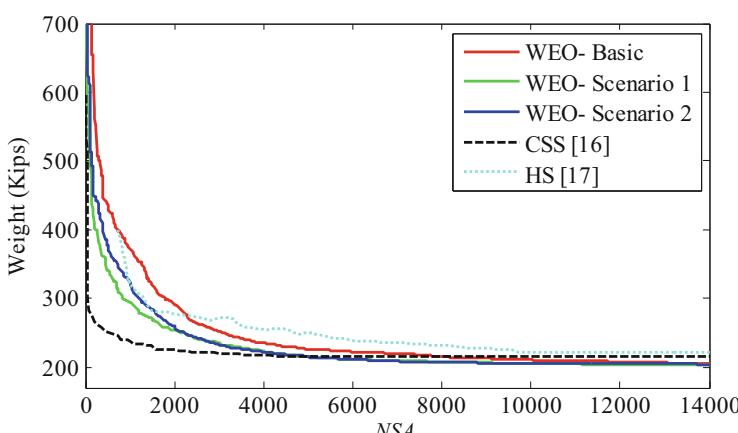
The new algorithm was successfully tested in four well-known discrete structural optimization problems. The first scenario which is in a good agreement with physical aspects of the water evaporation is more efficient than the second one and hence developed in this study. It turns out that the developed framework provides a significant enhancement in convergence rate compared to the basic WEO. However, further investigations should be carried out in order to reduce the computational cost of the accelerated WEO in frame problems with respect to other metaheuristic algorithms such as ECBO.



**Fig. 16.11** Schematic of the 3-bay 24-story frame

**Table 16.6** Comparison of optimization results obtained by WEO and some other metaheuristic algorithms for the 3-bay 24-story frame problem

Element group	Optimal cross-sectional areas (W types)				
	HS [17]	CSS [16]	ECBO [13]	Basic WEO	Accelerated WEO
	Present study [5]				
1	W14 × 176	W14 × 176	W14 × 145	W14 × 132	W14 × 159
2	W14 × 145	W14 × 145	W14 × 99	W14 × 109	W14 × 132
3	W14 × 176	W14 × 145	W14 × 132	W14 × 109	W14 × 99
4	W14 × 132	W14 × 132	W14 × 99	W14 × 90	W14 × 109
5	W14 × 132	W14 × 109	W14 × 99	W14 × 61	W14 × 68
6	W14 × 109	W14 × 109	W14 × 99	W14 × 38	W14 × 38
7	W14 × 109	W14 × 90	W14 × 90	W14 × 38	W14 × 30
8	W14 × 82	W14 × 82	W14 × 82	W14 × 22	W14 × 22
9	W14 × 82	W14 × 74	W14 × 74	W14 × 109	W14 × 90
10	W14 × 61	W14 × 68	W14 × 68	W14 × 109	W14 × 99
11	W14 × 74	W14 × 61	W14 × 38	W14 × 99	W14 × 99
12	W14 × 48	W14 × 43	W14 × 61	W14 × 90	W14 × 74
13	W14 × 34	W14 × 34	W14 × 38	W14 × 82	W14 × 68
14	W14 × 30	W14 × 34	W14 × 30	W14 × 68	W14 × 61
15	W14 × 22	W14 × 34	W14 × 22	W14 × 34	W14 × 34
16	W14 × 22	W14 × 22	W14 × 22	W14 × 22	W14 × 22
17	W30 × 90	W30 × 90	W30 × 90	W30 × 90	W30 × 90
18	W10 × 22	W21 × 50	W6 × 15	W6 × 15	W8 × 18
19	W18 × 40	W21 × 48	W24 × 55	W24 × 55	W24 × 55
20	W12 × 16	W12 × 19	W6 × 8.5	W6 × 8.5	W6 × 8.5
Best weight (lb)	214,860	212,364	201,618	202,626.00	202,194.02
Mean weight (lb)	222,620	215,226	209,644	204,954.03	203,412.88
NSA	13,924	5500	2800	13,510	11,300



**Fig. 16.12** Average convergence histories of algorithms for the 3-bay 24-story frame problem

## References

1. Xiong N, Molina D, Ortiz ML, Herrera F (2015) A walk into metaheuristics for engineering optimization: principles, methods and recent trends. *Int J Comput Intell Syst* 8:606–636
2. Talbi E-G (2009) Metaheuristics: from design to implementation. Wiley, Hoboken, NJ
3. Kaveh A, Bakhshpoori T (2016) Water Evaporation Optimization: a novel physically inspired optimization algorithm. *Comput Struct* 167:69–85
4. Kaveh A, Bakhshpoori T (2016) A novel metaheuristic for continuous structural optimization: Water Evaporation Optimization. *Struct Multidisip Optim* 54(1):23–43
5. Kaveh A, Bakhshpoori T (2016) An accelerated water evaporation optimization formulation for discrete optimization of skeletal structures. *Comput Struct* 177:218–228
6. Zarei G, Homaei M, Liaghat AM, Hoorfar AH (2010) A model for soil surface evaporation based on Campbell's retention curve. *J Hydro* 380:356–361
7. Wang S, Tu Y, Wan R, Fang H (2012) Evaporation of tiny water aggregation on solid surfaces with different wetting properties. *J Phys Chem B* 116:13863–13867
8. Kaveh A, Bakhshpoori T (2013) Optimum design of space trusses using cuckoo search algorithm with Lévy flights. *Iranian J Sci Technol Trans Civil Eng* 37:1–15
9. AISC (2001) Manual of steel construction: load and resistance factor design. AISC, Chicago, IL
10. Kaveh A, Bakhshpoori T (2013) Optimum design of steel frames using Cuckoo Search algorithm with Lévy flights. *Struct Des Tall Spec Build* 22:1023–1036
11. Sonmez M (2011) Discrete optimum design of truss structures using artificial bee colony algorithm. *Struct Multidiscip Optim* 43:85–97
12. Sadollah A, Bahreininejad A, Eskandar H, Hamdi M (2012) Mine blast algorithm for optimization of truss structures with discrete variables. *Comput Struct* 102–103:49–63
13. Kaveh A, Ilchi Ghazaan M (2015) A comparative study of CBO and ECBO for optimal design of skeletal structures. *Comput Struct* 153:137–147
14. Kaveh A, Talatahari S (2010) Optimum design of skeletal structures using imperialist competitive algorithm. *Comput Struct* 88:1220–1229
15. Kaveh A, Mahdavi VR (2014) Colliding Bodies Optimization method for optimum discrete design of truss structures. *Comput Struct* 139:43–53
16. Kaveh A, Talatahari S (2012) Charged system search for optimal design of frame structures. *Appl Soft Comput* 12:382–393
17. Degertekin SO (2008) Optimum design of steel frames using harmony search algorithm. *Struct Multidiscip Optim* 36:393–401

# Chapter 17

## Vibrating Particles System Algorithm

### 17.1 Introduction

In the recent years, many metaheuristics with different philosophies and characteristics are introduced and applied to a wide range of fields. The aim of these optimization methods is to efficiently explore the search space in order to find global or near-global solutions. Since they are not problem specific and do not require the derivatives of the objective function, they have received increasing attention from both academia and industry.

A novel population-based metaheuristic algorithm based on the damped free vibration of single degree of freedom system is introduced in Kaveh and Ilchi Ghazaan [1]. This algorithm is called a vibrating particles system (VPS) algorithm and considers each candidate solution as a particle that approaches its equilibrium position. By utilizing a combination of randomness and exploitation of obtained results, the quality of the particles improves iteratively as the optimization process proceeds. Here, viability of the proposed method is examined using the optimal design of two truss and two frame structures. The selected examples are among the most popular benchmarks previously studied by the researchers (Saka [2], Erbatur et al. [3], Lee and Geem [4], Hasançebi et al. [5], and Kazemzadeh Azad and Hasançebi [6]). The numerical results indicate the efficiency of the proposed algorithm compared to some other methods available in the literature.

The remaining sections of this chapter are organized as follows: The mathematical formulations of the structural optimization are presented in Sect. 17.2. The physical background of the VPS algorithm is presented in Sect. 17.3, and Sect. 17.4 introduces this new optimization method in detail. Section 17.5 investigates the parameter settings and the search behavior of the proposed method, and four structural design examples are studied in Sect. 17.6. Finally, concluding remarks are provided in Sect. 17.7.

## 17.2 Formulation of the Structural Optimization Problems

In this study, the objective is to minimize the weight of the structure while satisfying some constraints on stresses and/or buckling and/or deflection and/or natural frequencies. The design variables are cross-sectional areas of structural elements. The mathematical formulation of these problems is expressed as follows:

$$\begin{aligned} \text{Find } \quad & \{X\} = [x_1, x_2, \dots, x_{ng}] \\ \text{to minimize } \quad & W(\{X\}) = \sum_{i=1}^{nm} \rho_i A_i L_i \\ \text{subjected to : } \quad & \begin{cases} g_j(\{X\}) \leq 0, j = 1, 2, \dots, nc \\ x_{i \min} \leq x_i \leq x_{i \max} \end{cases} \end{aligned} \quad (17.1)$$

where  $\{X\}$  is a vector containing the design variables;  $ng$  is the number of design variables;  $W(\{X\})$  is the weight of the structure;  $nm$  is the number of elements of the structure;  $\rho_i$ ,  $A_i$ , and  $L_i$  denote the material density, cross-sectional area, and the length of the  $i$ th member, respectively;  $x_{i \min}$  and  $x_{i \max}$  are the lower and upper bounds of the design variable  $x_i$ , respectively;  $g_j(\{X\})$  denotes design constraints; and  $nc$  is the number of constraints.

To handle the constraints, the well-known penalty approach is employed. Thus, the objective function is redefined as follows:

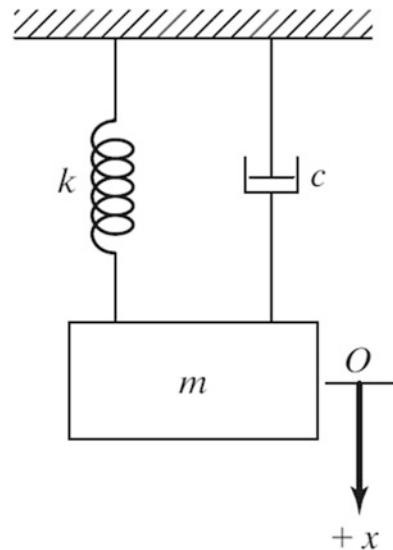
$$f(\{X\}) = (1 + \varepsilon_1 \cdot v)^{\varepsilon_2} \times W(\{X\}), \quad v = \sum_{j=1}^{nc} \max[0, g_j(\{X\})] \quad (17.2)$$

where  $v$  denotes the sum of the violations of the design constraints. The constant  $\varepsilon_1$  is set equal to 1 while  $\varepsilon_2$  starts from 1.5 and linearly increases to 3. Such a scheme penalizes 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 with no violation.

## 17.3 The Damped Free Vibration

A vibration is the oscillating motion of a particle or a body about a position of equilibrium. In general, there are two types of vibrations: (1) free vibration and (2) forced vibration. When the motion is maintained by the restoring forces only, the vibration is said to be a free vibration, and when a time-dependent force is applied to the system, the resulting motion is described as a forced vibration. In the study of a vibrating system, the effects of friction can be neglected resulting in an undamped vibration. However, all vibrations are actually damped to some degree by friction forces. These forces can be caused by dry friction, or Coulomb friction, between rigid bodies, by fluid friction when a rigid body moves in a fluid, or by

**Fig. 17.1** Free vibration of a system with damping



internal friction between the molecules of a seemingly elastic body. In this section, the free vibration of single degree of freedom systems with viscous damping is studied. The viscous damping is caused by fluid friction at low and moderate speeds. Viscous damping is characterized by the fact that the friction force is directly proportional and opposite to the velocity of the moving body (Beer et al. [7]).

The vibrating motion of a body or system of mass  $m$  having viscous damping can be characterized by a block and a spring of constant  $k$  that is shown in Fig. 17.1. The effect of damping is provided by the dashpot connected to the block, and the magnitude of the friction force exerted on the plunger by the surrounding fluid is equal to  $cx$ . ( $c$  is the coefficient of viscous damping, and its value depends on the physical properties of the fluid and the construction of the dashpot). If the block is displaced a distance  $x$  from its equilibrium position, the equation of motion can be expressed as:

$$m\ddot{x} + c\dot{x} + kx = 0 \quad (17.3)$$

Before presenting the solutions for this differential equation, we define the critical damping coefficient  $c_c$  as:

$$c_c = 2m\omega_n \quad (17.4)$$

$$\omega_n = \sqrt{\frac{k}{m}} \quad (17.5)$$

where  $\omega_n$  is the natural circular frequency of the vibration.

Depending on the value of the coefficient of viscous damping, three different cases of damping can be distinguished: (1) over-damped system ( $c > c_c$ ), (2) critically

damped system ( $c = c_c$ ), and (3) under-damped system ( $c < c_c$ ). The solutions of over-damped and critically damped systems correspond to a nonvibratory motion. Therefore, the system only oscillates and returns to its equilibrium position when  $c < c_c$ .

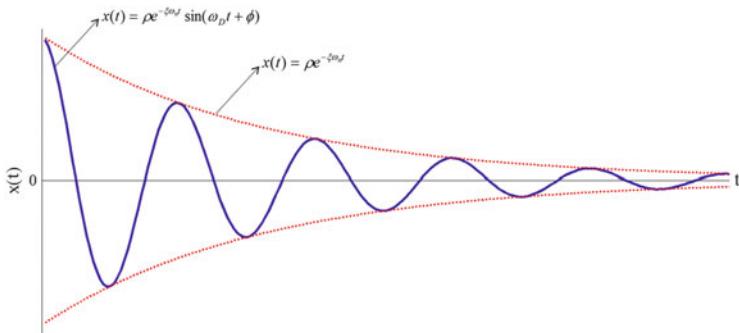
The solution of Eq. (17.3) for under-damped system is as follows:

$$x(t) = \rho e^{-\xi \omega_n t} \sin(\omega_D t + \phi) \quad (17.6)$$

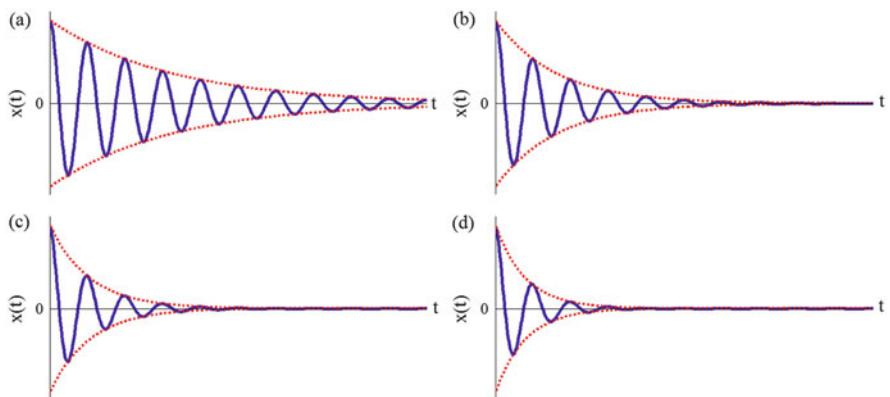
$$\omega_D = \omega_n \sqrt{1 - \xi^2} \quad (17.7)$$

$$\xi = \frac{c}{2m\omega_n} \quad (17.8)$$

where  $\rho$  and  $\phi$  are constants generally determined from the initial conditions of the problem and  $\omega_D$  and  $\xi$  are damped natural frequency and damping ratio, respectively. Equation (17.6) is shown in Fig. 17.2, and the effect of damping ratio on vibratory motion is illustrated in Fig. 17.3.



**Fig. 17.2** Vibrating motion of under-damped system



**Fig. 17.3** Free vibration of systems with four levels of damping: (a)  $\xi = 5\%$ , (b)  $\xi = 10\%$ , (c)  $\xi = 15\%$ , and (d)  $\xi = 20\%$

## 17.4 A New Metaheuristic Algorithm Based on the Vibrating Particles System

The vibrating particles system is a metaheuristic method inspired by the free vibration of single degree of freedom systems with viscous damping. The VPS involves a number of candidate solutions which represent the particles system. The particles are initialized randomly in an  $n$ -dimensional search space and gradually approach to their equilibrium positions. The pseudo code of VPS is provided in Fig. 17.4.

The steps of the VPS are as follows:

### Step 1: Initialization

The VPS parameters are set and the initial positions of all particles are determined randomly in an  $n$ -dimensional search space.

### Step 2: Evaluation of Candidate Solutions

The objective function value is calculated for each particle.

### Step 3: Updating the Particle Positions

For each particle, three equilibrium positions with different weights are defined that the particle tends to approach: (1) the best position achieved so far across the entire population (*HB*), (2) a good particle (*GP*), and (3) a bad particle (*BP*). In order to select the *GP* and *BP* for each candidate solution, the current population is sorted according to their objective function values in an increasing order, and then *GP* and *BP* are chosen randomly from the first and second half, respectively.

Figure 17.3 shows the important effect of damping level in the vibration. In order to model this phenomenon in the optimization algorithm, a descending function that is a function of the number of iterations is proposed as follows:

---

```

procedure Vibrating Particles System (VPS)
    Initialize algorithm parameters
    Initial positions are created randomly
    The values of objective function are evaluated and HB is stored
    While maximum iterations is not fulfilled
        for each particle
            The GP and BP are chosen
            if P<rand
                 $w_3=0$  and  $w_2=1-w_1$ 
            end if
            for each component
                New location is obtained by Eq. (17.10)
            end for
            Violated components are regenerated by harmony search-based handling approach
        end for
        The values of objective function are evaluated and HB is updated
    end while
end procedure

```

---

**Fig. 17.4** Pseudo code of the vibrating particles system algorithm

$$D = \left( \frac{iter}{iter_{max}} \right)^{-\alpha} \quad (17.9)$$

where  $iter$  is the current iteration number and  $iter_{max}$  is the total number of iterations for optimization process.  $\alpha$  is a constant, and Fig. 17.5 shows the effect of this parameter on  $D$ .

According to the mentioned concepts, the positions are updated by:

$$\begin{aligned} x_i^j &= w_1 \cdot [D \cdot A \cdot rand1 + HB^j] + w_2 \cdot [D \cdot A \cdot rand2 + GP^j] \\ &\quad + w_3 \cdot [D \cdot A \cdot rand3 + BP^j] \end{aligned} \quad (17.10)$$

$$A = [w_1 \cdot (HB^j - x_i^j)] + [w_2 \cdot (GP^j - x_i^j)] + [w_3 \cdot (BP^j - x_i^j)] \quad (17.11)$$

$$w_1 + w_2 + w_3 = 1 \quad (17.12)$$

where  $x_i^j$  is the  $j$ th variable of particle  $i$ ;  $w_1$ ,  $w_2$ , and  $w_3$  are three parameters to measure the relative importance of  $HB$ ,  $GP$ , and  $BP$ , respectively; and  $rand1$ ,  $rand2$ , and  $rand3$  are random numbers uniformly distributed in the range of  $[0, 1]$ . The effects of the parameters  $A$  and  $D$  in Eq. (17.10) are similar to those of  $\rho$  and  $e^{-\xi\omega_n t}$  in Eq. (17.6), respectively. Also, the value of  $\sin(\omega_D t + \varphi)$  is considered unity in Eq. (17.10) ( $x(t) = \rho e^{-\xi\omega_n t}$  is shown in Fig. 17.2 by red lines).

A parameter like  $p$  within  $(0, 1)$  is defined, which specifies whether the effect of  $BP$  must be considered in updating position or not. For each particle,  $p$  is compared with  $rand$  (a random number uniformly distributed in the range of  $[0, 1]$ ) and if  $p < rand$ , then  $w_3 = 0$  and  $w_2 = 1 - w_1$ .

Three essential concepts consisting of self-adaptation, cooperation, and competition are considered in this algorithm. Particles move toward  $HB$  so the

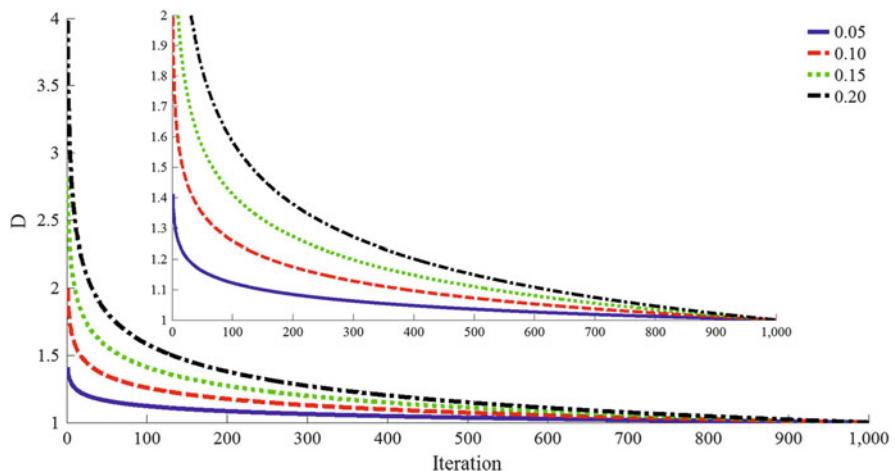


Fig. 17.5 The influence of  $\alpha$  on  $D$  function

self-adaptation is provided. Any particle has the chance to have influence on the new position of the other one, so the cooperation between the particles is supplied. Because of the  $p$  parameter, the influence of  $GP$  (good particle) is more than that of  $BP$  (bad particle), and therefore the competition is provided.

#### **Step 4: Handling the Side Constraints**

The particle moves in the search space to find a better result and it may violate the side constraints. If any component of the system violates a boundary, it must be regenerated by harmony search-based side constraint handling approach (Kaveh and Talatahari [8]). In this technique, there is a possibility like  $HMCR$  (harmony memory considering rate) that specifies whether the violating component should be regenerated considering the corresponding component of the historically best position of a random particle or it should be determined randomly in the search space. Moreover, if the component of a historically best position is selected, there is a possibility like  $PAR$  (pitch adjusting rate) that specifies whether this value should be changed with the neighboring value or not.

#### **Step 5: Terminating Criterion Controlling**

Steps 2 through 4 are repeated until a termination criterion is fulfilled. Any terminating condition can be considered, and in this study the optimization process is terminated after a fixed number of iterations.

## **17.5 Search Behavior of the Vibrating Particles System Algorithm**

In order to evaluate the effect of the algorithm parameters on the optimization results, a spatial 120-bar dome-shaped truss (Sect. 17.6.1) is considered as a benchmark. The effect of the population size, the maximum number of structural analyses (population size  $\times$  total number of iterations),  $\alpha$ ,  $p$ ,  $w_1$ , and  $w_2$  are investigated in this section. In the first step, these parameters are set to 20, 20000, 0.15, 70 %, 0.3, and 0.3, respectively, and then their proper values are obtained one after another.

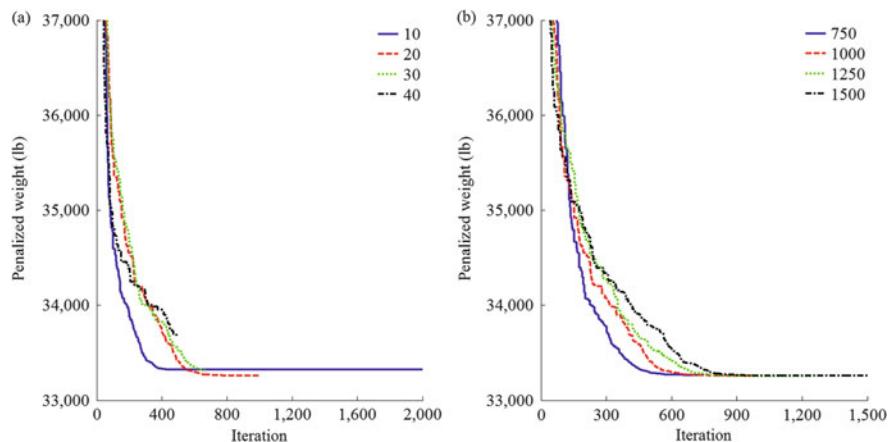
Tables 17.1 and 17.2 summarize the statistical results achieved for different values of population size (10, 20, 30, and 40) and total number of iterations (750, 1000, 1250, and 1500), respectively. As it can be seen from Table 17.1, when population size is

**Table 17.1** Sensitivity analysis on population size

	10	20	30	40
Best optimized weight (lb)	33,262.75	33,250.27	33,255.65	33,471.79
Worst optimized weight (lb)	33,413.99	33,282.16	33,432.60	33,903.75
Average optimized weight (lb)	33,322.28	33,258.58	33,315.24	33,668.73
Standard deviation on average weight (lb)	51.49	10.31	62.77	130.71
Number of structural analyses for the best design	8920	19,780	13,060	9780
Average number of structural analysis	10,106	16,930	12,746	7958

**Table 17.2** Sensitivity analysis on maximum number of iterations

	750	1000	1250	1500
Best optimized weight (lb)	33,251.34	33,250.27	33,250.83	33,250.24
Worst optimized weight (lb)	33,283.49	33,282.16	33,275.18	33,265.92
Average optimized weight (lb)	33,263.52	33,258.59	33,255.03	33,257.01
Standard deviation on average weight (lb)	12.75	10.31	7.71	5.97
Number of structural analyses for the best design	12,620	19,780	20,800	22,320
Average number of structural analysis	13,094	16,930	18,646	20,802

**Fig. 17.6** Sensitivity analysis on (a) population size and (b) maximum number of iterations

20, the VPS has better performance in terms of the best weight, worst weight, average optimized weight, and standard deviation on average weight. Table 17.2 demonstrates that considering 1500 iterations is the most efficient value for the total number of iterations. The corresponding average convergence curves are shown in Fig. 17.6. Since the maximum number of structural analyses is set to 20,000 in Fig. 17.6a, the total number of iterations for 10, 20, 30, and 40 particles are 2000, 1000, 667, and 500, respectively, considered as the termination criterion.

Performance of the VPS with different values of  $\alpha$  (0.05, 0.1, 0.15, and 0.2) and  $p$  (60 %, 70 %, 80 %, and 90 %) are compared in Tables 17.3 and 17.4, respectively. When  $\alpha$  is considered as 0.1, the best weight is achieved; however, comparison of the other variables shows that 0.05 is generally the most suitable value for  $\alpha$ . It can be concluded from Table 17.4 that 70 % is the most efficient value for  $p$ . Average convergence histories are depicted in Fig. 17.7. As mentioned before, the influence of damping level on vibration is similar to the effect of  $\alpha$  on particles convergence as can be seen in Fig. 17.7a. To make the curves of Fig. 17.7b clearer, the magnified version of lower part is also shown.

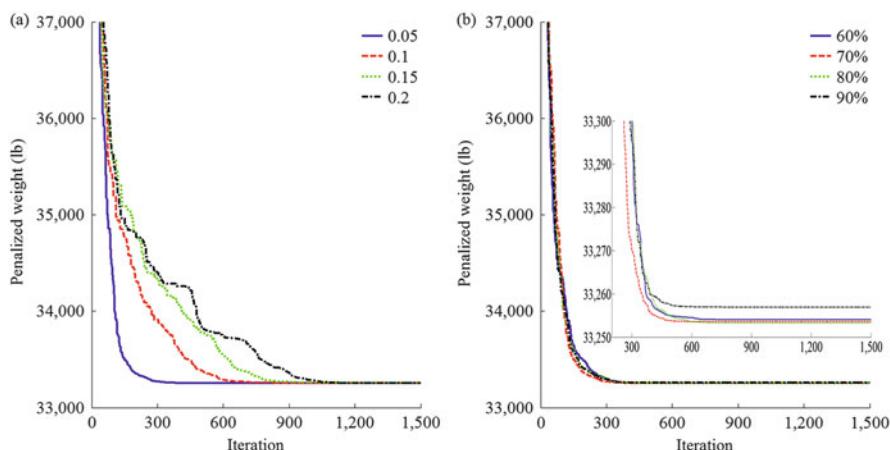
Results of sensitivity analysis on  $w_1$  and  $w_2$  are shown in Tables 17.5 and 17.6. According to the statistical results reported in these tables, the most suitable

**Table 17.3** Sensitivity analysis on  $\alpha$ 

	0.05	0.1	0.15	0.2
Best optimized weight (lb)	33,249.98	33,249.76	33,250.24	33,249.98
Worst optimized weight (lb)	33,262.74	33,283.46	33,265.92	33,261.82
Average optimized weight (lb)	33,253.56	33,254.91	33,257.01	33,254.02
Standard deviation on average weight (lb)	4.36	10.31	5.97	3.89
Number of structural analyses for the best design	8280	17,500	22,320	24,180
Average number of structural analyses	9846	17,794	20,802	24,834

**Table 17.4** Sensitivity analysis on  $p$ 

	60 %	70 %	80 %	90 %
Best optimized weight (lb)	33,250.06	33,249.98	33,250.89	33,250.08
Worst optimized weight (lb)	33,260.61	33,262.74	33,257.86	33,281.97
Average optimized weight (lb)	33,254.03	33,253.56	33,253.23	33,256.93
Standard deviation on average weight (lb)	4.27	4.36	2.54	9.75
Number of structural analyses for the best design	12,900	8280	10,580	10,740
Average number of structural analysis	11,114	9846	11,910	10,104

**Fig. 17.7** Sensitivity analysis on (a)  $\alpha$  and (b)  $p$ 

performance of the VPS is obtained when the value of 0.3 is considered for  $w_1$  and  $w_2$ . Figure 17.8 compares the average convergence curves. It can be seen from Fig. 17.8a that by decreasing the value of  $w_1$  (decreasing the effect of  $HB$  position in updating formula), the exploration is increased and vice versa. In order to make the curves of Fig. 17.8b more clear, the magnified version of lower part is also added.

**Table 17.5** Sensitivity analysis on  $w_1$ 

	0.2	0.25	0.3	0.35
Best optimized weight (lb)	33,386.52	33,250.16	33,249.98	33,252.79
Worst optimized weight (lb)	33,655.38	33,268.67	33,262.74	33,676.11
Average optimized weight (lb)	33,484.02	33,254.51	33,253.56	33,314.71
Standard deviation on average weight (lb)	83.68	5.38	4.36	130.57
Number of structural analyses for the best design	28,820	28,540	8280	5160
Average number of structural analysis	26,456	28,342	9846	6520

**Table 17.6** Sensitivity analysis on  $w_2$ 

	0.2	0.25	0.3	0.35
Best optimized weight (lb)	33,250.63	33,249.67	33,249.98	33,250.62
Worst optimized weight (lb)	33,268.43	33,271.20	33,262.74	33,321.25
Average optimized weight (lb)	33,255.19	33,255.38	33,253.56	33,264.93
Standard deviation on average weight (lb)	5.06	6.49	4.36	22.84
Number of structural analyses for the best design	12,940	12,180	8280	12,880
Average number of structural analysis	11,982	12,196	9846	9954

In summary, the values of population size, the total number of iteration,  $\alpha$ ,  $p$ ,  $w_1$ , and  $w_2$  are set to 20, 1500, 0.05, 70 %, 0.3, and 0.3 for all examples, respectively.

## 17.6 Test Problems and Optimization Results

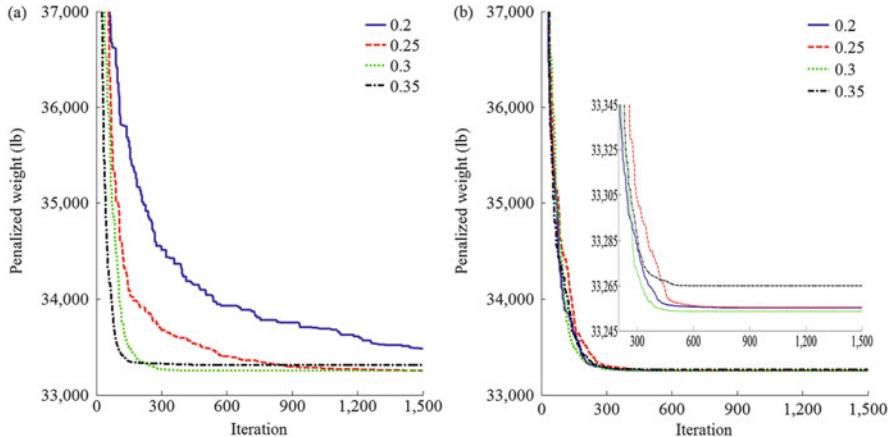
Four skeletal structures are optimized for minimum weight with the cross-sectional areas of the members being the design variables to evaluate the performance of the proposed method. The examples are classified into two groups: the first group consists of two truss structures with the number of truss bars of 120 and 200, respectively, and the second group includes two steel frames having 105 and 168 members, respectively. For all the considered examples, 20 independent optimization runs are carried out as metaheuristic algorithms have stochastic nature and their performance may be sensitive to initial population. The algorithm is coded in MATLAB, and the structures are analyzed using the direct stiffness method by our own codes.

### 17.6.1 A Spatial 120-Bar Dome-Shaped Truss

- The schematic and element grouping of the spatial 120-bar dome truss are shown in Fig. 17.9. The structure is divided into seven groups of elements because of

Table 17.7 Performance comparison for the spatial 120-bar dome-shaped truss structure

Optimal cross-sectional areas (in <sup>2</sup> )						
Element group	CSS (Kaveh and Talatahari [10])	IRO (Kaveh et al. [11])	MSPSO (Talatahari et al. [12])	CBO (Kaveh and Mahdavi [13])	TWO (Kaveh and Zolghadr [14])	WEQ (Kaveh and Bakhtshoori [15])
1	3.027	3.0252	3.0244	3.0273	3.0247	3.0243
2	14.606	14.8354	14.7804	15.1724	14.7261	14.7943
3	5.044	5.1139	5.0567	5.2342	5.1338	5.0618
4	3.139	3.1305	3.1359	3.119	3.1369	3.1358
5	8.543	8.4037	8.4830	8.1038	8.4545	8.4870
6	3.367	3.3315	3.3104	3.4166	3.2946	3.2886
7	2.497	2.4968	2.4977	2.4918	2.4956	2.4967
Weight (lb)	33,251.9	33,256.48	33,251.22	33,286.3	33,250.31	33,249.98
Average optimized weight (lb)	N/A	33,280.85	33,257.29	33,398.5	33,282.64	33,255.55
Standard deviation on average weight (lb)	N/A	N/A	4.29	67.09	25.38	N/A
Number of structural analyses	7000	18,300	15,000	14,960	16,000	19,510
						8280



**Fig. 17.8** Sensitivity analysis on (a)  $w_1$  and (b)  $w_2$

symmetry (for the sake of clarity, not all the element groups are numbered in Fig. 17.9). The modulus of elasticity is 30,450 ksi (210 GPa) and the material density is 0.288 lb/in<sup>3</sup> (7971.810 kg/m<sup>3</sup>). 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 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. Element cross-sectional areas can vary between 0.775 in<sup>2</sup> (5 cm<sup>2</sup>) and 20.0 in<sup>2</sup> (129.032 cm<sup>2</sup>). Displacement limitations of  $\pm 0.1969$  in ( $\pm 5$  mm) are imposed on all nodes in x-, y-, and z-coordinate directions. Constraints on member stresses are imposed according to the provisions of the AISc [9] as follows:

The allowable tensile stresses for tension members are calculated as:

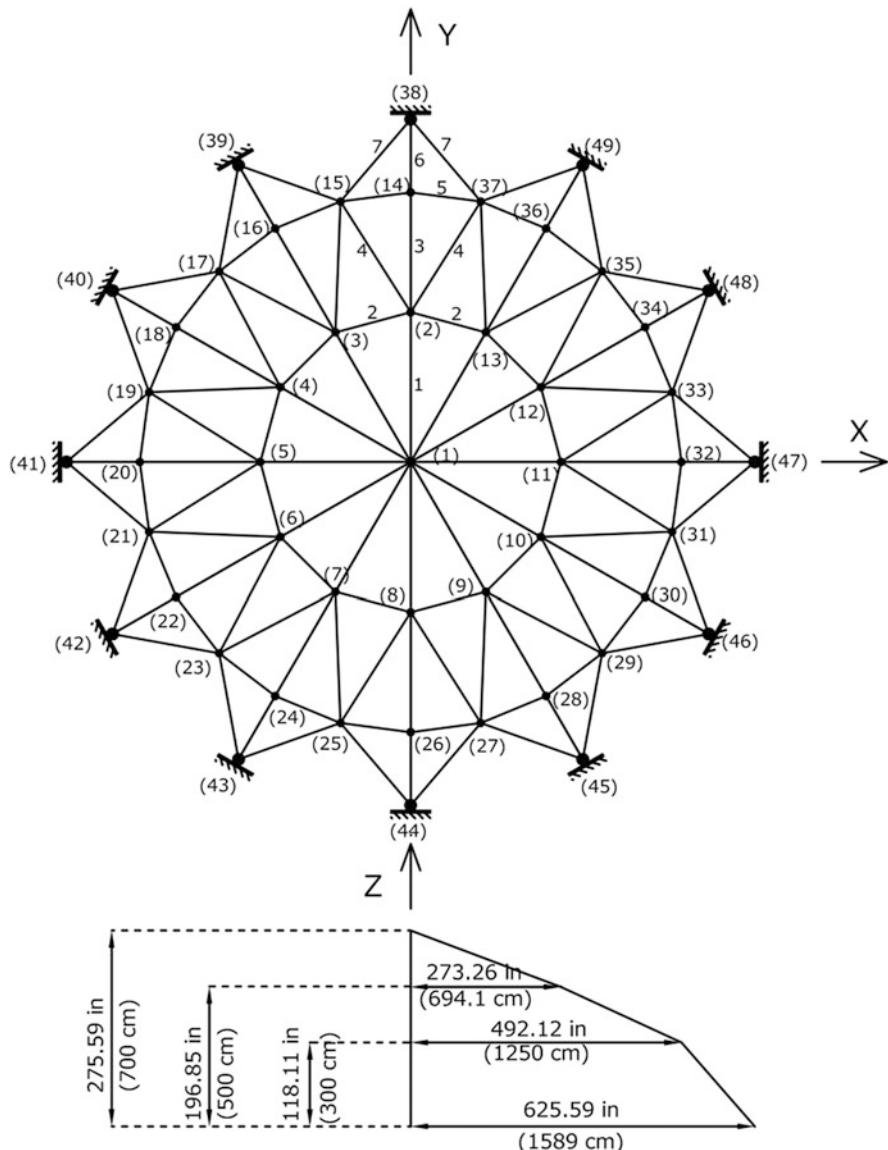
$$\sigma_i^+ = 0.6F_y \quad (17.13)$$

where  $F_y$  is the yield strength.

The allowable stress limits for compression members are calculated depending on two possible failure modes of the members known as elastic and inelastic buckling. Therefore

$$\sigma_i^- = \begin{cases} \left[ \left( 1 - \frac{\lambda_i^2}{2C_c^2} \right) F_y \right] / \left[ \frac{5}{3} + \frac{3\lambda_i}{8C_c} - \frac{\lambda_i^3}{8C_c^3} \right] & \text{for } \lambda_i < C_c \\ \frac{12\pi^2 E}{23\lambda_i^2} & \text{for } \lambda_i \geq C_c \end{cases} \quad (17.14)$$

where  $E$  is the modulus of elasticity;  $\lambda_i$  is the slenderness ratio ( $\lambda_i = kl_i/r_i$ );  $C_c$  denotes the slenderness ratio dividing the elastic and inelastic buckling regions



**Fig. 17.9** Schematic of the spatial 120-bar dome-shaped truss

$C_c = \sqrt{2\pi^2 E / F_y}$ ;  $k$  is the effective length factor ( $k$  is set equal to 1 for all truss members);  $L_i$  is the member length; and  $r_i$  is the minimum radius of gyration.

This truss was previously optimized by CSS (charged system search algorithm) (Kaveh and Talatahari [10]), IRO (improved ray optimization) (Kaveh et al. [11]),

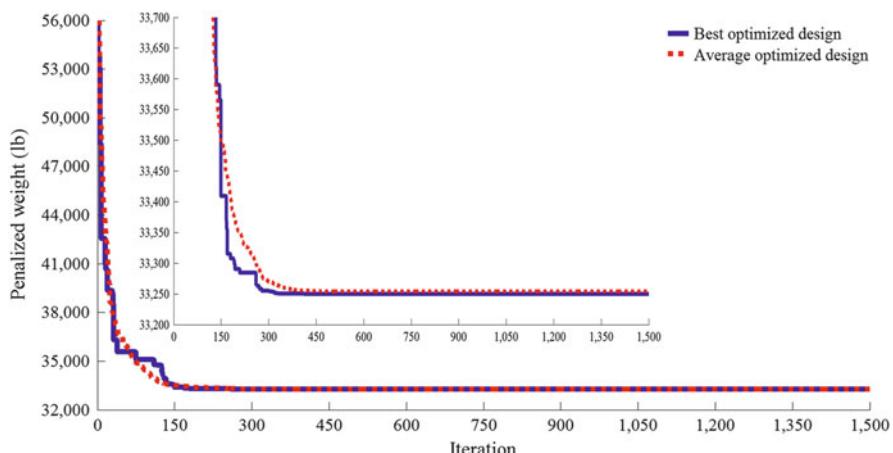
MSPSO (multi-stage particle swarm optimization) (Talatahari et al. [12]), CBO (colliding bodies optimization) (Kaveh and Mahdavi [13]), TWO (tug of war optimization) (Kaveh and Zolghadr [14]), and WEO (water evaporation optimization) (Kaveh and Bakhshpoori [15]).

Comparison of the optimal designs obtained by this work with those of the other researches is given in Table 17.7. It can be seen that the lightest design (i.e., 33,249.98 lb) and the best average optimized weight (i.e., 33,253.56 lb) are found by the proposed method. The VPS converges to the optimum solution after 8280 analyses. The CSS gives the best result as 33,251.9 lb in 7000 analyses. However, the VPS achieves this result after 6400 analyses. Figure 17.10 compares the convergence curves of the best and the average results obtained by the proposed method.

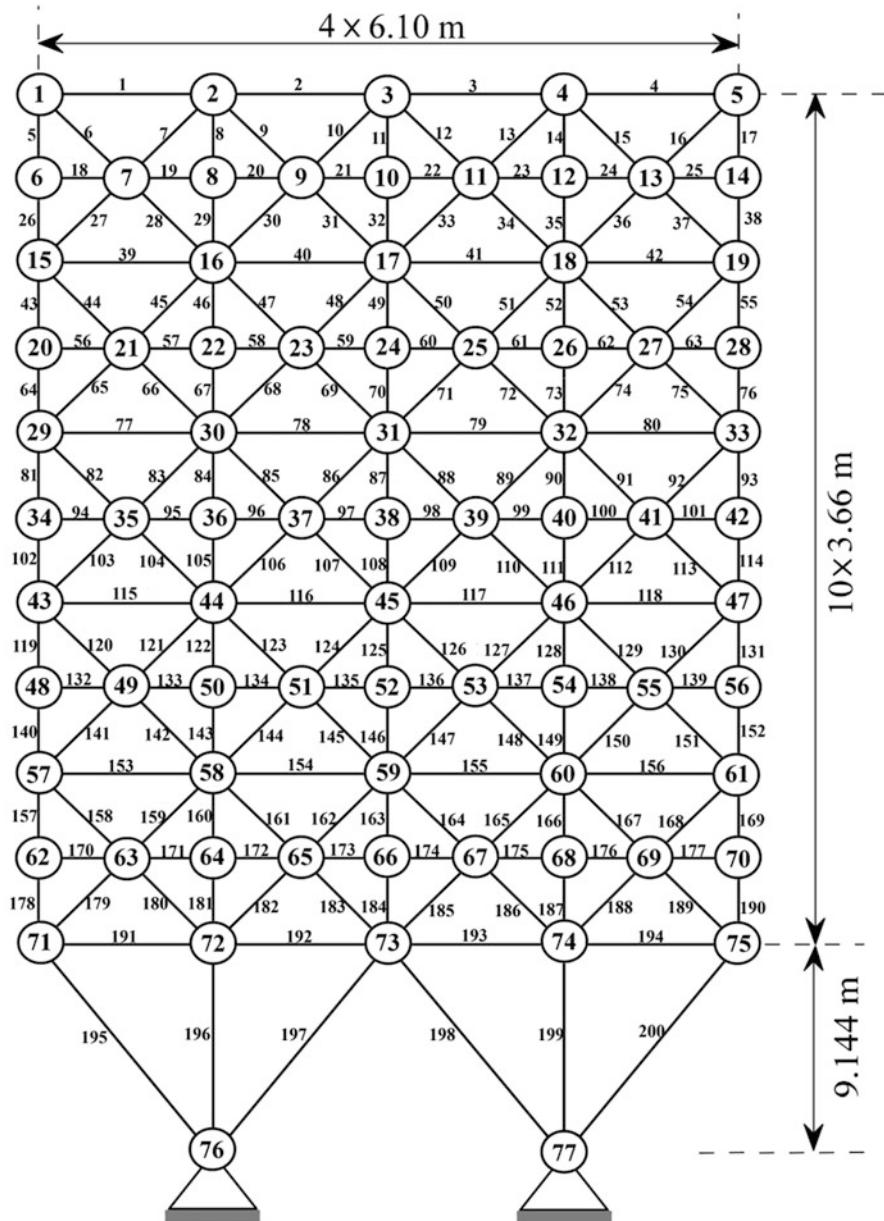
### 17.6.2 A 200-Bar Planar Truss

The second structural optimization problem solved in this chapter is the optimal design of the 200-bar planar truss schematized in Fig. 17.11. Due to the symmetry, the elements are divided into 29 groups. The modulus of elasticity and the material density of members are 210 GPa and  $7860 \text{ kg/m}^3$ , respectively. Nonstructural masses of 100 kg are attached to the upper nodes. A lower bound of  $0.1 \text{ cm}^2$  is assumed for the cross-sectional areas. The first three natural frequencies of the structure must satisfy the following limitations ( $f_1 \geq 5 \text{ Hz}$ ,  $f_2 \geq 10 \text{ Hz}$ ,  $f_3 \geq 15 \text{ Hz}$ ).

Table 17.8 presents the results of the optimal designs utilizing CSS–BBCB (a hybridization of the charged system search and the Big Bang–Big Crunch algorithms with trap recognition capability) (Kaveh and Zolghadr [16]), CBO



**Fig. 17.10** Convergence curves obtained for the 120-bar dome-shaped truss problem



**Fig. 17.11** Schematic of the 200-bar planar truss

(colliding bodies optimization) (Kaveh and Ilchi Ghazaan [17]), ECBO (enhanced colliding bodies optimization) (Kaveh and Ilchi Ghazaan [17]), CBO–PSO (a hybrid of CBO and PSO algorithms) (Kaveh and Mahdavi [18]), and the

**Table 17.8** Performance comparison for the 200-bar planar truss structure

Element group	Members in the group	Areas (cm <sup>2</sup> )				
		CSS-BBBC (Kaveh and Zolghadr [28])	CBO (Kaveh and Ilchi Ghazaan [29])	ECBO (Kaveh and Ilchi Ghazaan [29])	CBO-PSO (Kaveh and Mahdavi [30])	Present work [1]
1	1, 2, 3, 4	0.2934	0.3059	0.2993	0.2797	0.3031
2	5, 8, 11, 14, 17	0.5561	0.4476	0.4497	0.6968	0.4496
3	19, 20, 21, 22, 23, 24	0.2952	0.1000	0.1000	0.1000	0.1002
4	18, 25, 56, 63, 94, 101, 132, 139, 170, 177	0.1970	0.1001	0.1000	0.1000	0.1000
5	26, 29, 32, 35, 38	0.8340	0.4944	0.5137	0.5796	0.5086
6	6, 7, 9, 10, 12, 13, 15, 16, 27, 28, 30, 31, 33, 34, 36, 37	0.6455	0.8369	0.7914	0.8213	0.8204
7	39, 40, 41, 42	0.1770	0.1001	0.1013	0.1279	0.1000
8	43, 46, 49, 52, 55	1.4796	1.5514	1.4129	1.0152	1.4210
9	57, 58, 59, 60, 61, 62	0.4497	0.1000	0.1019	0.1000	0.1002
10	64, 67, 70, 73, 76	1.4556	1.5286	1.6460	1.5647	1.5900
11	44, 45, 47, 48, 50, 51, 53, 54, 65, 66, 68, 69, 71, 72, 74, 75	1.2238	1.1547	1.1532	1.6465	1.1530
12	77, 78, 79, 80	0.2739	0.1000	0.1000	0.2296	0.1277
13	81, 84, 87, 90, 93	1.9174	2.9980	3.1850	2.9007	2.9160
14	95, 96, 97, 98, 99, 100	0.1170	0.1017	0.1034	0.1000	0.1009
15	102, 105, 108, 111, 114	3.5535	3.2475	3.3126	3.0133	3.2826
16	82, 83, 85, 86, 88, 89, 91, 92, 103, 104, 106, 107, 109, 110, 112, 113	1.3360	1.5213	1.5920	1.6142	1.5856
17	115, 116, 117, 118	0.6289	0.3996	0.2238	0.2755	0.2794
18	119, 122, 125, 128, 131	4.8335	4.7557	5.1227	5.0951	5.0680
19	133, 134, 135, 136, 137, 138	0.6062	0.1002	0.1050	0.1000	0.1004

20	140,143,146,149,152	5.4393	5.1359	5.3707	5.5172	5.4760
21	120, 121, 123, 124, 126, 127, 129, 130, 141, 142, 144, 145, 147, 148, 150, 151	1.8435	2.1181	2.0645	2.2032	2.1169
22	153, 154, 155, 156	0.8955	0.9200	0.5443	0.8659	0.6939
23	157, 160, 163, 166, 169	8.1759	7.3084	7.6497	7.6477	7.6912
24	171, 172, 173, 174, 175, 176	0.3209	0.1185	0.1000	0.1000	0.1332
25	178, 181, 184, 187, 190	10.98	7.6901	7.6754	8.1273	7.9972
26	158, 159, 161, 162, 164, 165, 167, 168, 179, 180, 182, 183, 185, 186, 188, 189	2.9489	3.0895	2.7178	2.9665	2.7859
27	191, 192, 193, 194	10.5243	10.6462	10.8141	10.2386	10.4331
28	195, 197, 198, 200	20.4271	20.7190	21.6349	20.6364	21.2289
29	196, 199	19.0983	11.7463	10.3520	11.6468	10.7392
Weight (kg)		2298.61	2161.15	2158.08	2195.469	2156.62
Average optimized weight (kg)		N/A	2447.52	2159.93	N/A	2159.46
Standard deviation on average weight (kg)		N/A	301.29	1.57	N/A	2.79
Number of struc- tural analyses		N/A	10.500	14,700	9000	16,420

proposed method. The weight of the best result obtained by VPS is 2156.62 kg that is the best among the compared methods. Moreover, the average optimized weight for 20 independent optimization runs of the VPS is 2159.46 kg which is less than those of all other methods. The first three natural frequencies of the structure for the best design are 5.0000 Hz, 12.2086 Hz, and 15.0153 Hz. The proposed method requires 16,420 structural analyses to find the optimum solution, while CBO, ECBO, and CBO-PSO require 10,500, 14,700, and 9000 structural analyses, respectively. It should be noted that the designs found by VPS at 9000th, 10,500th, and 14,700th analyses are 2158.35 kg, 2158.06 kg, and 2157.72 kg, respectively. Comparison of the convergence rates between the best and the average curves of VPS is illustrated in Fig. 17.12.

### 17.6.3 A 3-Bay 15-Story Frame Structure

Figure 17.13 represents the schematic of the 3-bay 15-story frame. The applied loads and the numbering of member groups are also shown in this figure. The modulus of elasticity is 29 Ms (200 GPa) and the yield stress is 36 ksi (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 non-braced length for each beam member is specified as one-fifth of the span length. Limitation on displacement and strength are imposed according to the provisions of the AISC [19] as follows:

- (a) Maximum lateral displacement

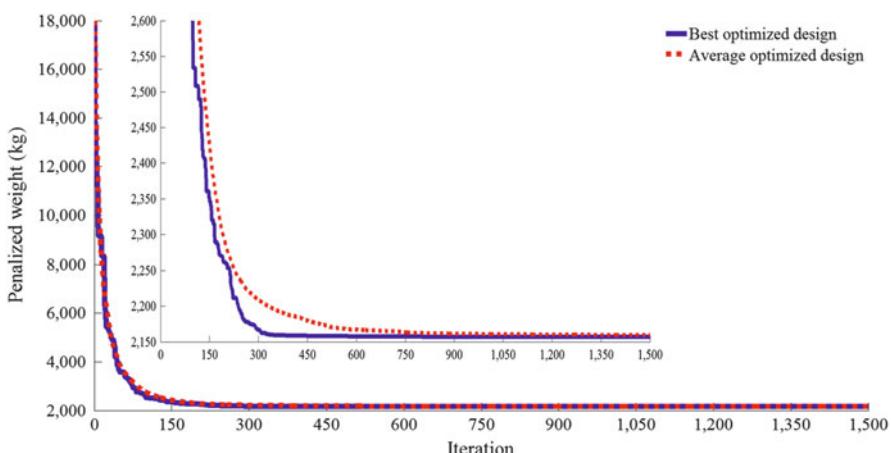
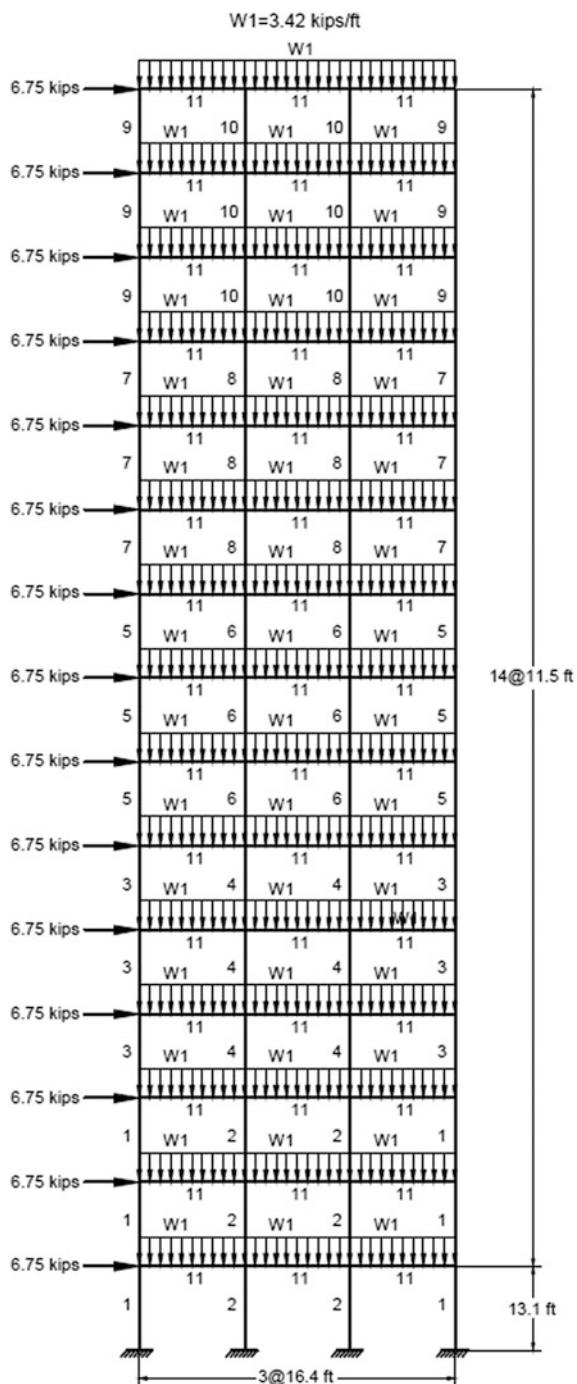


Fig. 17.12 Convergence curves obtained for the 200-bar planar truss problem

**Fig. 17.13** Schematic of the 3-bay 15-story frame



$$\frac{\Delta_T}{H} - R \leq 0 \quad (17.15)$$

where  $\Delta_T$  is the maximum lateral displacement;  $H$  is the height of the frame structure; and  $R$  is the maximum drift index which is equal to 1/300.

(b) The inter-story displacements

$$\frac{d_i}{h_i} - R_I \leq 0, \quad i = 1, 2, \dots, ns \quad (17.16)$$

where  $d_i$  is the inter-story drift;  $h_i$  is the story height of the  $i$ th floor;  $ns$  is the total number of stories; and  $R_I$  is the inter-story drift index (1/300).

(c) Strength constraints

$$\begin{cases} \frac{Pu}{2\varphi_c P_n} + \frac{M_u}{\varphi_b M_n} - 1 \leq 0, & \text{for } \frac{Pu}{\varphi_c P_n} < 0.2 \\ \frac{Pu}{\varphi_c P_n} + \frac{8M_u}{9\varphi_b M_n} - 1 \leq 0, & \text{for } \frac{Pu}{\varphi_c P_n} \geq 0.2 \end{cases} \quad (17.17)$$

where  $P_u$  is the required axial strength (tension or compression);  $P_n$  is the nominal axial strength (tension or compression);  $\varphi_c$  is the resistance factor ( $\varphi_c = 0.9$  for tension,  $\varphi_c = 0.85$  for compression);  $M_u$  is the required flexural strengths;  $M_n$  is the nominal flexural strengths; and  $\varphi_b$  denotes the flexural resistance reduction factor ( $\varphi_b = 0.90$ ).

The nominal tensile strength for yielding in the gross section is calculated by:

$$P_n = A_g \cdot F_y \quad (17.18)$$

The nominal compressive strength of a member is computed as:

$$P_n = A_g \cdot F_{cr} \quad (17.19)$$

where

$$\begin{cases} F_{cr} = (0.658 \lambda_c^2) F_y, & \text{for } \lambda_c \leq 1.5 \\ F_{cr} = \left(\frac{0.877}{\lambda_c^2}\right) F_y, & \text{for } \lambda_c > 1.5 \end{cases} \quad (17.20)$$

$$\lambda_c = \frac{kl}{r\pi} \sqrt{\frac{F_y}{E}} \quad (17.21)$$

where  $A_g$  is the cross-sectional area of a member and  $k$  is the effective length factor that is calculated by (Dumonteil [20]):

$$k = \sqrt{\frac{1.6G_A G_B + 4.0(G_A + G_B) + 7.5}{G_A + G_B + 7.5}} \quad (17.22)$$

where  $G_A$  and  $G_B$  are stiffness ratios of columns and girders at the two end joints, A and B, of the column section, respectively.

Also, in this example the sway of the top story is limited to 9.25 in (23.5 cm).

Table 17.9 presents the comparison of the results of the present algorithm with the outcomes of other algorithms. The proposed method yields the least weight for this example, which is 86,985 lb. The other design weights are 95,850 lb by HPSACO (a hybrid algorithm of harmony search, particle swarm and ant colony) (Kaveh and Talatahari [21]), 97,689 lb by HBB–BC (a hybrid Big Bang–Big Crunch optimization) (Kaveh and Talatahari [22]), 93,846 lb by ICA (imperialist competitive algorithm) (Kaveh and Talatahari [23]), 92,723 lb by CSS (Kaveh and Talatahari [24]), 93,795 lb by CBO (Kaveh and Ilchi Ghazaan [25]), 86,986 lb by ECBO (Kaveh and Ilchi Ghazaan [25]), and 93,315 lb by ES–DE (eagle strategy with differential evolution) (Talatahari et al. [26]). The best design of VPS has been achieved in 19,600 analyses. It should be noted that the proposed method achieved about 92,000 lb (the best weight among the other methods except ECBO) after 10,800 structural analyses. Figure 17.14 provides the convergence rates of the best and average results found by the VPS. Element stress ratio and inter-story drift evaluated at the best design optimized by VPS are shown in Fig. 17.15. The maximum stress ratio is 99.88 % and the maximum inter-story drift is 45.41.

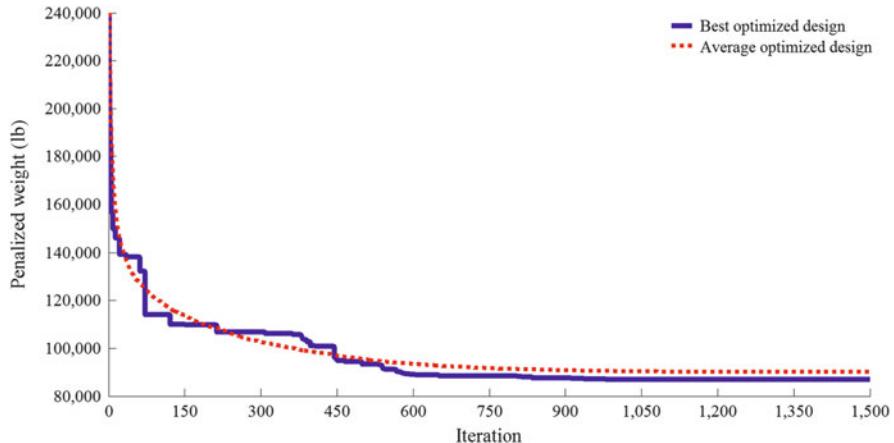
#### 17.6.4 A 3-Bay 24-Story Frame Structure

The last structural optimization problem solved in this chapter is the weight minimization of the 3-bay 24-story frame schematized in Fig. 17.16. Frame members are collected in 20 groups (16 column groups and 4 beam groups). 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. The material has a modulus of elasticity equal to  $E = 29.732$  Msi (205 GPa) and a yield stress of  $f_y = 33.4$  ksi (230.3 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  $k_y = 1.0$ . All columns and beams are considered as non-braced along their lengths. Similar to the previous example, the frame is designed following the LRFD–AISC specification and uses an inter-story drift displacement constraint (AISC [19]).

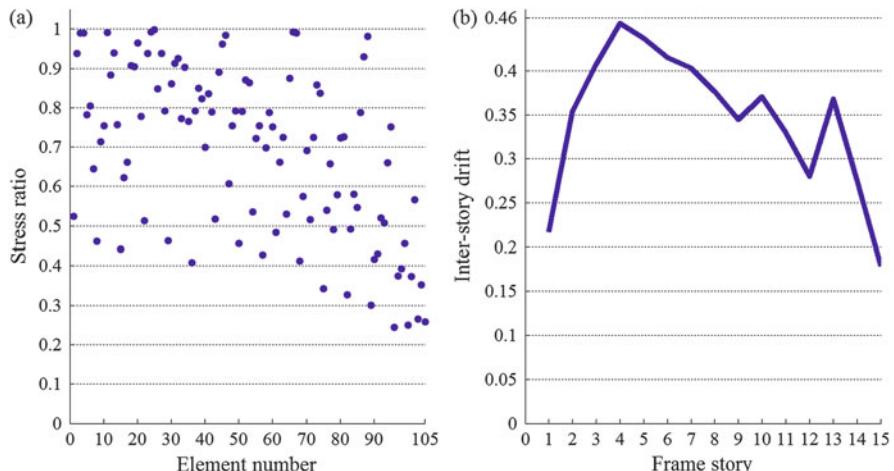
This steel frame structure was previously optimized by ACO (ant colony optimization) (Camp et al. [27]), HS (harmony search) (Degertekin [28]), ICA (Kaveh and Talatahari [23]), CSS (Kaveh and Talatahari [24]), CBO (Kaveh and Ilchi Ghazaan [25]), ECBO (Kaveh and Ilchi Ghazaan [25]), and ES–DE (Talatahari et al. [26]). Table 17.10 presents a comparison between the results of the optimal

Table 17.9 Performance comparison for the 3-bay 15-story frame structure

Optimal W-shaped sections						
Element group	HPSACO (Kaveh and Talatahari [21])	HBB-BC (Kaveh and Talatahari [22])	ICA (Kaveh and Talatahari [23])	CSS (Kaveh and Talatahari [24])	CBO (Kaveh and Ilchi Ghazaan [25])	ECBO (Kaveh and Ilchi Ghazaan [25])
1	W21 × 111	W24 × 117	W24 × 117	W21 × 147	W24 × 104	W14 × 99
2	W18 × 158	W21 × 132	W21 × 147	W18 × 143	W40 × 167	W27 × 161
3	W10 × 88	W12 × 95	W27 × 84	W12 × 87	W27 × 84	W12 × 79
4	W30 × 116	W18 × 119	W27 × 114	W30 × 108	W27 × 114	W24 × 104
5	W21 × 83	W21 × 93	W14 × 74	W18 × 76	W21 × 68	W14 × 61
6	W24 × 103	W18 × 97	W18 × 86	W24 × 103	W30 × 90	W30 × 90
7	W21 × 55	W18 × 76	W12 × 96	W21 × 68	W8 × 48	W14 × 48
8	W27 × 114	W18 × 65	W24 × 68	W14 × 61	W21 × 68	W14 × 61
9	W10 × 33	W18 × 60	W10 × 39	W18 × 35	W14 × 34	W14 × 30
10	W18 × 46	W10 × 39	W12 × 40	W10 × 33	W8 × 35	W10 × 38
11	W21 × 44	W21 × 48	W21 × 44	W21 × 44	W21 × 50	W12 × 40
Weight (lb)	95,850	97,689	93,846	92,723	93,795	86,986
Average optimized weight (lb)	N/A	N/A	N/A	N/A	98,738	88,410
Standard deviation on average weight (lb)	N/A	N/A	N/A	N/A	N/A	98,531
Number of structural analyses	6800	9900	6000	5000	9520	9000
						10,000
						19,600
						2533



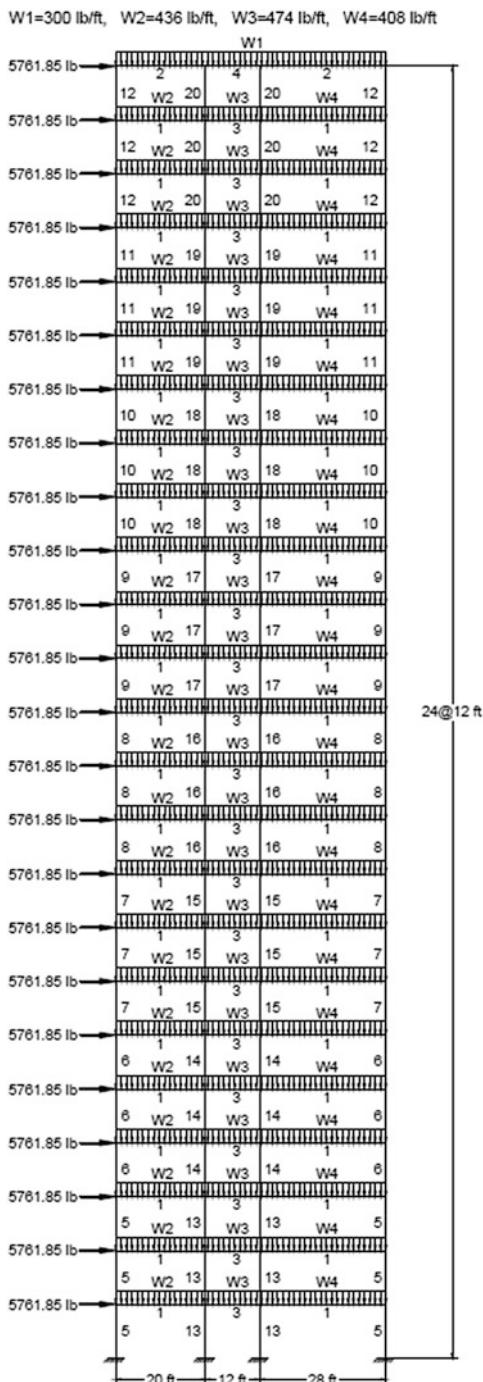
**Fig. 17.14** Convergence curves obtained for the 3-bay 15-story frame structure



**Fig. 17.15** Constraint margins for the best design obtained by VPS for the 3-bay 15-story frame problem: (a) element stress ratio and (b) inter-story drift

designs reported in the literature and the present work [1]. The lightest design (i.e., 201,618 lb) is found by ECBO algorithm and after that the best design belongs to VPS (i.e., 202,998 lb). The best design has been achieved at 16,220 analyses for VPS, and it obtained 209,532 lb after 8800 analyses, which is the best result compared to the weight achieved by the other methods. Figure 17.17 provides the convergence rates of the best and average results found by the proposed method.

**Fig. 17.16** Schematic of the 3-bay 24-story frame



**Table 17.10** Performance comparison for the 3-bay 24-story frame structure

Element group	Optimal W-shaped sections						ECBO (Kaveh and Ilchi Ghazaan [25])	ES-DE (Talatahari et al. [26])	Present work [1]
	ACO (Camp et al. [27])	HS (Degertekin [28])	ICA (Kaveh and Talatahari [23])	CSS (Kaveh and Talatahari [24])	CBO (Kaveh and Ilchi Ghazaan [25])	W30 × 90			
1	W30 × 90	W30 × 90	W30 × 90	W30 × 90	W21 × 50	W8 × 18	W30 × 90	W30 × 90	W30 × 90
2	W8 × 18	W10 × 22	W21 × 50	W21 × 50	W24 × 48	W24 × 55	W6 × 15	W21 × 55	W8 × 18
3	W24 × 55	W18 × 40	W24 × 55	W21 × 48	W12 × 19	W6 × 8.5	W24 × 55	W21 × 48	W21 × 48
4	W8 × 21	W12 × 16	W8 × 28	W12 × 19	W14 × 109	W14 × 176	W14 × 132	W10 × 45	W6 × 8.5
5	W14 × 145	W14 × 176	W14 × 109	W14 × 176	W14 × 176	W14 × 176	W14 × 145	W14 × 145	W14 × 176
6	W14 × 132	W14 × 159	W14 × 145	W14 × 145	W14 × 120	W14 × 132	W14 × 109	W14 × 109	W14 × 145
7	W14 × 132	W14 × 132	W14 × 120	W14 × 109	W14 × 145	W14 × 99	W14 × 99	W14 × 99	W14 × 99
8	W14 × 132	W14 × 109	W14 × 90	W14 × 90	W14 × 82	W14 × 90	W14 × 90	W14 × 145	W14 × 82
9	W14 × 68	W14 × 82	W14 × 74	W14 × 74	W14 × 61	W14 × 74	W14 × 74	W14 × 109	W14 × 82
10	W14 × 53	W14 × 74	W14 × 68	W14 × 61	W14 × 43	W14 × 38	W14 × 38	W14 × 48	W14 × 38
11	W14 × 43	W14 × 34	W14 × 30	W14 × 34	W14 × 34	W14 × 38	W14 × 38	W14 × 38	W14 × 30
12	W14 × 43	W14 × 22	W14 × 38	W14 × 34	W14 × 22	W14 × 22	W14 × 22	W14 × 30	W14 × 30
13	W14 × 145	W14 × 159	W14 × 145	W14 × 145	W14 × 99	W14 × 99	W14 × 99	W14 × 99	W14 × 90
14	W14 × 145	W14 × 132	W14 × 132	W14 × 132	W14 × 109	W14 × 99	W14 × 99	W14 × 132	W14 × 99
15	W14 × 120	W14 × 109	W14 × 99	W14 × 109	W14 × 82	W14 × 99	W14 × 99	W14 × 109	W14 × 99
16	W14 × 90	W14 × 82	W14 × 82	W14 × 82	W14 × 90	W14 × 82	W14 × 82	W14 × 68	W14 × 90
17	W14 × 90	W14 × 61	W14 × 68	W14 × 68	W14 × 74	W14 × 68	W14 × 68	W14 × 68	W14 × 61
18	W14 × 61	W14 × 48	W14 × 48	W14 × 43	W14 × 61	W14 × 61	W14 × 61	W14 × 68	W14 × 61
19	W14 × 30	W14 × 30	W14 × 34	W14 × 34	W14 × 30	W14 × 30	W14 × 30	W14 × 61	W14 × 34
20	W14 × 26	W14 × 22	W14 × 22	W14 × 22	W14 × 22	W14 × 22	W14 × 22	W14 × 22	W14 × 26
Weight (lb)	220,465	214,860	212,640	212,364	215,874	201,618	212,492	202,998	
Average optimized weight (lb)	229,555	222,620	N/A	215,226	225,071	209,644	N/A	212,289	(continued)

Table 17.10 (continued)

Optimal W-shaped sections						
Element group	ACO (Camp et al. [27])	HS (Degerstein [28])	ICA (Kaveh and Talatahari [23])	CSS (Kaveh and Talatahari [24])	CBO (Kaveh and Ilchi Ghazaan [25])	ECBO (Kaveh and Ilchi Ghazaan [25])
Standard deviation on average weight (lb)	4561	N/A	N/A	2448	N/A	N/A
Number of structural analyses	15,500	13,924	7500	5500	8280	15,360
						12,500
						16,220
					N/A	8292
						Present work [1]

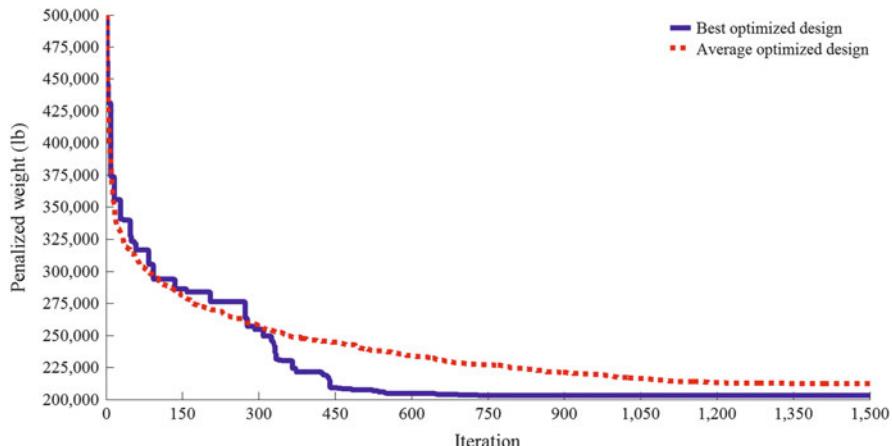


Fig. 17.17 Convergence curves obtained for the 3-bay 24-story frame problem

## 17.7 Concluding Remarks

This chapter presents a new population-based metaheuristic algorithm called vibrating particles system (VPS). This method is inspired by the damped free vibration of a single degree of freedom system. In the optimization process, particles gradually approach to their equilibrium positions. To maintain the balance between local search and global search, these equilibrium positions are obtained from the current population and the historically best position. Two truss and two frame benchmark structures are studied in order to show the performance of the VPS in terms of diversification, intensification, local optima avoidance, and convergence speed. The proposed algorithm finds superior optimal designs for three of the four problems investigated, illustrating the capability of the present method in solving constrained problems. Moreover, the average optimized results and standard deviation average results obtained by VPS are competitive with the other optimization methods. The convergence speed comparisons also reveal the fast-converging feature of the presented algorithm. For future research, it would be interesting to apply VPS to other optimization problems in different fields of science and engineering.

## References

1. Kaveh A, Ilchi Ghazaan M (2016) A new meta-heuristic algorithm: vibrating particles system. *Scientia Iranica*, in press
2. Saka MP (1990) Optimum design of pin-jointed steel structures with practical application. *J Struct Eng* 116:2599–2620
3. Erbatur F, Hasancebi O, Tütüncü I, Kılıç H (2000) Optimal design of planar and space structures with genetic algorithms. *Comput Struct* 75:209–224

4. Lee KS, Geem ZW (2004) A new structural optimization method based on the harmony search algorithm. *Comput Struct* 82:781–798
5. Hasançebi O, Çarbas S, Dogan E, Erdal F, Saka MP (2009) Performance evaluation of meta-heuristic search techniques in the optimum design of real size pin jointed structures. *Comput Struct* 87(5–6):284–302
6. Kazemzadeh Azad S, Hasançebi O (2015) Computationally efficient discrete sizing of steel frames via guided stochastic search heuristic. *Comput Struct* 156:12–28
7. Beer FP, Johnston ER Jr, Mazurek DF, Cornwell P, Self BP (2013) Vector mechanics for engineers. McGraw-Hill, New York, NY
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(5):267–283
9. American Institute of Steel Construction (AISC) (1989) Manual of steel construction: allowable stress design. American Institute of Steel Construction, Chicago, IL
10. Kaveh A, Talatahari S (2010) Optimal design of skeletal structures via the charged system search algorithm. *Struct Multidiscip Optim* 41:893–911
11. Kaveh A, Ilchi Ghazaan M, Bakhshpoori T (2013) An improved ray optimization algorithm for design of truss structures. *Period Polytech-Civ Eng* 57(2):97–112
12. Talatahari S, Kheirollahi M, Farahmandpour C, Gandomi AH (2013) A multi-stage particle swarm for optimum design of truss structures. *Neural Comput Appl* 23:1297–1309
13. Kaveh A, Mahdavi VR (2014) Colliding bodies optimization: a novel meta-heuristic method. *Comput Struct* 139:18–27
14. Kaveh A, Zolghadr A (2016) A novel meta-heuristic algorithm: tug of war optimization. *Int J Optim Civil Eng* 6(4):469–492
15. Kaveh A, Bakhshpoori T (2016) Water evaporation optimization: a novel physically inspired optimization algorithm. *Comput Struct* 167:69–85
16. Kaveh A, Zolghadr A (2012) Truss optimization with natural frequency constraints using a hybridized CSS–BBC algorithm with trap recognition capability. *Comput Struct* 102–103:14–27
17. Kaveh A, Ilchi Ghazaan M (2014) Enhanced colliding bodies algorithm for truss optimization with frequency constraints. *J Comput Civil Eng* 29(6). [10.1061/\(ASCE\) CP.1943-5487.0000445](https://doi.org/10.1061/(ASCE)CP.1943-5487.0000445), 04014104
18. Kaveh A, Mahdavi VR (2015) A hybrid CBO–PSO algorithm for optimal design of truss structures with dynamic constraints. *Appl Soft Comput* 34:260–273
19. American Institute of Steel Construction (AISC) (2001) Manual of steel construction: load and resistance factor design. American Institute of Steel Construction, Chicago, IL
20. Dumonteil P (1992) Simple equations for effective length factors. *Eng J AISC* 29(3):111–115
21. Kaveh A, Talatahari S (2009) Hybrid algorithm of harmony search, particle swarm and ant colony for structural design optimization. *Stud Comput Int* 239:159–198
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. Kaveh A, Talatahari S (2010) Optimum design of skeletal structure using imperialist competitive algorithm. *Comput Struct* 88:1220–1229
24. Kaveh A, Talatahari S (2012) Charged system search for optimal design of frame structures. *Appl Soft Comput* 12:382–393
25. Kaveh A, Ilchi Ghazaan M (2015) A comparative study of CBO and ECBO for optimal design of skeletal structures. *Comput Struct* 153:137–147
26. Talatahari S, Gandomi AH, Yang XS, Deb S (2015) Optimum design of frame structures using the eagle strategy with differential evolution. *Eng Struct* 91:16–25
27. Camp CV, Bichon BJ, Stovall S (2005) Design of steel frames using ant colony optimization. *J Struct Eng* 131:369–379
28. Degertekin SO (2008) Optimum design of steel frames using harmony search algorithm. *Struct Multidiscip Optim* 36:393–401

29. Kaveh A, Ilchi Ghazaan M (2014) Enhanced colliding bodies algorithm for truss optimization with frequency constraints. *J Comput Civil Eng* 29(6)
30. Kaveh A, Mahdavi VR (2015) A hybrid CBO–PSO algorithm for optimal design of truss structures with dynamic constraints. *Appl Soft Comput* 34:260–273. doi:[10.1061/\(ASCE\)CP.1943-5487.0000445,04014104](https://doi.org/10.1061/(ASCE)CP.1943-5487.0000445,04014104)

# Chapter 18

## Cyclical Parthenogenesis Optimization Algorithm

### 18.1 Introduction

Over the last few decades, metaheuristic algorithms have been successfully used for solving complex global optimization problems in science and engineering. These methods, which are usually inspired by natural phenomena, do not require any gradient information of the involved functions and are generally independent of the quality of the starting points. As a result, metaheuristic optimizers are favorable choices when dealing with discontinuous, multimodal, non-smooth, and non-convex functions, especially when near-global optimum solutions are sought, and the intended computational effort is limited.

Different characteristics of these algorithms cause them to perform dissimilarly on different classes of optimization problems, and therefore none of them can defeat all the others on all cases. As an everlasting source of inspiration, nature continues to provide researchers with different ideas in their search for new efficient optimization algorithms.

In the present chapter, a new metaheuristic algorithm is presented based on the reproduction and social behavior of some zoological species like aphids [1]. Aphids, which are considered as a highly successful group of organisms from a zoological standpoint [2], can switch between sexual and asexual reproduction mechanisms. In favorable circumstances, parthenogenesis and telescoping of generations enable aphids to achieve very high rates of reproduction [3]. This enables the aphids to exploit advantageous environmental conditions like abundance of food. However, probably to maintain genetic diversity and in response to adverse changes in environmental situation, female aphids reproduce sexual females and males when they find it necessary. Although the reasons behind reproduction behavior of aphids are not completely agreed upon in zoology, advantages of both of these alternating systems are evident from an optimization point of view.

The remainder of this chapter is organized as follows. In Sect. 18.2, the new optimization algorithm is presented subsequent to a concise and simplified

introduction to real aphids. Sensitivity of CPA to its parameters is studied in Sect. 18.3. Some mathematical and engineering design benchmark problems are then studied in Sect. 18.4 in order to examine the efficiency of the proposed algorithm. The concluding remarks are finally presented in Sect. 18.5.

## 18.2 Cyclical Parthenogenesis Algorithm

In this section cyclical parthenogenesis algorithm (CPA) is introduced and described as a population-based metaheuristic algorithm for global optimization. The main rules of CPA are explained using some key aspects of the lives of aphids as a highly successful organism. Not all the details of the complicated life cycles of about 4000 species of aphids are known to us; however, there are some common features of these intricate life cycles such as the ability to reproduce both sexually and asexually (cyclical parthenogenesis), which seem to be interesting from an optimization point of view.

### 18.2.1 *Aphids and Cyclical Parthenogenesis*

Aphids are small sap-sucking insects, and members of the superfamily Aphidoidea [4]. As one of the most destructive insect pests on cultivated plants in temperate regions, aphids have fascinated and frustrated man for a very long time. This is mainly because of their intricate life cycles and close association with their host plants and their ability to reproduce both asexually and sexually [3]. Figure 18.1 shows a female aphid surrounded by her offspring on a host plant.

Aphids are capable of reproducing offspring both sexually and asexually. In asexual reproduction the offspring arise from the female parent and inherit the genes of that parent only. In asexual reproduction most of the offspring are genetically identical to their mother and genetic changes occur relatively rarely [3]. This form of reproduction is chosen by female aphids in suitable and stable environments and allows them to rapidly grow a population of similar aphids, which can exploit the favorable circumstances. Sexual reproduction, on the other hand, offers a net advantage by allowing more rapid generation of genetic diversity, making adaptation to changing environments available [5].

Since the habitat occupied by an aphid species is not uniform but consists of a spatial-temporal mosaic of many different patches, each with its own complement of organisms and resources [3], aphids employ sexual reproduction in order to maintain the genetic diversity required for increasing the chance of including the fittest genotype for a particular patch. This is the basis of the lottery model proposed by Williams [6] for explaining the role of sexual reproduction in evolution.

Some aphid species produce winged offspring in response to poor conditions on the host plant or when the population on the plant becomes too large. These winged

**Fig. 18.1** A female aphid surrounded by her offspring on a host plant



offspring, which are called alates, can disperse to other food sources [4]. Flying aphids have little control over the direction of their flight because of their low speed. However, once within the layer of relatively still air around vegetation, aphids can control their landing on plants and respond to either olfactory or visual cues, or both.

### 18.2.2 *Description of Cyclical Parthenogenesis Algorithm*

Cyclical Parthenogenesis Algorithm (CPA) is a population-based metaheuristic optimization algorithm inspired from social and reproduction behavior of aphids. It starts with a population of randomly generated candidate solutions metaphorized as aphids. The quality of the candidate solutions is then improved using some simplified rules inspired from the life cycle of aphids.

Naturally, CPA does not attempt to represent an exact model of the life cycle of aphids, which is neither possible nor necessary. Instead, it encompasses certain features of their behavior to construct a global optimization algorithm.

Like many other population-based metaheuristic algorithms, CPA starts with a population of  $N_a$  candidate solutions randomly generated in the search space. These candidate solutions, which are considered as aphids, are grouped into  $N_c$  colonies, each inhabiting a host plant. These aphids reproduce offspring through sexual and asexual reproduction mechanisms. Like real aphids, in general, larger (fitter) individuals within a colony have a greater reproductive potential than smaller

ones. Some of the aphids prefer to leave their current host plant and search for better conditions. In CPA it is assumed that these flying aphids cannot fly much further due to their weak wings and end up on a plant occupied by another colony nearby. Like real aphids, the agents of the algorithm can reproduce for multiple generations. However, the life span of aphids is naturally limited, and less fit ones are more likely to be dead in adverse circumstance. The main steps of CPA can be stated as follows:

### **Step 1: Initialization**

A population of  $N_a$  initial solutions is generated randomly:

$$x_{ij}^0 = x_{j,\min} + \text{rand}(x_{j,\max} - x_{j,\min}) \quad j = 1, 2, \dots, n \quad (18.1)$$

where  $x_{ij}^0$  is the initial value of the  $j$ th variable of the  $i$ th candidate solution;  $x_{j,\max}$  and  $x_{j,\min}$  are the maximum and minimum permissible values for the  $j$ th variable, respectively;  $\text{rand}$  is a random number from a uniform distribution in the interval  $[0, 1]$ ; and  $n$  is the number of optimization variables. The candidate solutions are then grouped into  $N_c$  colonies, each inhabiting a host plant. The number of aphids in all colonies  $N_m$  is equal.

### **Step 2: Evaluation, Reproduction, and Flying**

The objective function values for the candidate solutions are evaluated. The aphids on each plant are sorted in the ascending order of their objective function values and saved in a female memory (FM). Each of the members of the female memory is capable of asexually reproducing a genetically identical clone in the next iteration.

In each iteration,  $N_m$  new candidate solutions are generated in each of the colonies in addition to identical clones. These new solutions can be reproduced either sexually or asexually. A ratio  $F_r$  of the best of these new solutions is considered as female aphids; the rest are considered as male aphids.

#### **Asexually Generated New Solutions**

A female parent is selected randomly from the population of all female parents of the colony (identical clones and newly produced females). Then, this female parent reproduces a new offspring asexually by the following expression:

$$x_{ij}^{k+1} = F_j^k + \alpha_1 \times \frac{\text{randn}}{k} \times (x_{j,\max} - x_{j,\min}) \quad j = 1, 2, \dots, n \quad (18.2)$$

where  $x_{ij}^{k+1}$  is the value of the  $j$ th variable of the  $i$ th candidate solution in the  $(k+1)$ th iteration;  $F_j^k$  is the value of the corresponding variable of the female parent in the  $k$ th iteration;  $\text{randn}$  is a random number drawn from a normal distribution; and  $\alpha_1$  is a scaling parameter.

#### **Sexually Generated New Solutions**

Each of the male aphids selects a female randomly in order to produce an offspring sexually:

$$x_{ij}^{k+1} = M_j^k + \alpha_2 \times \text{rand} \times (F_j^k - M_j^k) \quad j = 1, 2, \dots, n \quad (18.3)$$

where  $M_j^k$  is the value of the  $j$ th variable of the male solution in the  $k$ th iteration and  $\alpha_2$  is a scaling factor. It can be seen that in a sexual reproduction, two different solutions share information, while in an asexual reproduction, the new solution is generated using merely the information of one single parent solution.

### Death and Flight

When all of the new solutions of all colonies are generated, flying occurs with a probability of  $P_f$  where two of the colonies are selected randomly and a winged aphid asexually reproduced by and identical to the best female of Colony1 flies to Colony2. In order to keep the number of members of each colony constant, it is assumed that the worst member of Colony2 dies.

### Step 3: Updating the Colonies

The objective function values of the newly generated candidate solutions are evaluated and the female memories are updated.

### Step 4: Termination

Steps 2 and 3 are repeated until a termination criterion is satisfied. The pseudo code of CPA is presented in Table 18.1.

## 18.3 Sensitivity Analysis of CPA

Performance of a metaheuristic algorithm is highly dependent on the values of its internal parameters. In this section, the sensitivity of CPA to its parameters is investigated considering the weight minimization of a 10-bar truss, as depicted in Fig. 18.2. The details of this problem are further explained in the next section. A parametric study is performed considering  $N_a$ ,  $N_c$ ,  $F_r$ ,  $P_f$ ,  $\alpha_1$ , and  $\alpha_2$ . In each case, the problem is solved ten times in order to obtain statistically significant results. The total number of structural analyses is set as 24,000. The results of the sensitivity analysis of CPA for this example are provided in Table 18.2. When studying each parameter, the other parameters are kept unchanged. The initial values of parameters to be used in the sensitivity analysis process are  $N_a = 60$ ,  $N_c = 4$ ,  $F_r = 0.4$ ,  $\alpha_1 = 1$ , and  $\alpha_2 = 2$ . A linear function increasing from 0 to 1 is considered for  $P_f$ .

A diversity index as introduced by Kaveh and Zolghadr [7] is also used to study the exploration/exploitation behavior of the algorithm for different parameter values. The index reflects the relative positions of the agents of an algorithm and can be used to observe the diversity of the agents of an algorithm during an optimization process:

**Table 18.1** Pseudo code of the CPA algorithm [1]

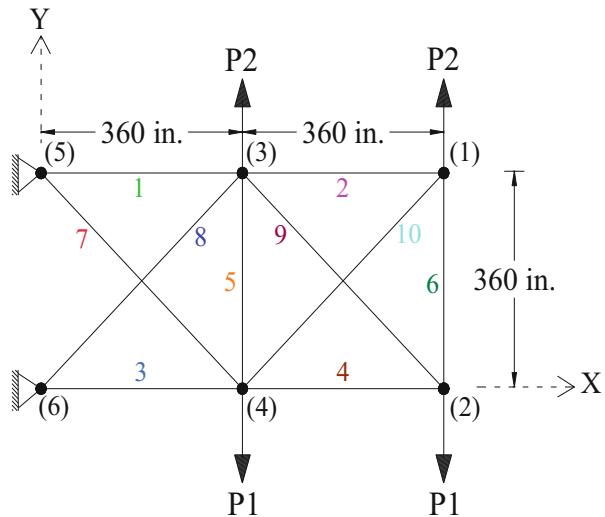
```

procedure Cyclical Parthenogenesis Algorithm
begin
    Initialize parameters;
    Initialize a population of  $N_a$  random candidate solutions;
    Group the candidate solutions in  $N_c$  colonies with each having  $N_m$  members;
    Evaluate and Sort the candidate solutions of each colony and save the best  $N_m$  ones in
    Female Memory

    while (termination condition not met) do
        for m: 1 to  $N_c$ 
            Reproduce an identical solution by each of the solutions of the Female
            Memory
            Divide the newly generated offspring into male and female considering  $F_r$ 
            for i: 1 to  $F_r \times N_m$ 
                Generate new solution  $i$  asexually using Eq. (18.2)
            end for
            for i:  $F_r \times N_m + 1$  to  $N_m$ 
                Generate new solution  $i$  sexually using Eq. (18.3)
            end for
            if rand <  $P_f$ 
                Select two colonies randomly
                Generate an winged identical offspring from the best solution of Colony1
                Eliminate the worst solution of Colony2 and move winged aphid to Colony2
            end if
            Evaluate the objective function values of new aphids
            Update the Female Memory
        end for
    end while
end

```

**Fig. 18.2** Schematic of the planar 10-bar truss structure



$$\text{Diversity Index} = \frac{1}{nP} \sum_{j=1}^{nP} \sqrt{\sum_{i=1}^{nVAR} \left( \frac{GB(i) - X_j(i)}{X_{i,\max} - X_{i,\min}} \right)^2} \quad (18.4)$$

where  $X_j(i)$  is the value of the  $i$ th variable of the  $j$ th particle;  $X_{i,\min}$  and  $X_{i,\max}$  are the minimum and maximum values of the  $i$ th variable, respectively;  $nVAR$  is the number of design variables; and  $nP$  is the number of particles. In order to decrease random fluctuations, the mean values of diversity index in the ten independent runs are plotted against iteration numbers for each parameter set.

In order to study the effect of population size ( $N_a$ ), the algorithm is run with 20, 40, 60, 80, and 100 aphids, while keeping the total number of structural analyses unchanged. From Table 18.2 it can be seen that the best performance of CPA in terms of the best weight, mean weight, and standard deviation is obtained for  $N_a = 60$ . The second best results are obtained when considering  $N_a = 80$ . The best and mean performance of the algorithm for different values of  $N_a$  is depicted in Fig. 18.3.

The concept of multiple colonies allows CPA to search different portions of the search spaces more or less independently and prevents the unwanted premature convergence phenomenon. Table 18.2 shows the statistical information for different values of  $N_c$ , i.e., the number of colonies, where the population size is kept unchanged ( $N_a = 60$ ). As it can be seen, the best performance in terms of best weight, mean weight, and standard deviation is obtained for  $N_c = 4$ . This parameter value provides a good balance between the diversification and intensification tendencies of the algorithm. In fact, dividing the population of aphids into more colonies limits the circulation of information among the aphids and allows the aphids of a colony to explore the search space more independently without being affected by other colonies. This generally results in a more diverse search. On the

**Table 18.2** Results of the sensitivity analysis of CPA for the 10-bar truss

Parameter/value	Best weight (lb)	Mean weight (lb)	Standard deviation (lb)
<b>Population size</b>			
Na = 20	5062.89	5074.42	8.61
Na = 40	5061.11	5063.68	5.18
Na = 60	5060.94	5061.35	0.23
Na = 80	5061.08	5061.50	0.51
Na = 100	5061.17	5061.98	0.53
<b>Number of colonies</b>			
Nc = 1	5061.01	5064.76	6.42
Nc = 2	5061.15	5062.01	2.04
Nc = 3	5061.04	5062.97	4.98
Nc = 4	5060.94	5061.35	0.23
Nc = 6	5061.24	5062.10	0.65
Nc = 12	5061.89	5067.62	7.05
<b>Female ratio</b>			
Fr = 0.2	5060.98	5061.49	0.53
Fr = 0.4	5060.94	5061.35	0.23
Fr = 0.6	5061.06	5061.76	0.59
Fr = 0.8	5061.64	5067.59	7.54
<b>Flight probability</b>			
Pf = 1	5061.04	5063.34	4.98
Pf = 0	5061.06	5062.89	2.90
Pf linear	5060.94	5061.35	0.23
<b>Step size</b>			
$\alpha_1 = 0.5$ and $\alpha_2 = 0.5$	5314.20	5785.27	35.50
$\alpha_1 = 0.5$ and $\alpha_2 = 1$	5169.72	5684.86	223.67
$\alpha_1 = 0.5$ and $\alpha_2 = 2$	5061.03	5061.90	0.59
$\alpha_1 = 1$ and $\alpha_2 = 0.5$	5100.97	5301.86	149.41
$\alpha_1 = 1$ and $\alpha_2 = 1$	5084.95	5410.06	235.75
$\alpha_1 = 1$ and $\alpha_2 = 2$	5060.94	5061.35	0.23
$\alpha_1 = 2$ and $\alpha_2 = 0.5$	5085.01	5165.37	111.6
$\alpha_1 = 2$ and $\alpha_2 = 1$	5062.51	5088.97	35.51
$\alpha_1 = 2$ and $\alpha_2 = 2$	5061.04	5062.97	4.85
$\alpha_1 = 2$ and $\alpha_2 = 4$	5062.06	5073.81	6.37
$\alpha_1 = 4$ and $\alpha_2 = 2$	5061.61	5065.18	6.63
$\alpha_1 = 4$ and $\alpha_2 = 4$	5062.23	5069.80	7.84

other hand, having more aphids in a colony (less number of colonies) permits the algorithm to search a particular region of the search space more thoroughly (more intensification), but, at the same time, limits the diversification of the algorithm. The best and mean performance of the algorithm for different values of  $N_c$  is depicted in Fig. 18.4. The diversity index curves of CPA for different values of  $N_c$  are shown in Fig. 18.5. It can be seen that the curves corresponding to  $N_c = 3$  and  $N_c = 4$ , which result in the best performance of the algorithm, are placed between

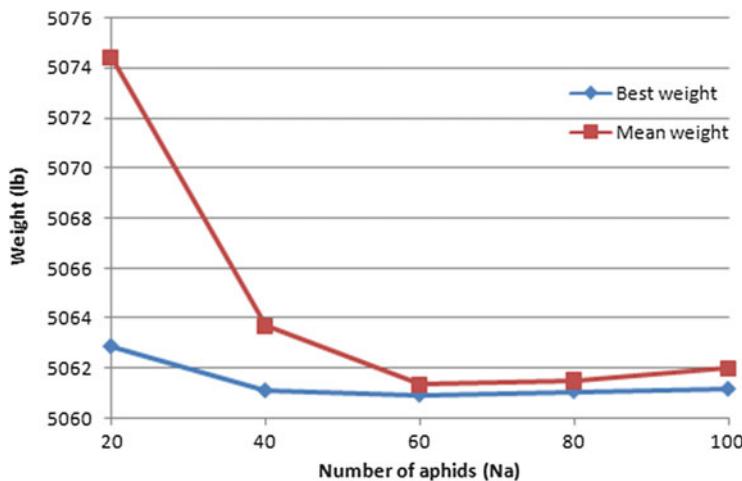


Fig. 18.3 Best and mean performance of CPA for different values of  $N_a$

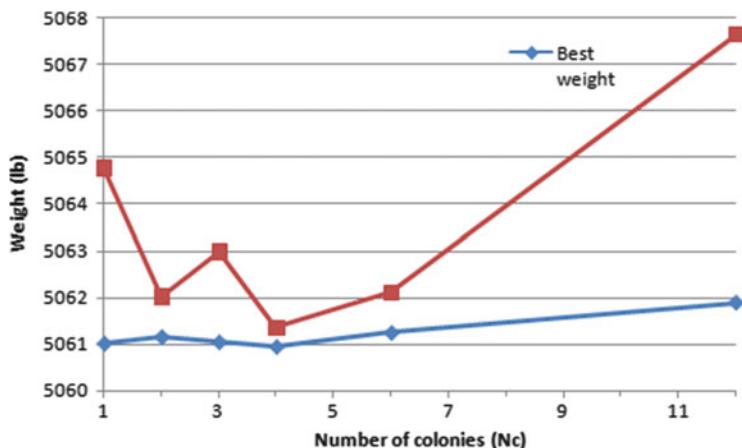


Fig. 18.4 Best and mean performance of CPA for different values of  $N_c$

those of  $N_c = 1$  and  $N_c = 2$  from below and  $N_c = 6$  and  $N_c = 12$  from above. This conforms to the above discussion, i.e., dividing the aphids into more colonies generally results in slower rate of convergence (more diversity index values).  $N_c = 4$  seems to provide a good balance between the exploration and exploitation tendencies of the algorithm.

Parameter  $F_r$  defines the ratio of female aphids to all of the newly generated aphids. Increasing the value of this parameter results in an increase in the number of asexually generated aphids of the next generation. Since such aphids use a single source of information (female parent), they can be interpreted as means of searching a localized region around their parent. This localized region gets smaller gradually as the optimization process proceeds. Sexual reproduction, on the other

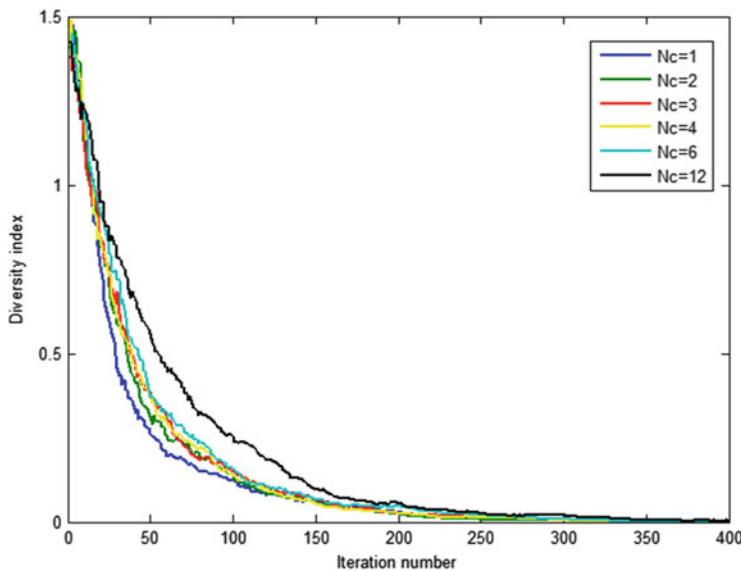


Fig. 18.5 Diversity index curves of CPA for different values of  $N_c$

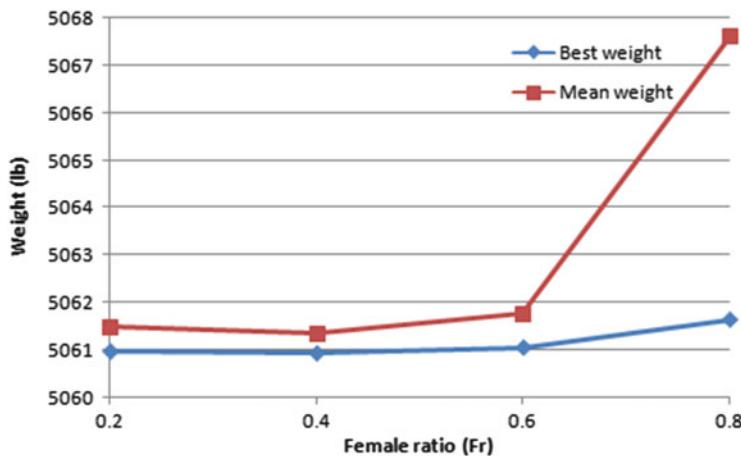
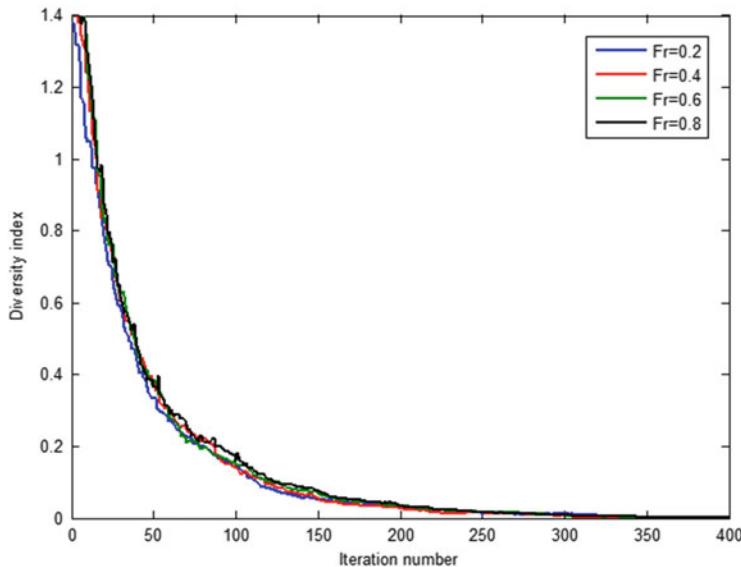


Fig. 18.6 Best and mean performance of CPA for different values of  $F_r$

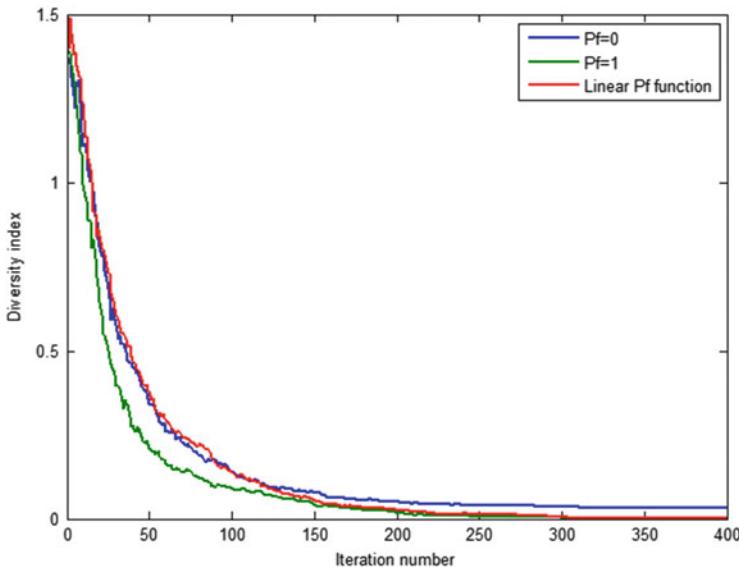
hand, incorporates two different sources of information and therefore contributes to the convergence of the algorithm by letting the aphids share information. According to Table 18.2 which summarizes the statistical information for different values of  $F_r$ , it can be seen that the proposed algorithm exhibits its best performance in terms of mean weight when the value of this parameter is taken as 0.4, while the second and third best performances are obtained for  $F_r = 0.2$  and  $F_r = 0.6$ . The best and mean performance of the algorithm for different values of  $N_a$  is depicted in Fig. 18.6. The diversity index curves of CPA for different values of  $F_r$  are shown



**Fig. 18.7** Diversity index curves of CPA for different values of  $F_r$

in Fig. 18.7. It can be seen that increasing the value of  $F_r$  slightly increases the diversity of the aphids in the search space, i.e., the curves corresponding to  $F_r = 0.8$  and  $F_r = 0.2$  are placed above and below the other curves, respectively. However, the effect is not very significant and the curves are very close to each other.

Parameter  $P_f$  is responsible for defining the level of information exchange among the colonies. With no possible flights, the colonies would be performing their search in a completely independent manner, i.e., an optimization run with  $N_a$  aphids divided into four colonies would be similar to four independent runs each with  $N_a/4$  aphids per colony. It is obvious that this would not be particularly favorable, since it is in fact changing the population of aphids without actually utilizing the abovementioned benefits of multiple colonies. On the other hand, permitting too many flights results corresponds to merging the information sources of different colonies. It is important to note that at the early stages of the optimization process, it is more favorable to give the colonies a higher level of independence so that they can search the problem space without being affected by the other colonies. However, as the optimization process proceeds, it is desirable to let the colonies share more information so as to provide the opportunity for the more promising regions of the search space to be searched more thoroughly. Three different cases are considered for  $P_f$  in this study:  $P_f = 0$ ,  $P_f = 1$ , and  $P_f$  linearly increasing from 0 to 1. It can be clearly seen from Table 18.2 that the best performance of the algorithm corresponds to the linear case, since it conforms to the abovementioned discussion on information circulation. The diversity index curves of CPA for different values of  $P_f$  are shown in Fig. 18.8. It can be seen



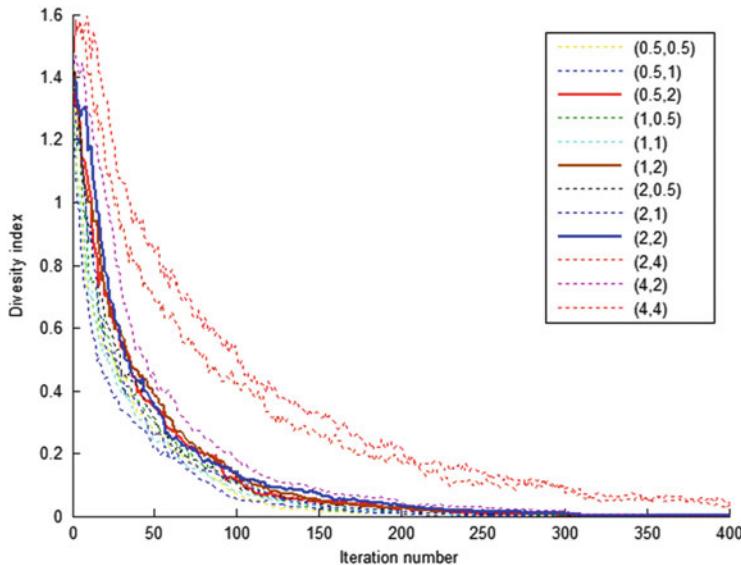
**Fig. 18.8** Diversity index curves of CPA for different values of  $P_f$

that when  $P_f=0$ , the diversity index values are relatively high even at the end of the optimization process. This means that the different colonies have converged to different results. The curve corresponding to the linear function is similar to that of  $P_f=0$  at the early stages of the optimization process, while it gets closer to the curve corresponding to  $P_f=1$  at the final stages. This explains the favorable behavior of the algorithm when a linear function is chosen for  $P_f$ .

Parameters  $\alpha_1$  and  $\alpha_2$  represent the step size of the agents in asexual and sexual reproductions, respectively. The sensitivity of CPA to these two parameters is also shown in Table 18.2. The diversity index curves of CPA for different values of  $\alpha_1$  and  $\alpha_2$  are shown in Fig. 18.9. According to Table 18.2 the best performance of the algorithm corresponds to  $\alpha_1 = 1$  and  $\alpha_2 = 2$ . There are two other cases which result in relatively good performance of the algorithm, i.e.,  $\alpha_1 = 0.5$  and  $\alpha_2 = 2$  and  $\alpha_1 = 2$  and  $\alpha_2 = 2$ . As it can be seen in Fig. 18.9, the diversity index curves for these three cases are almost the same.

## 18.4 Test Problems and Optimization Results

In order to evaluate the efficiency of the proposed algorithm, some benchmark test problems are considered from the literature. A set of unimodal and multimodal mathematical optimization problems are studied in Sect. 18.4.1. In addition, truss weight minimizations of a planar 10-bar truss, a spatial 25-bar transmission tower, a spatial 72-bar truss, a 120-bar dome-shaped truss, and a planar 200-bar truss are



**Fig. 18.9** Diversity index curves of CPA for different values of  $\text{Alpha1}$ ,  $\text{Alpha2}$

considered as structural optimization problems. The results of utilizing CPA on these structural optimization problems are then compared to some of the state-of-the-art metaheuristic algorithms.

#### 18.4.1 Mathematical Optimization Problems

In this section, the efficiency of the CPA is evaluated by solving the mathematical benchmark problems summarized in Table 18.3. These benchmark problems are taken from Ref. [8], where some variants of GA were used as the optimization algorithm. The results obtained by CPA are presented in Table 18.4 along with those of some GA variants. Each objective function is optimized 50 times independently starting from different initial populations, and the average number of function evaluations required by each algorithm is presented. The numbers in the parentheses indicate the ratio of the successful runs in which the algorithm has obtained the global minimum with predefined accuracy, which is taken as  $\epsilon = f_{\min} - f_{final} = 10^{-4}$ . The absence of the parentheses indicates that the algorithm has been successful in all independent runs.

As it can be seen from Table 18.4, CPA generally performs better than GA and its variants in the mathematical optimization problems considered in this study.

**Table 18.3** Details of the benchmark mathematical problems solved in this study

Function name	Side constraints	Function	Global minimum
<i>Aluffi-Pentini</i>	$\mathbf{X} \in [-10, 10]^2$	$f(\mathbf{X}) = \frac{1}{4}x_1^4 - \frac{1}{2}x_1^2 + \frac{1}{10}x_1 + \frac{1}{2}x_2^2$	-0.352386
<i>Bohachevsky 1</i>	$\mathbf{X} \in [-100, 100]^2$	$f(\mathbf{X}) = x_1^2 + 2x_2^2 - \frac{3}{10} \cos(3\pi x_1) - \frac{4}{10} \cos(4\pi x_2) + \frac{7}{10}$	0.0
<i>Bohachevsky 2</i>	$\mathbf{X} \in [-50, 50]^2$	$f(\mathbf{X}) = x_1^2 + 2x_2^2 - \frac{3}{10} \cos(3\pi x_1) \cos(4\pi x_2) + \frac{3}{10}$	0.0
<i>Becker and Lago</i>	$\mathbf{X} \in [-10, 10]^2$	$f(\mathbf{X}) = ( x_1  - 5)^2 + ( x_2  - 5)^2$	0.0
<i>Branin</i>	$0 \leq x_2 \leq 15 - 5 \leq x_1 \leq 10$	$f(\mathbf{X}) = (x_2 - \frac{5}{4\pi}x_1^2 + \frac{5}{\pi}x_1)^2 + 10(1 - \frac{1}{8\pi}) \cos(x_1) + 10$	0.397887
<i>Camel</i>	$\mathbf{X} \in [-5, 5]^2$	$f(\mathbf{X}) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	-1.0316
<i>Cb3</i>	$\mathbf{X} \in [-5, 5]^2$	$f(\mathbf{X}) = 2x_1^2 - 1.05x_1^4 + \frac{1}{6}x_1^6 + x_1x_2 + x_2^2$	0.0
<i>Cosine mixture</i>	$n = 4, \mathbf{X} \in [-1, 1]^n$	$f(\mathbf{X}) = \sum_{i=1}^n x_i^2 - \frac{1}{10} \sum_{i=1}^n \cos(5\pi x_i)$	-0.4
<i>DeJong</i>	$\mathbf{X} \in [-5, 12, 5, 12]^3$	$f(\mathbf{X}) = x_1^2 + x_2^2 + x_3^2$	0.0
<i>Exponential</i>	$n = 2, 4, \mathbf{X} \in [-1, 1]^n$	$f(\mathbf{X}) = -\exp\left(-0.5 \sum_{i=1}^n x_i^2\right)$	-1
<i>Goldstein and Price</i>	$\mathbf{X} \in [-2, 2]^2$	$f(\mathbf{X}) = \begin{aligned} & [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2] \\ & \times [30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]^2 \end{aligned}$	3.0
<i>Griewank</i>	$\mathbf{X} \in [-100, 100]^2$	$f(\mathbf{X}) = 1 + \frac{1}{200} \sum_{i=1}^2 x_i^2 - \prod_{i=1}^2 \cos\left(\frac{x_i}{\sqrt{i}}\right)$	0.0
<i>Hartman 3</i>	$\mathbf{X} \in [0, 1]^3$	$f(\mathbf{X}) = -\sum_{i=1}^4 c_i \exp\left(-\sum_{j=1}^3 a_{ij}(x_j - p_{ij})^2\right)$	-3.862782
		$c = \begin{bmatrix} 3 & 10 & 30 \\ 0.1 & 10 & 35 \\ 3 & 10 & 30 \\ 0.1 & 10 & 35 \end{bmatrix}, p = \begin{bmatrix} 1 \\ 1.2 \\ 3 \\ 3.2 \end{bmatrix}, and$	$\begin{bmatrix} 0.3689 & 0.1117 & 0.2673 \\ 0.4699 & 0.4387 & 0.747 \\ 0.1091 & 0.8732 & 0.5547 \\ 0.03815 & 0.5743 & 0.8828 \end{bmatrix}$

<p><i>Hartman 6</i></p> <p><math>\mathbf{X} \in [0, 1]^6</math></p>	$f(\mathbf{X}) = -\sum_{i=1}^4 c_i \exp \left( -\sum_{j=1}^6 a_{ij} (x_j - p_{ij})^2 \right)$ $a = \begin{bmatrix} 10 & 3 & 17 & 3.5 & 1.7 & 8 \\ 0.05 & 10 & 17 & 0.1 & 8 & 14 \\ 3 & 3.5 & 17 & 10 & 17 & 8 \end{bmatrix}, c = \begin{bmatrix} 1 \\ 1.2 \\ 3 \\ 3.2 \end{bmatrix}, \text{ and}$ $p = \begin{bmatrix} 0.1312 & 0.1696 & 0.5569 & 0.0124 & 0.8283 & 0.5886 \\ 0.2329 & 0.4135 & 0.8307 & 0.3736 & 0.1004 & 0.9991 \\ 0.2348 & 0.1451 & 0.3522 & 0.2883 & 0.3047 & 0.6650 \\ 0.4047 & 0.8828 & 0.8732 & 0.5743 & 0.1091 & 0.0381 \end{bmatrix}$	$-3.322368$
---	--	-------------

**Table 18.4** Performance comparison of CPA and some GA variants in the mathematical optimization problems

Function name	GEN	GEN-S	GEN-S-M	GEN-S-M-LS	CPA
AP	1360 (0.99)	1360	1277	1253	560
Bf1	3992	3356	1640	1615	1173
Bf2	20,234	3373	1676	1636	1376
BL	19,596	2412	2439	1436	424
Branin	1442	1418	1404	1257	708
Camel	1358	1358	1336	1300	482
Cb3	9771	2045	1163	1118	548
CM	2105	2105	1743	1539	1612
DeJoung	9900	3040	1462	1281	670
Exp2	938	936	817	807	435
Exp4	3237	3237	2054	1496	781
Exp8	3237	3237	2054	1496	1105
Goldstein and Price	1478	1478	1408	1325	805
Griewank	18,838 (0.91)	3111 (0.91)	1764	1652 (0.99)	1572
Hartman 3	1350	1350	1332	1274	1128
Hartman 6	2562 (0.54)	2562 (0.54)	2530 (0.67)	1865 (0.68)	1533

### 18.4.2 Truss Design Problems

In order to further investigate the efficiency of the CPA, five continuous truss design problems are considered in this section. The results are compared to those obtained by some of the state-of-the-art metaheuristic optimization algorithms. 60 aphids are considered for all of the benchmark truss optimization problems. The total number of iterations is considered as 400 for all of the examples except for the last one, where 600 iterations are permitted. These constrained optimization problems are turned into unconstrained ones using a penalty approach. If the constraints are satisfied, then the amount of penalty will be zero; otherwise its value can be calculated as the ratio of violated constraint to the corresponding allowable limit. The CPA is coded in the MATLAB software environment. The required structural analyses are carried out using the direct stiffness method also coded in MATLAB.

#### 18.4.2.1 Design of a Planar 10-Bar Truss Structure

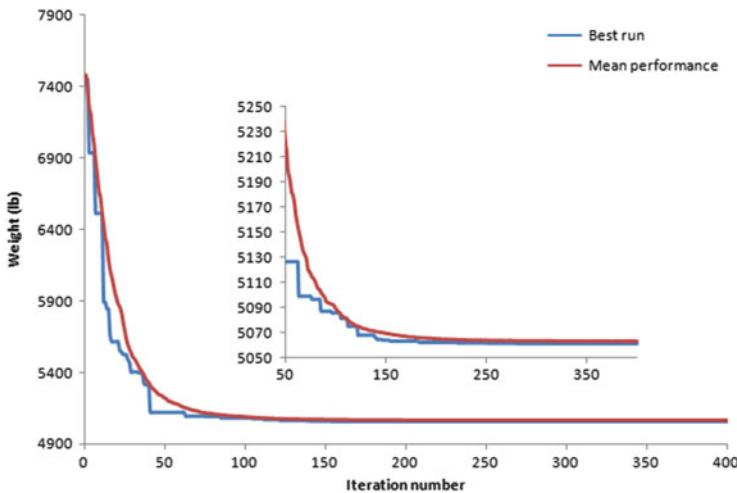
A 10-bar truss as shown in Fig. 18.2 is considered as the first structural design problem. This is a well-known problem in the field of structural optimization and has been solved by many researchers using different optimization algorithms. The material density is 0.1 lb/in<sup>3</sup> and the modulus of elasticity is 10,000 ksi. The members are subjected to stress limitation of 25 ksi, while the horizontal and vertical displacements of all nodes are limited to  $\pm 2.0$  in. Each of the members is

**Table 18.5** Optimized results obtained by CPA and some other metaheuristic algorithms for the 10-bar truss problem

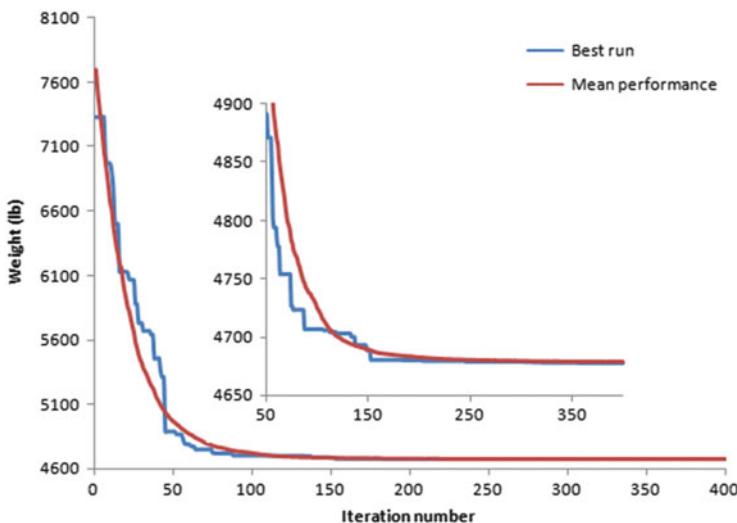
Element group	Optimal cross-sectional areas (in <sup>2</sup> )					
	ABC-AP [9]	SAHS [10]	TLBO [11]	MSPSO [12]	WEO [13]	Present work
Case 1						
1	30.5480	30.3940	30.4286	30.5257	30.5755	30.5022
2	0.1000	0.1000	0.1000	0.1001	0.1000	0.1000
3	23.1800	23.0980	23.2436	23.2250	23.3368	23.2170
4	15.2180	15.4910	15.3677	15.4114	15.1497	15.2204
5	0.1000	0.1000	0.1000	0.1001	0.1000	0.1001
6	0.5510	0.5290	0.5751	0.5583	0.5276	0.5587
7	7.4630	7.4880	7.4404	7.4395	7.4458	7.4548
8	21.0580	21.1890	20.9665	20.9172	20.9892	21.0371
9	21.5010	21.3420	21.5330	21.5098	21.5236	21.5295
10	0.1000	0.1000	0.1000	0.1000	0.1000	0.1002
Best weight (lb)	5060.880	5061.42	5060.96	5061	5060.99	5060.92
Mean weight (lb)	N/A	5061.95	5062.08	5064.46	5062.09	5062.45
Standard dev. (lb)	N/A	0.71	0.79	5.72	2.05	3.77
No. structural analyses	$500 \times 10^3$	7081	16,872	N/A	19,540	23700
Case 2						
1	23.4692	23.5250	23.5240	23.4432	23.5804	23.5515
2	0.1005	0.1000	0.1000	0.1000	0.1003	0.1000
3	25.2393	25.4290	25.4410	25.3718	25.1582	25.5440
4	14.3540	14.4880	14.4790	14.1360	14.1801	14.1674
5	0.1001	0.1000	0.1000	0.1000	0.1002	0.1000
6	1.9701	1.9920	1.9950	1.9699	1.9708	1.9698
7	12.4128	12.3520	12.3340	12.4335	12.4511	12.3533
8	12.8925	12.6980	12.6890	13.0173	12.9349	12.8167
9	20.3343	20.3410	20.3540	20.2717	20.3595	20.3302
10	0.1000	0.1000	0.1000	0.1000	0.1001	0.1001
Best weight (lb)	4677.077	4678.84	4678.31	4677.26	4677.31	4677.16
Mean weight (lb)	N/A	4680.08	4680.12	4681.45	4679.06	4678.62
Standard dev. (lb)	N/A	1.89	1.016	2.19	2.07	0.95
No. structural analyses	$500 \times 10^3$	7267	14,857	N/A	19,890	23640

considered as an independent design variable with lower and upper bounds of 0.1 and 35.0 in<sup>2</sup>, respectively. There are two independent loading cases acting on the structure: Case 1,  $P_1 = 100$  kips and  $P_2 = 0$ , and Case 2,  $P_1 = 150$  kips and  $P_2 = 50$  kips.

Table 18.5 compares the optimized designs found by CPA together with some of the state-of-the-art optimization algorithms for the loading cases. It can be seen that the results found by CPA are comparable to those of the other state-of-the-art

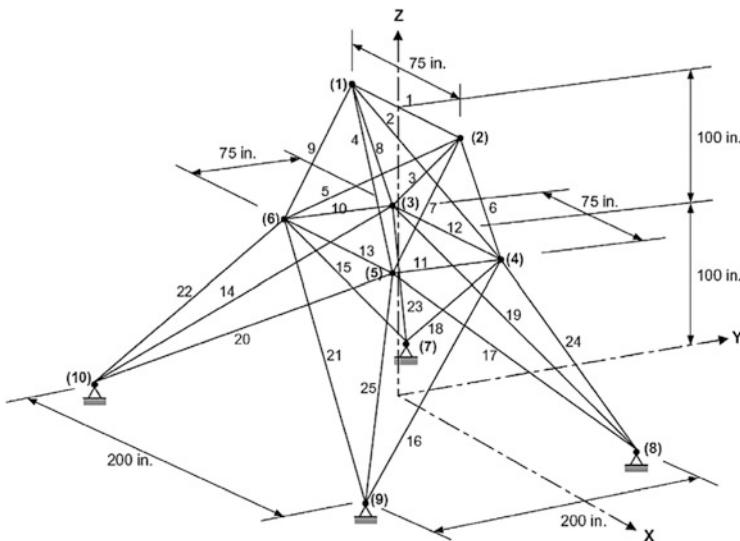


**Fig. 18.10** Convergence curves of the best result of the CPA together with the mean performance of the algorithm for the 10-bar planar truss (Case 1)



**Fig. 18.11** Convergence curves of the best result of the CPA together with the mean performance of the algorithm for the 10-bar planar truss (Case 2)

algorithms. In both cases CPA has obtained the best result after that of ABC-AP. It should be noted that CPA requires less than 5 % of the structural analyses used by ABC-AP (less than 24,000 compared to 500,000). Figures 18.10 and 18.11 show the convergence curves of the best run and the mean performance of CPA on the 10-bar planar truss.



**Fig. 18.12** Schematic of the spatial 25-bar transmission tower

**Table 18.6** Independent loading conditions acting on the spatial 25-bar truss

Node	Case 1			Case 2		
	P <sub>x</sub> kips	P <sub>y</sub> kips	P <sub>z</sub> kips	P <sub>x</sub> kips	P <sub>y</sub> kips	P <sub>z</sub> kips
1	0.0	20.0	-5.0	1.0	10.0	-5.0
2	0.0	-20.0	-5.0	0.0	10.0	-5.0
3	0.0	0.0	0.0	0.5	0.0	0.0
6	0.0	0.0	0.0	0.5	0.0	0.0

#### 18.4.2.2 Design of a 25-Bar Transmission Tower Truss

Weight minimization of a 25-bar transmission tower as schematically depicted in Fig. 18.12 is considered as the second structural optimization problem. The material density and modulus of elasticity are 0.1 lb/in<sup>3</sup> and 10,000 ksi, respectively.

Table 18.6 shows the two independent loading conditions applied to the structure. The 25 bars of the truss are classified into eight groups as follows:

(1) A<sub>1</sub>, (2) A<sub>2</sub>–A<sub>5</sub>, (3) A<sub>6</sub>–A<sub>9</sub>, (4) A<sub>10</sub>–A<sub>11</sub>, (5) A<sub>12</sub>–A<sub>13</sub>, (6) A<sub>14</sub>–A<sub>17</sub>, (7) A<sub>18</sub>–A<sub>21</sub>, and (8) A<sub>22</sub>–A<sub>25</sub>.

Maximum displacement limitations of 0.350 in are imposed on all nodes in all directions. The axial stress constraints, which are different for each group, are shown in Table 18.7. The cross-sectional areas vary continuously from 0.01 to 3.4 in<sup>2</sup> for all members.

This is a very well-known test problem in the field of structural optimization and is investigated by many researchers using different optimization methods. Table 18.8 shows that the different optimization methods converged almost to the

**Table 18.7** Member stress limits for the 25-bar spatial truss

Element group	Compressive stress limits ksi (MPa)	Tensile stress limits 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)

same structural weight. Figure 18.13 shows the convergence curve of the best result of CPA together with the mean performance of the algorithm for the 25-bar spatial truss.

#### 18.4.2.3 Design of a 72-Bar Spatial Truss

Weight optimization of a spatial 72-bar truss structure shown in Fig. 18.14 is considered as the third truss design example. The two loading conditions acting on the structure are summarized in Table 18.9. The elements are grouped to form 16 design variables according to Table 18.10. The material density and the modulus of elasticity are taken as 0.1 lb/in<sup>3</sup> and 10,000 ksi, respectively. All members are subjected to a stress limitation of  $\pm 25$  ksi. The displacements of the uppermost nodes along x- and y-axes are limited to  $\pm 0.25$  in. Cross-sectional areas of bars can vary between 0.10 and 4.00 in<sup>2</sup>, respectively.

This problem has been studied by Erbatur et al. [16] using genetic algorithms, Camp and Bichon [17] using ant colony optimization, Perez and Behdinan [18] using particle swarm optimization, Camp [15] using Big Bang–Big Crunch algorithm, Kaveh and Khayatazad [19] using ray optimization, Degertekin using variants of harmony search [10], and Degertekin and Hayalioglu using teaching–learning-based optimization [11], among others.

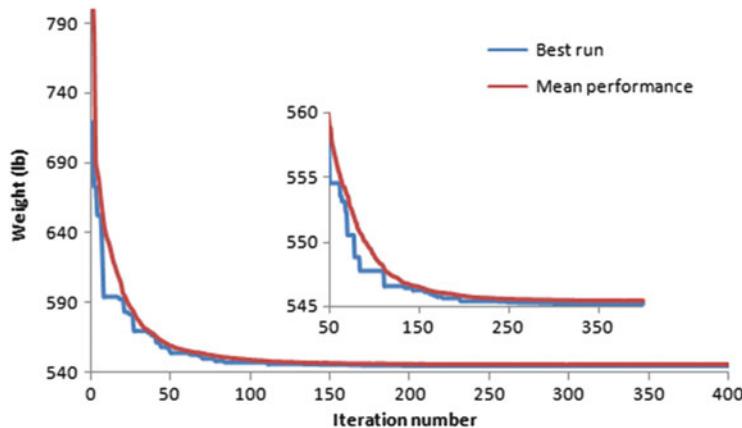
Table 18.11 compares the results obtained by the present method to those previously reported in the literature. It can be seen that the present method have obtained the lightest design with a weight of 379.62 lb. Figure 18.15 presents the convergence curve for the best result and the mean performance of the CPA on 50 independent runs on this example.

#### 18.4.2.4 Design of a 120-Bar Dome Truss

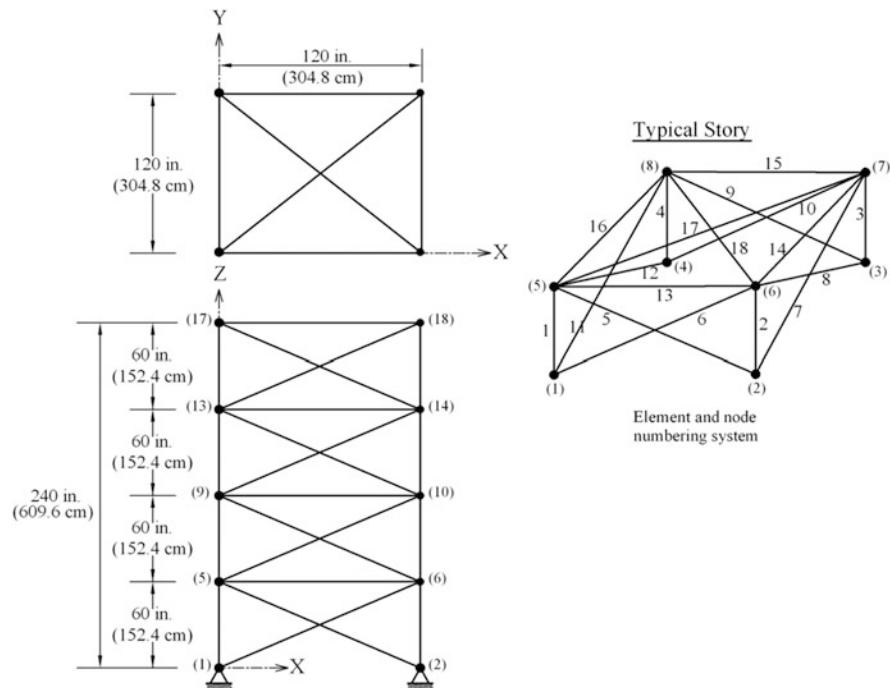
The fourth test problem is the weight minimization of a 120-bar dome truss shown in Fig. 18.16. This structure was considered by Soh and Yang [20] as a configuration optimization problem. It has been solved by Lee and Geem [21], Kaveh et al.

Table 18.8 Comparison of the optimization results obtained in the spatial 25-bar truss problem

Element group	Optimal cross-sectional areas (in <sup>2</sup> )				Degerterkin [10]	Degerterkin and Hayalioglu [11]	Present work	
	Li et al. [14]	PSO	PSOPC	HPSO				
1 A1	9.863	0.010	0.010	0.010	0.010	0.010	0.010	0.010
2 A2–A5	1.798	1.979	1.970	2.092	1.995	2.074	2.0712	1.989
3 A6–A9	3.654	3.011	3.016	2.964	2.980	2.961	2.9570	2.988
4 A10–A11	0.100	0.100	0.010	0.010	0.010	0.010	0.0100	0.010
5 A12–A13	0.100	0.100	0.010	0.010	0.010	0.010	0.0100	0.010
6 A14–A17	0.596	0.657	0.694	0.689	0.696	0.691	0.6891	0.689
7 A18–A21	1.659	1.678	1.681	1.601	1.679	1.617	1.6209	1.678
8 A22–A25	2.612	2.693	2.643	2.686	2.652	2.674	2.6768	2.658
Best weight (lb)	627.08	545.27	545.19	545.38	545.49	545.12	545.09	545.18
Average weight (lb)	N/A	N/A	N/A	545.78	546.52	545.94	545.41	545.49
Std Dev (lb)	N/A	N/A	N/A	0.491	1.05	0.91	0.42	0.24
No. of analyses	150,000	150,000	150,000	20,566	10,391	9051	15,318	22800



**Fig. 18.13** Convergence curves of the best result of the CPA together with the mean performance of the algorithm for the 25-bar spatial truss



**Fig. 18.14** Schematic of the spatial 72-bar truss structure

**Table 18.9** Independent loading conditions acting on the spatial 72-bar truss

Node	Case 1			Case 2		
	P <sub>x</sub> kips	P <sub>y</sub> kips	P <sub>z</sub> kips	P <sub>x</sub> kips	P <sub>y</sub> kips	P <sub>z</sub> kips
1	5	5	-5	-	-	-5
2	-	-	-	-	-	-5
3	-	-	-	-	-	-5
4	-	-	-	-	-	-5

[22], Kaveh and Khayatazzad [19], and Kaveh and Mahdavi [23] as a sizing optimization problem. The members of the structure are divided into seven groups as shown in Fig. 18.16.

The allowable tensile and compressive stresses are set according to the ASD-AISC [24] provisions as follows:

$$\begin{cases} \sigma_i^+ = 0.6F_y & \text{for } \sigma_i \geq 0 \\ \sigma_i^- & \text{for } \sigma_i \leq 0 \end{cases} \quad (18.5)$$

where  $\sigma_i^-$  is the compressive allowable stress and depends on the slenderness ratios of the elements.

$$\sigma_i^- = \begin{cases} \left[ \left( 1 - \frac{\lambda_i^2}{2C_c^2} \right) F_y \right] \Big/ \left( \frac{5}{3} + \frac{3\lambda_i}{8C_c} - \frac{\lambda_i^3}{8C_c^3} \right) & \text{for } \lambda_i \leq C_c \\ \frac{12\pi^2 E}{23\lambda_i^2} & \text{for } \lambda_i \geq C_c \end{cases} \quad (18.6)$$

where  $E$  is the modulus of elasticity,  $F_y$  is the material's yield stress,  $\lambda_i$  is the slenderness ratio ( $\lambda_i = \frac{K_i L_i}{r_i}$ ),  $K_i$  is the effective length factor,  $L_i$  is the length of the member, and  $r_i$  is the radius of gyration.  $C_c$  is the critical slenderness ratio separating elastic and inelastic buckling regions ( $C_c = \sqrt{2\pi^2 E/F_y}$ ).

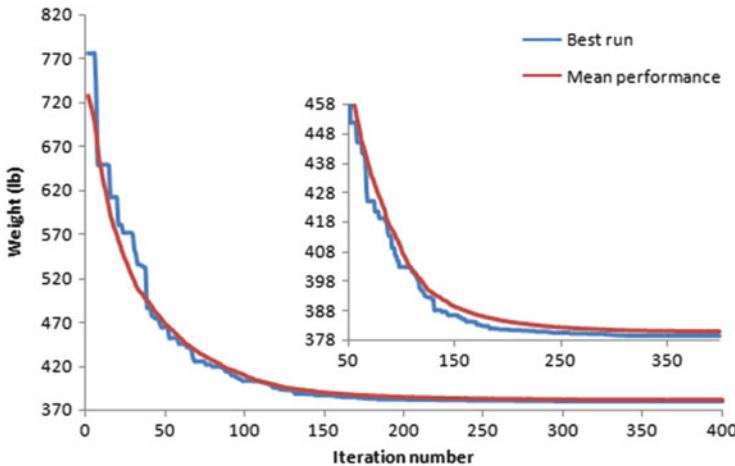
The modulus of elasticity and the material density are taken as 30,450 ksi (210 GPa) and 0.288 lb/in<sup>3</sup>, respectively. The yield stress is taken as 58.0 ksi (400 MPa). The radius of gyration is expressed in terms of cross-sectional areas of the members as  $r_i = aA_i^b$  [25]. Constants  $a$  and  $b$  depend on the types of sections adopted for the members such as pipes, angles, etc. In this example pipe sections are used for the bars for which  $a = 0.4993$  and  $b = 0.6777$ . The dome is considered to be subjected to vertical loads at all unsupported nodes. These vertical 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 other nodes. Four different problem variants are considered for this structure: with stress constraints and no displacement constraints (Case 1), with stress constraints and displacement limitations of  $\pm 0.1969$  in (5 mm)

**Table 18.10** Comparison of the optimization results obtained in the spatial 72-bar truss problem

Element group	GA	Optimal cross-sectional areas (in <sup>2</sup> )						Degertekin and Hoyalioglu [11]	Present work
		Erbatu et al. [16]	Camp and Bichon [17]	Perez and Behdinan [18]	Camp [15]	Kavesh and Khayyatzad [19]	Degertekin [10]		
	ACO	PSO	BB- BC	RO	EHS	SAHS	TLBO	CPA	
1–4	1.755	1.948	1.7427	1.8577	1.8365	1.967	1.860	1.9064	1.8873
5–12	0.505	0.508	0.5185	0.5059	0.5021	0.510	0.521	0.50612	0.5111
13–16	0.105	0.101	0.1000	0.1000	0.1000	0.100	0.100	0.100	0.1000
17–18	0.155	0.102	0.1000	0.1000	0.1004	0.100	0.100	0.100	0.1000
19–22	1.155	1.303	1.3079	1.2476	1.2522	1.293	1.271	1.2617	1.2554
23–30	0.585	0.511	0.5193	0.5269	0.5033	0.511	0.509	0.5111	0.5141
31–34	0.100	0.101	0.1000	0.1000	0.1002	0.100	0.100	0.100	0.1000
35–36	0.100	0.100	0.1000	0.1012	0.1001	0.100	0.100	0.100	0.1000
37–40	0.460	0.561	0.5142	0.5209	0.5730	0.499	0.485	0.5317	0.5312
41–48	0.530	0.492	0.5464	0.5172	0.5499	0.501	0.501	0.51591	0.5174
49–52	0.120	0.100	0.1000	0.1004	0.1004	0.100	0.100	0.100	0.1000
53–54	0.165	0.107	0.1095	0.1005	0.1001	0.100	0.100	0.100	0.1000
55–58	0.155	0.156	0.1615	0.1565	0.1576	0.160	0.168	0.1562	0.1564
59–66	0.535	0.550	0.5092	0.5507	0.5222	0.522	0.584	0.54927	0.5443
67–70	0.480	0.390	0.4967	0.3922	0.4356	0.478	0.433	0.40966	0.4106
71–72	0.520	0.592	0.5619	0.5922	0.5971	0.591	0.520	0.56976	0.5717
Best weight (lb)	385.76	380.24	381.91	379.85	380.458	381.03	380.62	379.63	379.62
Mean weight (lb)	N/A	383.16	N/A	382.08	382.553	383.51	382.42	380.20	380.83
Standard devi- ation (lb)	N/A	3.66	N/A	1.912	1.221	1.92	1.38	0.41	0.61
Number of analyses	N/A	18,500	N/A	19,621	19,084	15,044	13,742	19,778	23,580

**Table 18.11** Comparison of the optimization results obtained by CPA in the 120-bar dome problem

Optimal cross-sectional areas (in <sup>2</sup> )						
Case 1						
Element group	HS [21]	RO [19]	CBO [23]	Present work	HS [21]	Case 2
1	3.295	3.128	3.1229	3.1229	3.296	3.084
2	3.396	3.357	3.3538	3.3538	2.789	3.360
3	3.874	4.114	4.1120	4.1120	3.872	4.093
4	2.571	2.783	2.7822	2.7822	2.570	2.762
5	1.150	0.775	0.7750	0.7750	1.149	1.593
6	3.331	3.302	3.3005	3.3005	3.331	3.294
7	2.784	2.453	2.4458	2.4458	2.781	2.434
Best weight (lb)	19707.77	19476.193	19454.7	19454.68	19893.34	20071.9
Average weight (lb)	—	—	19466.0	19454.72	—	—
Std (lb)	—	33.966	7.02	0.032	—	112.135
Case 3						
Element group	RO [19]	CBO [23]	Present work	RO [19]	RO [22]	CBO [23]
1	2.044	2.0660	1.9617	3.030	3.0252	3.0273
2	15.665	15.9200	15.3594	14.806	14.8354	15.1724
3	5.848	5.6785	5.8966	5.440	5.1139	5.2342
4	2.290	2.2987	2.1967	3.124	3.1305	3.1119
5	9.001	9.0581	9.4318	8.021	8.4037	8.1038
6	3.673	3.6365	3.5973	3.614	3.3315	3.4166
7	1.971	1.9320	1.9498	2.487	2.4968	2.4918
Best weight (lb)	31733.2	31724.1	31681.11	33317.8	33256.48	33286.3
Average weight (lb)	—	32162.4	31701.56	—	—	33398.5
Std (lb)	274.991	240.22	15.52	354.333	—	67.09
Case 4						
Element group	HS [21]	RO [19]	CBO [23]	Present work	HS [21]	Case 2
1	3.3526	3.3526	3.3526	3.3526	3.3526	3.3526
2	4.0928	4.0928	4.0928	4.0928	4.0928	4.0928
3	2.7613	2.7613	2.7613	2.7613	2.7613	2.7613
4	1.5923	1.5923	1.5923	1.5923	1.5923	1.5923
5	3.2928	3.2928	3.2928	3.2928	3.2928	3.2928
6	2.4336	2.4336	2.4336	2.4336	2.4336	2.4336
7	2.4975	2.4975	2.4975	2.4975	2.4975	2.4975
Best weight (lb)	33286.3	33286.3	33286.3	33286.3	33286.3	33286.3
Average weight (lb)	33398.5	33398.5	33398.5	33398.5	33398.5	33398.5
Std (lb)	16.45	16.45	16.45	16.45	16.45	16.45



**Fig. 18.15** Convergence curves for the best result and the mean performance of 20 independent runs for the 72-bar spatial truss

imposed on all nodes in x- and y-directions (Case 2), no stress constraints and displacement limitations of  $\pm 0.1969$  in (5 mm) imposed on all nodes in z direction (Case 3), and all the above mentioned constraints imposed together (Case 4). For Cases 1 and 2, the maximum cross-sectional area is taken as  $5.0 \text{ in}^2$  ( $32.26 \text{ cm}^2$ ) while for Cases 3 and 4, it is taken as  $20 \text{ in}^2$  ( $129.03 \text{ cm}^2$ ). The minimum cross-sectional area is taken as  $0.775 \text{ in}^2$  ( $5 \text{ cm}^2$ ) for all cases.

Table 18.12 compares the results obtained by different optimization techniques for this example. It can be seen that CPA obtains the best results in Cases 1, 3, and 4 among the compared methods.

#### 18.4.2.5 Design of a 200-Bar Planar Truss

A planar 200-bar truss as shown in Fig. 18.17 is optimized as the last test problem. The elastic modulus of the material is 30,000 ksi while density is  $0.283 \text{ lb/in}^3$ . The allowable stress for all members is 10 ksi (the same in tension and compression). No displacement constraints are included in the optimization process. The structure is divided into 29 groups of elements. The minimum cross-sectional area of all design variables is taken as  $0.1 \text{ in}^2$ . This truss is subjected to three independent loading conditions: (1) 1.0 kip acting in the positive x direction at nodes 1, 6, 15, 20, 29, 43, 48, 57, 62, and 71; (2) 10.0 kips 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) loading conditions 1 and 2 acting together. The members of the structure are linked together in 29 groups according to Table 18.12.

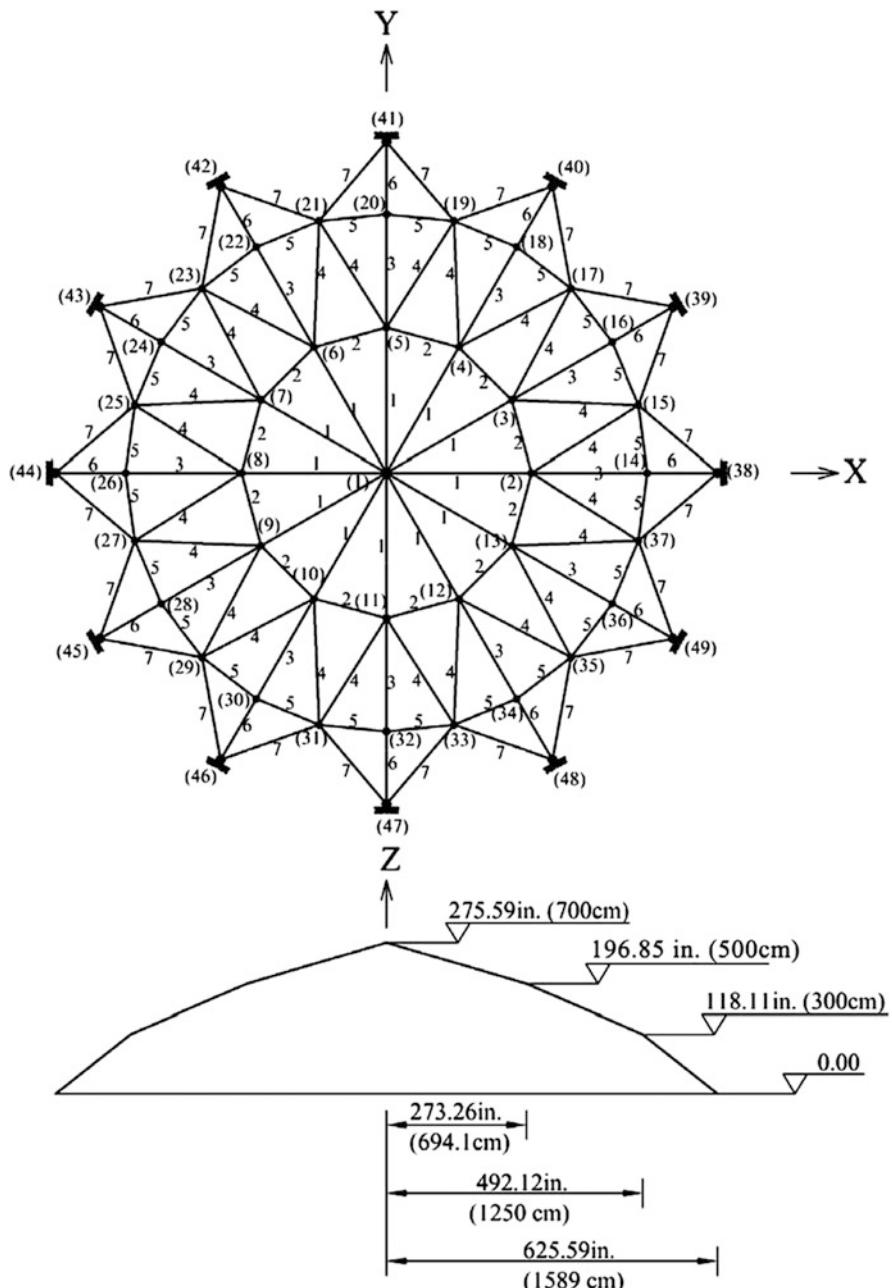
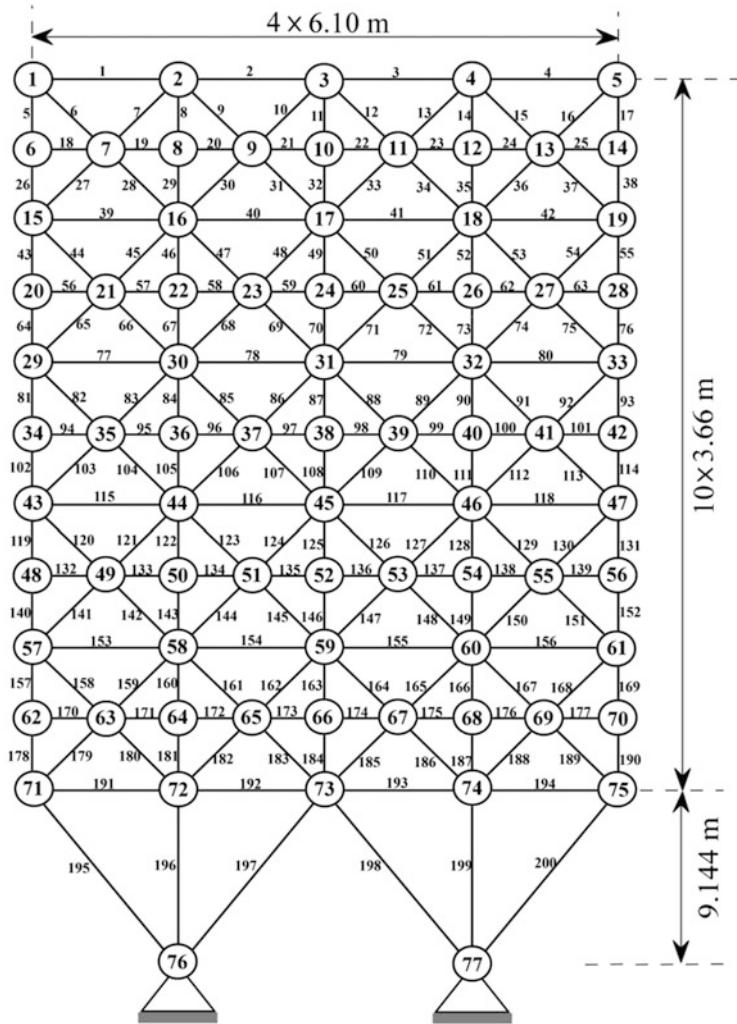


Fig. 18.16 Schematic of the 120-bar dome truss structure

**Table 18.12** Comparison of optimization results obtained by CPA and other metaheuristic algorithms for the 200-bar truss problem

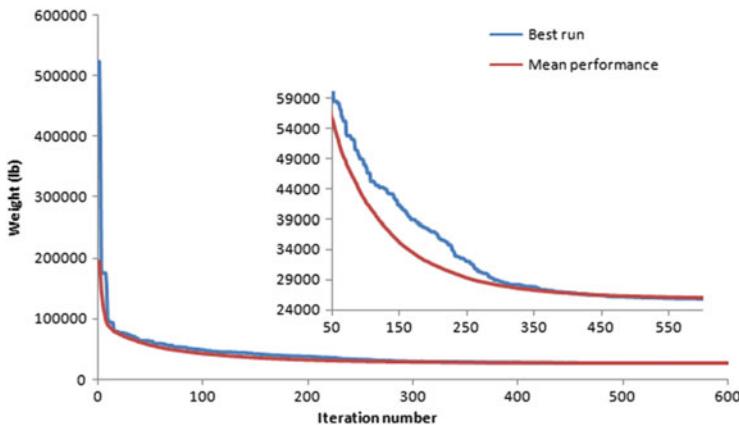
Element group	Members	Optimal cross-sectional areas (in <sup>2</sup> )						Present method
		CMLPSA [26]	SAHS [10]	TLBO [11]	HPSO [27]	WEO [13]		
1	1, 2, 3, 4	0.1468	0.1540	0.1460	0.1213	0.1144	0.1721	
2	5, 8, 11, 14, 17	0.9400	0.9410	0.9410	0.9426	0.9443	0.9553	
3	19, 20, 21, 22, 23, 24	0.1000	0.1000	0.1000	0.1220	0.1310	0.1000	
4	18, 25, 56, 63, 94, 101, 132, 139, 170, 177	0.1000	0.1000	0.1010	0.1000	0.1016	0.1004	
5	26, 29, 32, 35, 38	1.9400	1.9420	1.9410	2.0143	2.0353	1.9662	
6	6, 7, 9, 10, 12, 13, 15, 16, 27, 28, 30, 31, 33, 34, 36, 37	0.2962	0.3010	0.2960	0.2800	0.3126	0.3055	
7	39, 40, 41, 42	0.1000	0.1000	0.1000	0.1589	0.1679	0.1000	
8	43, 46, 49, 52, 55	3.1042	3.1080	3.1210	3.0666	3.1541	3.1618	
9	57, 58, 59, 60, 61, 62	0.1000	0.1000	0.1000	0.1002	0.1003	0.1152	
10	64, 67, 70, 73, 76	4.1042	4.1060	4.1730	4.0418	4.1005	4.2405	
11	44, 45, 47, 48, 50, 51, 53, 54, 65, 66, 68, 69, 71, 72, 74, 75	0.4034	0.4090	0.4010	0.4142	0.4350	0.4046	
12	77, 78, 79, 80	0.1912	0.1910	0.1810	0.4852	0.1148	0.1000	
13	81, 84, 87, 90, 93	5.4284	5.4280	5.4230	5.4196	5.3823	5.4132	
14	95, 96, 97, 98, 99, 100	0.1000	0.1000	0.1000	0.1000	0.1607	0.1545	
15	102, 105, 108, 111, 114	6.4284	6.4270	6.4220	6.3749	6.4152	6.3976	
16	82, 83, 85, 86, 88, 89, 91, 92, 103, 104, 106, 107, 109, 110, 112, 113	0.5734	0.5810	0.5710	0.6813	0.5629	0.5555	
17	115, 116, 117, 118	0.1327	0.1510	0.1560	0.1576	0.4010	0.4425	
18	119, 122, 125, 128, 131	7.9717	7.9730	7.9580	8.1447	7.9735	8.0928	
19	133, 134, 135, 136, 137, 138	0.1000	0.1000	0.1000	0.1000	0.1092	0.1004	
20	140, 143, 146, 149, 152	8.9717	8.9740	8.9580	9.0920	9.0155	8.9918	
21	120, 121, 123, 124, 126, 127, 129, 130, 141, 142, 144, 145, 147, 148, 150, 151	0.7049	0.7190	0.7200	0.7462	0.8628	0.8925	
22	153, 154, 155, 156	0.4196	0.4220	0.4780	0.2114	0.2220	0.2544	

23	157, 160, 163, 166, 169	10.8636	10.8920	10.8970	10.9587	11.0254	11.1214
24	171, 172, 173, 174, 175, 176	0.1000	0.1000	0.1000	0.1000	0.1397	0.1000
25	178, 181, 184, 187, 190	11.8606	11.8870	11.8970	11.9832	12.0340	12.3304
26	158, 159, 161, 162, 164, 165, 167, 168, 179, 180, 182, 183, 185, 186, 188, 189	1.0339	1.0400	1.0800	0.9241	1.0043	1.0110
27	191, 192, 193, 194	6.6818	6.6460	6.4620	6.7676	6.5762	6.4103
28	195, 197, 198, 200	10.8113	10.8040	10.7990	10.9639	10.7265	10.5814
29	196, 199	13.8404	13.8700	13.9220	13.8186	13.9666	14.1288
Best Weight (lb)		25445.63	25491.9	25488.15	25698.85	25674.83	25651.58
Average weight (lb)		N/A	25610.2	25533.14	28386.72	26613.45	25957.15
Std (lb)		N/A	141.85	27.44	2403	702.80	254.06
No. of structural analyses		9650	19,670	28,059	14,406	19,410	34,560



**Fig. 18.17** Schematic of a 200-bar planar truss

Table 19.12 presents the optimum designs obtained by CPA, WEO [13], a Corrected Multi-Level and Multi-Point Simulated Annealing algorithm (CMLPSA) [26], SAHS [10], TLBO [11], and HPSSO [27]. Figure 18.18 shows the convergence curves for the best result and the mean performance of 50 independent runs for the 200-bar planar truss.



**Fig. 18.18** Convergence curves for the best result and the mean performance of 50 independent runs for the 200-bar planar truss

## 18.5 Concluding Remarks

In this chapter a new population-based metaheuristic optimization method, namely, cyclical parthenogenesis algorithm (CPA), is presented. The algorithm is inspired by reproduction and social behavior of some zoological species like aphids, which alternate between sexual and asexual reproduction. CPA starts with a random population, considered as aphids, and iteratively improves the quality of solutions utilizing reproduction and displacement mechanisms.

Some mathematical and benchmark truss design problems are employed in order to investigate the viability of the proposed algorithm. Sensitivity of the proposed algorithm to its parameters is analyzed and the convergence behavior of the algorithm is studied using the diversity index. The results of the numerical examples indicate that the performance of the newly proposed algorithm is comparable to other state-of-the-art metaheuristic algorithms. CPA has found many interesting applications in design of structures, in particular in optimal design of truss structures with constraints as natural frequencies [28].

## References

1. Kaveh A, Zolghadr A (2016) Cyclical parthenogenesis algorithm: a new metaheuristic algorithm. *Adv Eng Softw* (in press)
2. Piper R (2007) Extraordinary animals: an encyclopedia of curious and unusual animals. Greenwood Press, Westport, CT
3. Dixon AFG (1998) Aphid ecology: an optimization approach, 2nd edn. Blackie and Son, Glasgow
4. McGavin GC (1999) Bugs of the World. Blandford, London

5. Dawson KJ (1995) The advantage of asexual reproduction: when is it twofold? *J Theor Bio* 176 (3):341–347
6. Williams GC (1975) Sex and evolution. Princeton University Press, Princeton, NJ
7. Kaveh A, Zolghadr A (2014) Comparison of nine meta-heuristic algorithms for optimal design of truss structures with frequency constraints. *Adv Eng Softw* 76:9–30
8. Tsoulos IG (2008) Modifications of real code genetic algorithm for global optimization. *Appl Math Comput* 203:598–607
9. Sonmez M (2011) Artificial Bee Colony algorithm for optimization of truss structures. *Appl Soft Comput* 11:2406–2418
10. Degertekin SO (2012) Improved harmony search algorithms for sizing optimization of truss structures. *Comput Struct* 92–93:229–241
11. Degertekin SO, Hayalioglu MS (2013) Sizing truss structures using teaching-learning-based optimization. *Comput Struct* 119:177–188
12. Talatahari S, Kheirollahi M, Farahmandpour C, Gandomi AH (2013) A multi-stage particle swarm for optimum design of truss structures. *Neural Comput Applic* 23:1297–1309
13. Kaveh A, Bakhshpoori T (2016) Water evaporation optimization: a novel physically inspired optimization algorithm. *Comput Struct* 167:69–85
14. 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
15. Camp CV (2007) Design of space trusses using Big Bang–Big Crunch optimization. *ASCE J Struct Eng* 133:999–1008
16. Erbatur F, Hasançebi O, Tütüncü I, Kılıç H (2000) Optimal design of planar and space structures with genetic algorithms. *Comput Struct* 75:209–224
17. Camp CV, Bichon J (2004) Design of space trusses using ant colony optimization. *ASCE J Struct Eng* 130:741–751
18. Perez RE, Behdinan K (2007) Particle swarm approach for structural design optimization. *Comput Struct* 85:1579–1588
19. Kaveh A, Khayatazad M (2012) A novel meta-heuristic method: ray optimization. *Comput Struct* 112–113:283–294
20. Soh CK, Yang J (1996) Fuzzy controlled genetic algorithm search for shape optimization. *ASCE J Comput Civil Eng* 10:143–150
21. Lee KS, Geem ZW (2004) A new structural optimization method based on the harmony search algorithm. *Comput Struct* 82:781–798
22. Kaveh A, Ilchi Ghazaan M, Bakhshpoori T (2013) An improved ray optimization algorithm for design of truss structures. *Period Polytech Civil Eng* 57:97–112
23. Kaveh A, Mahdavi VR (2014) Colliding bodies optimization: a novel meta-heuristic method. *Comput Struct* 139:18–27
24. American Institute of Steel Construction (AISC) (1989) Manual of steel construction allowable stress design, 9th edn. Chicago, Illinois
25. Saka MP (1990) Optimum design of pin-jointed steel structures with practical application. *ASCE J Struct Eng* 116:2599–2620
26. Lamberti L (2008) An efficient simulated annealing algorithm for design optimization of truss structures. *Comput Struct* 86:1936–1953
27. Kaveh A, Bakhshpoori T, Afshari E (2014) An efficient hybrid particle swarm and swallow swarm optimization algorithm. *Comput Struct* 143:40–59
28. Kaveh A, Zolghadr A (2016) Cyclical parthenogenesis algorithm for shape and size optimization of truss structures with frequency constraints. *Eng Optim* (in press)

# Chapter 19

## Optimal Design of Large-Scale Frame Structures

### 19.1 Introduction

Discrete or continuous size optimization of large-scale, high-rise, or complex structures leads to problems with large number of design variables and large search spaces and requires the control of a great number of design constraints. Separate design decisions for each variable would be allowed. Thus, the optimizer invoked to process such a sizing problem is given the possibility to really optimize the objective function by detecting the optimum solution within a vast amount of possible design options. The huge number of available design options typically confuses an optimizer and radically decreases the potential of effective search for a high-quality solution. This chapter is based on the recent development on design of large-scale frame structures (Kaveh and Bolandgerami [1]).

Various optimization approaches have been investigated and successfully applied to optimum design of large-scale structures in the recent years. Classical optimization algorithms (Schulz and Book [2]; Dreyer et al. [3]; Wang and Arora [4]) for large-scale problems need many powerful computational systems. Furthermore, many of such algorithms are known as local optimizer, and their final results cannot be considered as the global optimum. Contrary to mathematical programming algorithms, there are metaheuristic algorithms which are often stochastic algorithms and can efficiently explore the search space of the large-scale problems.

Yang et al. [5] proposed a new cooperative coevolution framework that is capable of optimizing large-scale non-separable problems. A random grouping scheme and adaptive weighting are introduced in problem decomposition and coevolution. Instead of conventional evolutionary algorithms, a novel differential evolutionary algorithm was adopted. Hsieh et al. [6] presented a variation on the traditional PSO algorithm, so-called the efficient population utilization strategy for particle swarm optimization (EPUS-PSO) for solving large-scale global optimization. This is achieved by using variable particles in swarms to enhance the searching ability and drive particles more efficiently. Moreover, sharing principals are

constructed to stop particles from falling into the local minimum and make the global optimal solution easier to find by particles. Fister et al. [7] used memetic computation (MC) which is emerged recently as a new paradigm of efficient algorithms for solving the hardest optimization problems. Artificial bee colony algorithm and memetic computation are used under the same roof. As a result, a memetic artificial bee colony algorithm (MABC) algorithm has been developed to solve large-scale global optimization problems. Self-organizing migrating algorithm with quadratic interpolation (SOMAQI) has been extended by Singha and Agrawalb [8] to solve large-scale global optimization problems for dimensions ranging from 100 to 3000 with a constant population size of 10 only. This produces high-quality optimal solution with very low computational cost and converges very fast to optimal solution.

In particular, some researchers have performed optimization on large-scale structures. Kaveh and Talatahari [9] introduced a modified version of the charged system search for large structure optimization which was based on the combination of charged system search algorithm and particle swarm optimization. Lagaros [10] presented a computing platform for real-world and large-scale structures. Talatahari and Kaveh [11] introduced an improved bat algorithm for optimizing large-scale structures. Aydogdu et al. [12] improved the performance of artificial bee colony algorithm by adding Lévy flight distribution in the search of scout bees and successfully applied to large steel space frames. Except improving the algorithm, other methods have been proposed. For example, Papadrakakis et al. [13] used neural network in order to replace the structural analysis phase and to compute the necessary data for the evolution strategies optimization procedure. The use of neural network was motivated by the time-consuming repeated analyses required by ES during the optimization process.

In order to overcome the dilemma of using large design variables, in addition to modifying or introducing new optimization algorithms, some methods have been proposed in literature. Sobiesczanski-Sobieski et al. [14] introduced a multilevel optimization and generalized multilevel optimization (Sobiesczanski-Sobieski et al. [15]) which break large optimization problems into several smaller subproblems, and a coordination problem is formulated to preserve the couplings among these subproblems. A very important benefit of such an approach, in addition to making the entire problem more tractable, is preservation of the customary organization of the design office in which many engineers work concurrently on different parts of the problem. Charmpis et al. [16] employed the concept of cascading, allowing a single optimization problem to be tackled in a number of successive autonomous optimization stages. Under this context, several coarse versions of the same full-size database are formed, in order to utilize a different database in each cascade stage executed with an evolutionary optimization algorithm. The early optimization stages of the resulting multi-database cascade procedure make use of the coarsest database versions available and serve the purpose of the basic design space exploration. The last stages exploit finer databases (including the original full-size database) and aim in fine-tuning to achieve optimal solution.

Optimum design of large-scale dome trusses using cascade optimization has been carried out by Kaveh and Ilchi Ghazaan [17].

Cascade sizing optimization utilizing a series of design variable configurations (DVCs) is used in this study. Several design variable configurations are constructed, in order to utilize a different configuration at each cascade optimization stage. Each new cascade stage is coupled with the previous one by initializing the new stage using the finally attained optimum design of the previous one. The early optimization stages of the cascade procedure make use of the coarsest configurations with small numbers of design variables and serve the purpose of basic design space exploration. The last stages exploit finer configurations with larger numbers of design variables and aim in fine-tuning the achieved optimal solution.

Utilized optimizer in all stages of the cascade process is enhanced colliding bodies optimization (ECBO) which is introduced by Kaveh and Ilchi Ghazaan [18]. However, any other metaheuristic algorithm could be used. Colliding bodies optimization (CBO) which is introduced by Kaveh and Mahdavi [19] is a new multi-agent algorithm inspired by a collision between two objects in one dimension. Each agent is modeled as a body with a specified mass and velocity. A collision occurs between pairs of objects, and the new positions of the colliding bodies are updated based on the collision laws. Enhanced colliding bodies optimization (ECBO) uses memory to save some best solutions and has a mechanism to escape from local optima.

The present chapter is organized as follows. Code-based design optimization of steel frames is presented in Sect. 19.2. Section 19.3 introduces cascade sizing optimization utilizing a series of design variable configurations. CBO algorithm and its enhanced version are introduced in Sect. 19.4. In Sect. 19.5, optimal designs of three large-scale space frames with the proposed approach are investigated. Finally, some conclusions are derived in Sect. 19.6.

## 19.2 Code-Based Design Optimization of Steel Frames

For a steel frame structure consisting of  $N_m$  members that are collected in  $N_d$  design groups (variables), the optimum design problem according to ASD-AISC (American Institute of Steel Construction [20]) code yields the following discrete programming problem, if the design groups are selected from steel sections in a given profile list.

Find a vector of integer values  $\mathbf{I}$  representing the sequence numbers of steel sections assigned to  $N_d$  member groups as:

$$\mathbf{I}^T = [I_1, I_2, \dots, I_{N_d}] \quad (19.1)$$

to minimize the weight ( $W$ ) of the frame:

$$W = \sum_{i=1}^{N_d} \rho_i A_i \sum_{j=1}^{N_t} L_j \quad (19.2)$$

where  $A_i$  and  $\rho_i$  are the area and unit weight of the steel section adopted for member group  $i$ , respectively,  $N_t$  is the total number of members in group  $i$ , and  $L_j$  is the length of the member  $j$  which belongs to the group  $i$ .

The members subjected to a combination of axial compression and flexural stress must be sized to meet the following stress constraints:

$$\text{if } \frac{f_a}{F_a} > 0.15; \quad \left[ \frac{f_a}{F_a} + \frac{C_{mx} f_{bx}}{\left(1 - \frac{f_a}{F'_{ex}}\right) F_{bx}} + \frac{C_{my} f_{by}}{\left(1 - \frac{f_a}{F'_{ey}}\right) F_{by}} \right] - 1 \leq 0 \quad (19.3)$$

$$\left[ \frac{f_a}{0.60 F_y} + \frac{f_{bx}}{F_{bx}} + \frac{f_{by}}{F_{by}} \right] - 1 \leq 0 \quad (19.4)$$

$$\text{if } \frac{f_a}{F_a} < 0.15; \quad \left[ \frac{f_a}{F_a} + \frac{f_{bx}}{F_{bx}} + \frac{f_{by}}{F_{by}} \right] - 1 \leq 0 \quad (19.5)$$

If the flexural member is under tension, then the following formula is used instead:

$$\left[ \frac{f_a}{0.60 F_y} + \frac{f_{bx}}{F_{bx}} + \frac{f_{by}}{F_{by}} \right] - 1 \leq 0 \quad (19.6)$$

In Eqs. (19.3) and (19.6),  $F_y$  is the material yield stress, and  $f_a = (P/A)$  represents the computed axial stress, where  $A$  is the cross-sectional area of the member. 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 using Formulas 1.5-1 and 1.5-2 of ASD-AISC. The allowable bending compressive stresses about major and minor axes are designated by  $F_{bx}$  and  $F_{by}$ , which are computed using Formulas 1.5-6a or 1.5-6b and 1.5-7 provided in ASD-AISC.  $C_{mx}$  and  $C_{my}$  are the reduction factors, introduced to counterbalance overestimation of the effect of secondary moments by the amplification factor  $(1 - f_a/F'_e)$ . For braced frame members without transverse loading between their ends, these are calculated from  $C_m = 0.6 - 0.4(M_1/M_2)$ , where  $M_1/M_2$  is the ratio of smaller end moment to the larger end moment. For braced frame members having transverse loading between their ends, these are determined from the formula  $C_m = 1 + \psi(f_a/F'_e)$  based on a rational approximate analysis outlined in ASD-AISC Commentary-H1, where  $\psi$  is a parameter that considers maximum deflection and maximum moment in the member.

For computation of allowable compression and Euler stresses, the effective length factors  $K$  are required. For beam and bracing members,  $K$  is taken equal to unity. For column members, alignment charts furnished in ASD-AISC can be utilized. In this study, however, the effective length factors of columns in a braced frame are calculated from the following approximate formula developed by Dumonteil [21], which are accurate within about  $-1.0\%$  and  $+2.0\%$  of the exact results (Hellesland [22]):

$$K = \frac{3G_A G_B + 1.4(G_A + G_B) + 0.64}{3G_A G_B + 2.0(G_A + G_B) + 1.28} \quad (19.7)$$

where  $G_A$  and  $G_B$  refer to the stiffness ratio or relative stiffness of a column at its two ends.

It is also required that computed shear stresses ( $f_v$ ) in members should be smaller than the allowable shear stresses ( $F_v$ ), as formulated in Eq. (19.8):

$$f_v \leq F_v = 0.40C_v F_y \quad (19.8)$$

In the above equation,  $C_v$  is referred to as web shear coefficient. It is taken equal to  $C_v = 1.0$  for the rolled W-shaped members with  $h/t_w \leq 2.24E/F_y$ , where  $h$  is the clear distance between flanges,  $E$  is the elasticity modulus, and  $t_w$  is the thickness of web. For all other symmetric shapes,  $C_v$  is calculated from Formulas G2-3, G2-4, and G2-5 in ANSI/AISC 360-05 (Specification [23]).

Apart from stress constraints, slenderness limitations are also imposed on all members such that the maximum slenderness ratio ( $\lambda = KL/r$ ) is limited to 300 for members under tension and to 200 for members under compression loads. The displacement constraints are imposed such that the maximum lateral displacements are restricted to be less than  $H/400$  and the upper limit of story drift is set to be  $h/400$ , where  $H$  is the total height of the frame building and  $h$  is the height of a story.

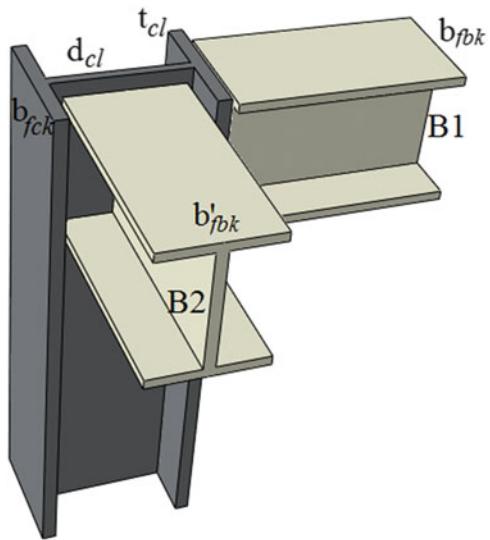
Finally, we consider geometric constraints between beams and columns framing into each other at a common joint for practicality of an optimum solution generated. For the two beams  $B1$  and  $B2$  and the column shown in Fig. 19.1, one can write the following geometric constraints:

$$\frac{b_{fb}}{b_{fc}} - 1.0 \leq 0 \quad (19.9)$$

$$\frac{b'_{fb}}{(d_c - 2t_f)} - 1.0 \leq 0 \quad (19.10)$$

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 (19.9) simply ensures that the flange width of the beam  $B1$  remains smaller than that of the column. On the other hand, Eq. (19.10) guarantees

**Fig. 19.1** Schematic of a beam–column geometric constraints [1]



that flange width of the beam  $B2$  remains smaller than clear distance between the flanges of the column ( $d_c - 2t_f$ ).

### 19.3 Cascade Sizing Optimization Utilizing a Series of Design Variable Configurations

Cascade sizing optimization approach utilized in this work is presented in this section. First, the concept of cascade optimization is introduced, and then multi-design variable configuration (DVC) cascade optimization is presented.

#### 19.3.1 Cascade Optimization Strategy

There is no unique optimization algorithm capable of effectively solving all optimization problems. Cascade optimization strategy has been introduced as a multi-stage procedure, which employs various optimizers in a successive manner to solve an optimization problem (Patnaik et al. [24]). Each autonomous optimization stage of the cascade procedure starts from an initial design, which is either a “cold start” or a “hot start.” The early stage starts from cold start which is a user-specified or randomly selected design. After running the early-stage optimizer, the optimal solution reached is used as the starting solution for the second cascade stage. This new starting solution is called a hot start, because during the execution of the initial optimizer, the achieved optimal solution has moved toward the region of the global

optimum. Then, each optimization stage of the cascade procedure starts from the optimum solution achieved at the previous stage.

The actual aim of cascading is to take advantage of the combined strength and the differentiated computations of a number of optimizers executed in a successive manner. This advantage has been used for structural sizing optimization (Charmpis et al. [16], Lagaros [10], Kaveh and Ilchi Ghazaan [17]).

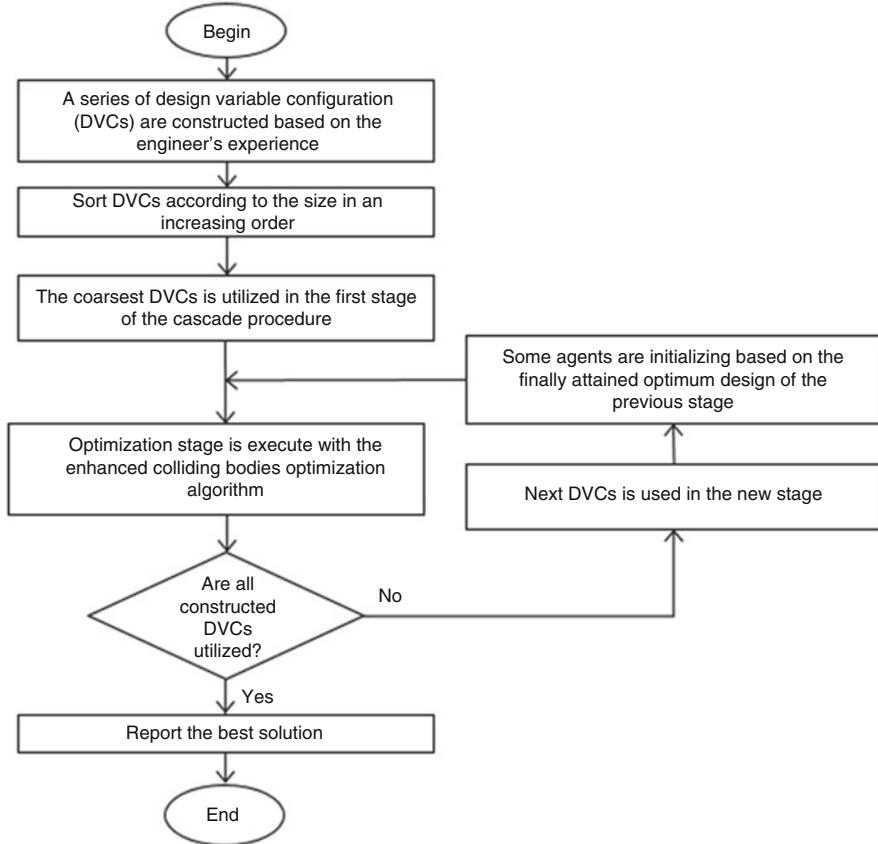
### 19.3.2 Multi-DVC Cascade Optimization

Multi-DVC cascade optimization method employs a series of configurations for the design variable configurations (DVCs) of the problem at hand and at each stage of cascade uses from different DVCs. DVCs are defined in a manner that DVCs used in early stages have less design variables than DVCs utilized in final stages. The early stage of the cascade procedure executed with the coarsest DVCs aims at a basic non-detailed search of the entire design space. This search is facilitated by the manageable DVCs handled, which avoid confusing the employed optimizer with huge design spaces although it does not consider a lot of design options. With this method, global search is performed in early stages and more detailed and local search is performed in final stages. The final stage of cascade optimization in this method uses the finest DVC. Figure 19.2 shows the flowchart of the multi-DVC cascade optimization procedure.

DVCs can be generated based on engineering experience. Here, variables of one kind (exterior beams, interior beams, corner columns, etc.) for some stories are integrated in coarser DVC, while in finer DVC structural members are grouped separately for each story.

For each stage of the cascade optimization method, one can choose different optimization algorithm. Choosing an optimization algorithm which is capable in global search for early stages and local search algorithms for final stages may result in a better cascade method performance. However, in this chapter enhanced colliding bodies optimization, presented in the subsequent section, is used as the optimization algorithm in all the stages of the cascade optimization.

In order to explain the utilized method (without loss of generality), consider three stages of the cascade procedure. In this method,  $C_0$  is the number of DVCs that is defined by the problem, so two other coarser DVCs should be defined ( $C_1$  and  $C_2$ ). In the first stage of the cascade procedure, the first optimizer works with  $C_2$  which is the coarsest DVC and finally attains  $d_I(C_2)$  design. In the first stage, optimizer deals with relatively small number of variables and thus in a reasonable number of iterations achieves the  $d_I(C_2)$  design.  $d_I(C_2)$  is a vector with  $n_d(C_2)$  entries, and before starting the second stage of cascade procedure,  $d_I(C_2)$  must be converted to the vector  $d_I(C_1)$  with  $n_d(C_1)$  entries. For this conversion, design information of each structural member should be extracted. Notation  $n_d$  refers to the number of design variables in each DVC. After conversion, the second optimizer in the second stage of the cascade procedure starts from  $d_I(C_1)$  and achieves



**Fig. 19.2** Flowchart of the multi-DVC cascade optimization procedure [1]

the  $d_2(C_1)$ . Due to the second optimizer efforts in the second stage of the cascade procedure,  $d_2$  will have better fitness value than  $d_1$ . Finally, in the third stage of the cascade method, the third optimizer starts from  $d_2(C_0)$  and achieves the  $d_3(C_0)$ . Here, the  $d_2(C_0)$  is the converted vector of the  $d_2(C_1)$ , and  $d_3$  is the best achieved design in all stages of the cascade procedure.

For basic and non-detailed search of all region of design space, the first stages of cascade procedure executed with the coarsest DVCs. Due to relatively small number of design variables in the coarsest DVCs, optimizer will not be confused and search can be facilitated by the manageable DVCs handled. Accordingly, the appropriate regions of design space are identified by detecting optimum solutions among the relatively limited design options provided. With increasing numbers of design variables in the next stages of cascade procedure, more detailed search will be available, and the optimizer is given the opportunity to improve the quality of the optimal solution reached. Optimal design vector over a number of cascade stages is upgraded gradually by the transfer of optimization results between successive

stages. Although in the last stages of cascade procedure optimizer deals with large number of design variables, the appropriate initialization of each cascade stage prevents the optimizer from being trapped in a process of purposeless and ineffective searching. Hence, the first optimization stages of the cascade procedure serve the purpose of basic design space exploration, while the last stages aim in fine-tuning the achieved optimal solution.

## 19.4 Colliding Bodies Optimization and Its Enhanced Version

### 19.4.1 A Brief Explanation of the CBO Algorithm

The colliding bodies optimization (CBO) is a metaheuristic algorithm introduced by Kaveh and Mahdavi [19] and contains a number of colliding bodies (CB) where each one collides with other objects to explore the search space. 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). In discrete problems, CBs are allowed to select discrete values from a list. Actually, real numbers are rounded to the nearest integer.

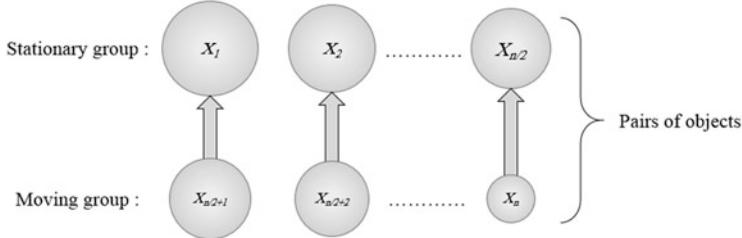
Each CB is a solution candidate so its objective function (here, weight of structure multiplied by the penalty function) value can be calculated. In this way, constraints are applied using a penalty function. CBs according to their objective function values take specified mass defined as:

$$m_k = \frac{\frac{1}{fit(k)}}{\sum_{i=1}^n \frac{1}{fit(i)}}, \quad k=1, 2, \dots, n \quad (19.11)$$

where  $fit(i)$  represents the objective function value of the  $i$ th CB and  $n$  is the number of colliding bodies. Thus, in each iteration, all the CBs must be evaluated. Here, finite element analysis constraints check must be performed.

After sorting colliding bodies according to their objective function values in an increasing order, two equal groups are created: (i) stationary group and (ii) moving group (Fig. 19.3). Moving objects collide to stationary objects to improve their positions and push stationary objects toward better positions. The velocities of the stationary and moving bodies before collision ( $v_i$ ) are calculated by:

$$v_i = 0, \quad i = 1, 2, \dots, \frac{n}{2} \quad (19.12)$$



**Fig. 19.3** Pairs of CBs for collision

$$v_i = x_{i-\frac{n}{2}} - x_i, \quad i = \frac{n}{2} + 1, \frac{n}{2} + 2, \dots, n \quad (19.13)$$

where  $x_i$  is the position vector of the  $i$ th CB.

The velocities of stationary and moving CBs after the collision ( $v'_i$ ) are evaluated by:

$$v'_i = \frac{\left(m_{i+\frac{n}{2}} + \epsilon m_{i-\frac{n}{2}}\right)v_{i+\frac{n}{2}}}{m_i + m_{i+\frac{n}{2}}} \quad i = 1, 2, \dots, \frac{n}{2} \quad (19.14)$$

$$v'_i = \frac{\left(m_i - \epsilon m_{i-\frac{n}{2}}\right)v_i}{m_i + m_{i-\frac{n}{2}}} \quad i = \frac{n}{2} + 1, \frac{n}{2} + 2, \dots, n \quad (19.15)$$

$$\epsilon = 1 - \frac{iter}{iter_{max}} \quad (19.16)$$

where  $\epsilon$  is the coefficient of restitution (COR) and  $iter$  and  $iter_{max}$  are the current iteration number and the total number of iterations for optimization process, respectively.

New positions of each group are stated by the following formulas:

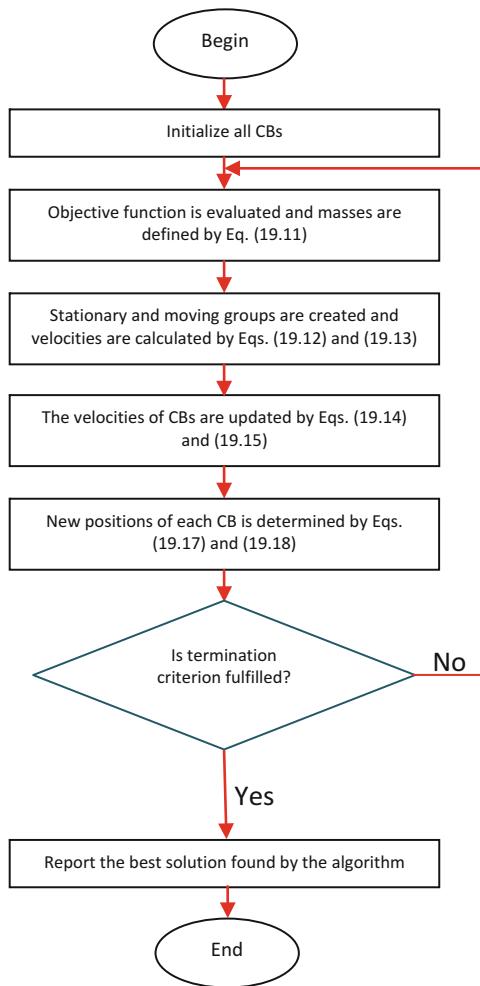
$$x_i^{new} = x_i + rand \circ v'_i, \quad i = 1, 2, \dots, \frac{n}{2} \quad (19.17)$$

$$x_i^{new} = x_{i-\frac{n}{2}} + rand \circ v'_i, \quad i = \frac{n}{2} + 1, \frac{n}{2} + 2, \dots, n \quad (19.18)$$

where  $x_i^{new}$ ,  $x_i$ , and  $v'_i$  are the new position, the previous position, and the velocity after the collision of the  $i$ th CB, respectively;  $rand$  is a random vector uniformly distributed in the range of  $(-1, 1)$  and the sign “ $\circ$ ” denotes an element-by-element multiplication.

The flowchart of CBO algorithm is depicted in Fig. 19.4.

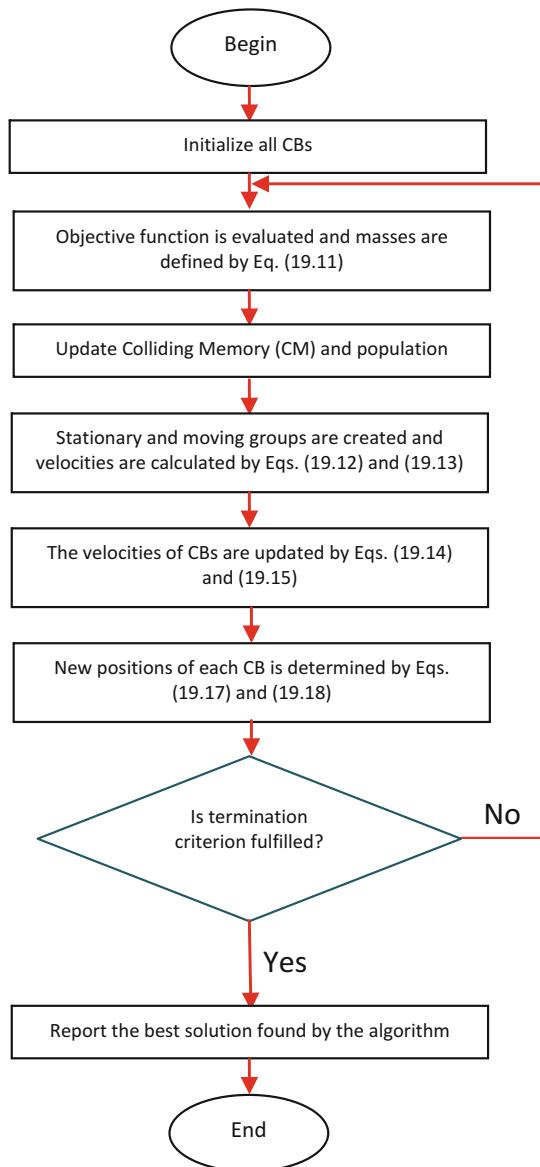
**Fig. 19.4** Flowchart of the CBO algorithm



### 19.4.2 The ECBO Algorithm

The enhanced version of the CBO is introduced by Kaveh and Ilchi Ghazaan [18]. In the enhanced colliding bodies optimization (ECBO), a memory that saves a number of historically best CBs is utilized to improve the performance of the CBO and to reduce the computational cost. Furthermore, ECBO changes some components of the CBs randomly to prevent premature convergence. In order to introduce the ECBO, the following steps are developed and the corresponding flowchart is provided in Fig. 19.5.

**Fig. 19.5** Flowchart of the ECBO algorithm



### Level 1: Initialization

**Step 1:** The initial locations of CBs are created randomly in an  $m$ -dimensional search space.

$$x_i^0 = x_{min} + \text{random} \circ (x_{max} - x_{min}), \quad i = 1, 2, \dots, n \quad (19.19)$$

where  $x_i^0$  is the initial solution vector of the  $i$ th CB,  $x_{min}$  and  $x_{max}$  are the minimum and the maximum allowable variable vectors, and random is a random vector with each component being in the interval  $[0, 1]$ .

### Level 2: Search

**Step 1:** The value of the mass for each CB is calculated by Eq. (19.11).

**Step 2:** Colliding memory (CM) is considered to save some historically best CB vectors and their related mass and objective function values. The size of the CM is taken as  $n/10$  in this study. At each iteration, solution vectors that are saved in the CM are added to the population, and the same number of the current worst CBs is deleted.

**Step 3:** CBs are sorted according to their objective function values in an increasing order. To select the pairs of CBs for collision, they are divided into two equal groups: (i) stationary group and (ii) moving group (Fig. 19.3).

**Step 4:** The velocities of stationary and moving bodies before collision are evaluated in Eqs. (19.12) and (19.13), respectively.

**Step 5:** The velocities of stationary and moving bodies after collision are calculated in Eqs. (19.14) and (19.15), respectively.

**Step 6:** The new location of each CB is evaluated by Eq. (19.17) or Eq. (19.18).

**Step 7:** A parameter like **pro** within  $(0, 1)$  is introduced and specified whether a component of each CB must be changed or not. For each CB **pro** is compared with  $rn_i$  ( $i = 1, 2, \dots, n$ ) which is a random number uniformly distributed within  $(0, 1)$ . If  $rn_i < \text{pro}$ , one dimension of the  $i$ th CB is selected randomly and its value is regenerated by:

$$x_{ij} = x_{j,min} + \text{random} \cdot (x_{j,max} - x_{j,min}) \quad (19.20)$$

where  $x_{ij}$  is the  $j$ th variable of the  $i$ th CB and  $x_{j,min}$  and  $x_{j,max}$  are the lower and upper bounds of the  $j$ th variable.

### Level 3: Termination Condition Check

**Step 1:** After the predefined maximum evaluation number, the optimization process is terminated.

This algorithm is used in many papers and its efficiency has been proven in structural size optimization. ECBO algorithm being capable of maintaining a proper balance between the diversification and the intensification inclinations is utilized in all stages of cascade process. Local search algorithms are proper for the last stages of the proposed method, but these cannot have suitable global search in the first stages. Moreover, the ECBO algorithm has recently been used for structural optimization.

Although the idea of cascade procedure has been introduced to be used for multiple optimizers, however, in this chapter another kind of cascade procedure is implemented in which, instead of using multiple optimizers, multiple DVCs are

utilized. The potential of using multiple optimizers has already been shown. For this reason single optimizer is used in all stages of the cascade procedure in order to examine the efficiency of proposed method independent of the utilized optimizers.

## 19.5 Numerical Examples

Three large-scale steel space frames are studied to investigate the efficiency of the cascade-enhanced colliding bodies optimization in weight optimum design. These examples consist of a 1860-member steel space frame with 72 member groups (design variables), a 3590-member steel space frame with 124 member groups, and finally a 3328-member steel space frame with 186 member groups.

The structural members are sized using wide-flange W-sections, such that columns are selected from the complete set of 297 W-sections, beams are selected from a set of 171 economical W-sections chosen based on area and inertia properties, and finally bracings are selected from a set of 147 economical W-sections chosen based on area and radii of gyration properties. The material properties of the steel are assumed as follows: modulus of elasticity  $E = 29,000$  ksi (203893.6 MPa) and yield stress  $F_y = 36$  ksi (253.1 MPa). Design constraints and objective function are considered as defined in Sect. 19.2.

The objective function utilized is as follows:

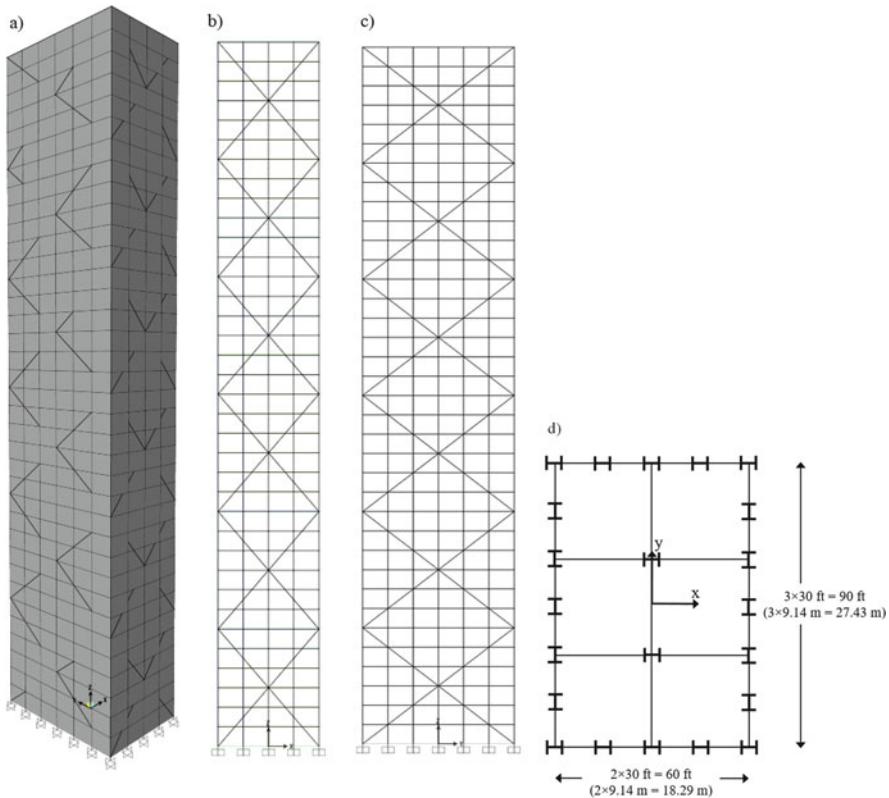
$$fit(x) = W(x) \times (1 + v) \quad (19.21)$$

$$v = \sum_{i=1}^{nc} \max[0, g_i(x)] \quad (19.22)$$

where  $W$  is the weight of the structure obtained using Eq. (19.2) and  $g$  contains all the constraints mentioned in Sect. 19.2. In colliding bodies optimization algorithm, the value of the objective function for each candidate is obtained using the above equations, and these values are incorporated in calculating the mass of the colliding bodies (Eq. (19.11)).

For all examples, **pro** parameter in ECBO is set to 0.3 and a population of 40 CBs is implemented. Colliding memory (CM) size for non-cascade ECBO and cascade ECBO is set to 2 and 3, respectively, and three best attained designs are moved to the next stage in the cascade method. All examples started with a random population. Due to randomness property of the algorithms, five cascade and five non-cascade runs are performed for each example, and the best results, means, and standard deviations are provided.

The algorithms are coded in MATLAB (The MathWorks [25]), and the structures are analyzed using OpenSees (Mazzoni et al. [26]). These two softwares are linked together and static linear finite element analysis is performed in OpenSees (Mazzoni et al. [26]). Then, structural response is received by MATLAB (The MathWorks [25]) to check the constraints and evaluate the objective function values.



**Fig. 19.6** Schematic of a 1860-member braced space steel frame [1]. (a) Three-dimensional view, (b) front view, (c) side view, and (d) plan view

### 19.5.1 A 1860-Member Steel Space Frame

The first design example considered in this section is a 36-story braced space steel frame consisting of 814 joints and 1860 members. This structure has also been investigated by Saka and Hasançebi [27]. The side, plan, and three-dimensional views of the frame are shown in Fig. 19.6. 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 1860 frame members are collected in 72 different member groups, considering the symmetry of the structure and practical fabrication requirements. That is, the columns in each 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 inner columns, outer columns, inner beams, and outer beams. Bracing members on each facade are designed as three-

**Table 19.1** Number of design variables and their allocation to structural parts for each DVC of the 1860-member braced steel space frame

DVC	Member type	Number of member groups	Total	Remarks
C <sub>0</sub>	Columns	$3 \times 12 = 36$	72	Corner columns, inner columns, and outer columns grouped every three stories
	Beams	$2 \times 12 = 24$		Outer beams and inner beams grouped every three stories
	Bracings	$2 \times 6 = 12$		Bracings in x direction and y direction grouped every six stories
C <sub>1</sub>	Columns	$3 \times 6 = 18$	34	Corner columns, inner columns and outer columns grouped every six stories
	Beams	$2 \times 6 = 12$		Outer beams and inner beams grouped every six stories
	Bracings	$2 \times 2 = 4$		Bracings in x direction and y direction grouped every 18 stories
C <sub>2</sub>	Columns	$3 \times 2 = 6$	10	Corner columns, inner columns, and outer columns grouped every 18 stories
	Beams	$2 \times 1 = 2$		Outer beams and inner beams grouped in all stories
	Bracings	$2 \times 1 = 2$		Bracings in x direction and y direction grouped in all stories

story deep members, and two bracing groups are specified in every six stories. Table 19.1 shows the number of design variables and their allocation in structural parts for each of the three DVCs defined for cascade method.

The 1860-member braced space steel frame is subjected to two loading conditions of combined gravity and wind forces. These forces are computed as per ASCE 7-05 based on the following design values: a design dead load of  $2.88 \text{ kN/m}^2$  ( $60.13 \text{ lb/ft}^2$ ), a design live load of  $2.39 \text{ kN/m}^2$  ( $50 \text{ lb/ft}^2$ ), a ground snow load of  $1.20 \text{ kN/m}^2$  ( $25 \text{ lb/ft}^2$ ), and a basic wind speed of  $55.21 \text{ m/s}$  (123.5 mph). Lateral (wind) loads acting at each floor level on windward and leeward faces of the frame are tabulated in Table 19.2, and the gravity loading on the beams of roof and floors is given in Table 19.3. In the first loading condition, gravity loads are applied together with wind loads acting along x-axis ( $1.0 \text{ GL} + 1.0\text{WL-x}$ ), whereas in the second one, they are applied with wind loads acting along y-axis ( $1.0 \text{ GL} + 1.0\text{WL-y}$ ).

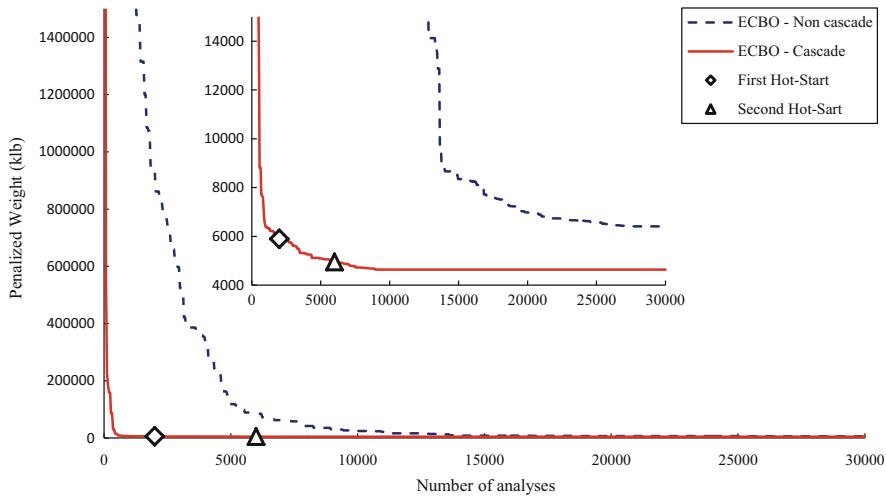
The 1860-member braced space steel frame is designed separately by using both ECBO and cascade ECBO search method. The maximum number of analyses was 30,000. The design history of both runs is shown in Fig. 19.7. The minimum weight for the frame is determined as 4637.45 klb by the cascade ECBO method, while non-cascade ECBO arrived at 6406.53 klb which is 38.1 % heavier. The mean and standard deviation of the independent runs for cascade optimization procedure are 4647.10 klb and 18.37 klb, respectively, whereas the values of these parameters for non-cascade procedure are 6420.53 klb and 25.54 klb, respectively. Figure 19.8 shows the ratio of story drift over the maximum allowable drift of the story for final

**Table 19.2** Wind loading on a 1860-member braced space steel frame

Floor	Windward kN/m (lb/ft)	Leeward kN/m (lb/ft)
1	2.05 (140.64)	3.57 (244.70)
2	2.50 (171.44)	3.57 (244.70)
3	2.81 (192.49)	3.57 (244.70)
4	3.05 (208.98)	3.57 (244.70)
5	3.25 (222.74)	3.57 (244.70)
6	3.42 (234.65)	3.57 (244.70)
7	3.58 (245.22)	3.57 (244.70)
8	3.72 (254.75)	3.57 (244.70)
9	3.85 (263.47)	3.57 (244.70)
10	3.96 (271.52)	3.57 (244.70)
11	4.07 (279.02)	3.57 (244.70)
12	4.18 (286.04)	3.57 (244.70)
13	4.27 (292.66)	3.57 (244.70)
14	4.36 (298.92)	3.57 (244.70)
15	4.45 (304.87)	3.57 (244.70)
16	4.53 (310.55)	3.57 (244.70)
17	4.61 (315.97)	3.57 (244.70)
18	4.69 (321.18)	3.57 (244.70)
19	4.76 (326.18)	3.57 (244.70)
20	4.83 (330.99)	3.57 (244.70)
21	4.90 (335.64)	3.57 (244.70)
22	4.97 (340.13)	3.57 (244.70)
23	5.03 (344.48)	3.57 (244.70)
24	5.09 (348.69)	3.57 (244.70)
25	5.15 (352.78)	3.57 (244.70)
26	5.21 (356.76)	3.57 (244.70)
27	5.27 (360.62)	3.57 (244.70)
28	5.32 (364.39)	3.57 (244.70)
29	5.37 (368.06)	3.57 (244.70)
30	5.43 (371.65)	3.57 (244.70)
31	5.48 (375.14)	3.57 (244.70)
32	5.53 (378.56)	3.57 (244.70)
33	5.58 (381.90)	3.57 (244.70)
34	5.62 (385.18)	3.57 (244.70)
35	5.67 (388.38)	3.57 (244.70)
36	2.86 (195.76)	1.79 (122.35)

**Table 19.3** Gravity loading on the beams of 1860-member braced steel space frame

Beam type	Uniformly distributed load, kN/m (lb/ft)		
	Dead load	Live load	Snow load
Roof beams	22.44 (1536.66)	N/A	5.88 (402.50)
Floor beams	22.44 (1536.66)	18.66 (1277.78)	N/A



**Fig. 19.7** Convergence curves for the 1860-member steel space frame

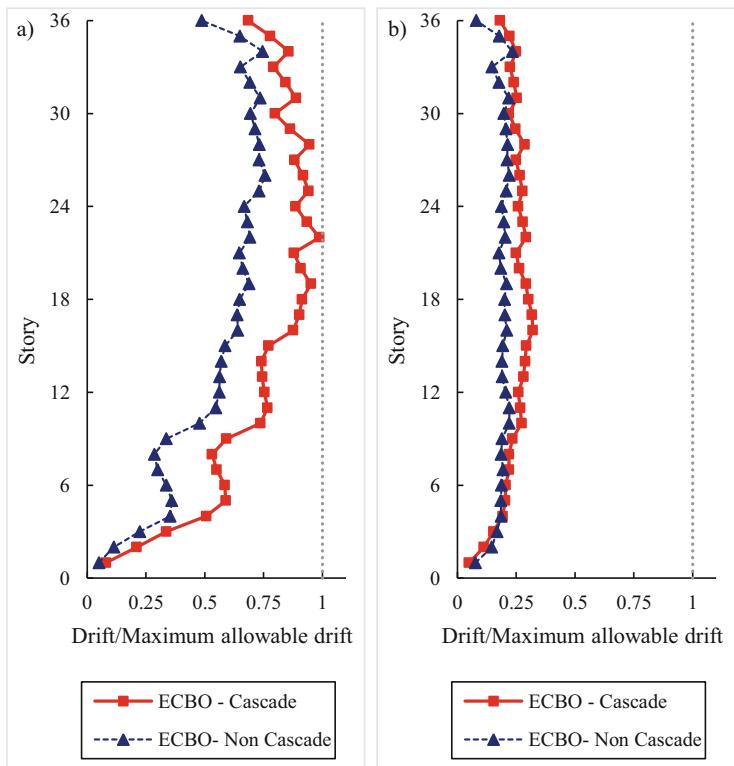
design of two mentioned methods. In Fig. 19.9 maximum values of stress ratio for members of each design group for ECBO and cascade ECBO are also compared.

Multi-DVC cascade procedure provides some other options for the designer. The designer can choose more uniform structure by spending high cost for materials (that may cause to less final cost). In the first example, attained optimal design with  $C_1$  DVC which contains 34 types of frame sections is only 6.66 % heavier than finally attained design which contains 72 types of frame section.

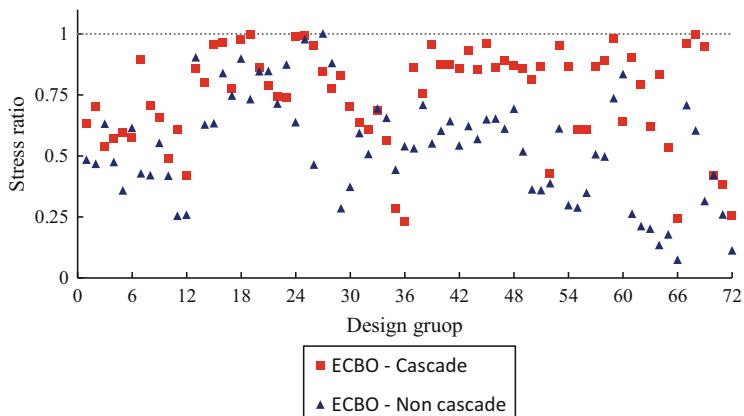
### 19.5.2 A 3590-Member Steel Space Frame

The second design example is a braced space steel frame consisting of 1540 joints and 3590 members that are to be built in three adjacent blocks with 30, 18, and 12 stories. The three-dimensional and plan views of the frame at different story levels are shown in Fig. 19.10. 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 as well as on the adjacent sides of the blocks. The diagonal members are also known to participate in transmitting gravity forces. The 3590 frame members are collected in 124 different member groups altogether. Table 19.4 shows the member groups for this example and two other built DVCs.

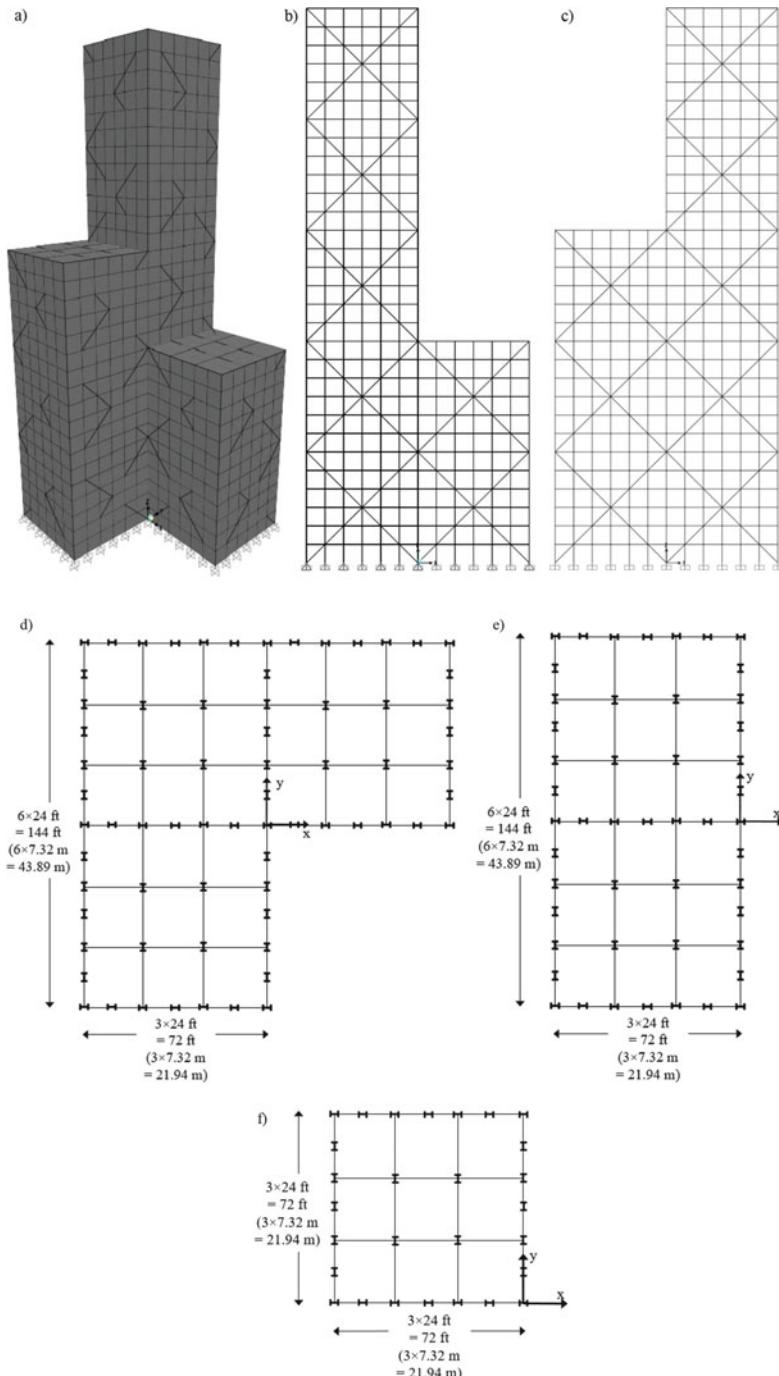
The combined stress, stability, and geometric constraints are imposed according to the defined provisions in Sect. 19.2. Similar to the previous example, the 1860-member braced space steel frame is subjected to two loading conditions of combined wind and gravity forces which are tabulated in Tables 19.5 and 19.6,



**Fig. 19.8** Drift divided by the maximum allowable drift ratio in final design of the 1860-member steel space frame. (a) X direction, (b) y direction



**Fig. 19.9** Maximum stress ratio of members in each design group in final design of the 1860-member steel space frame



**Fig. 19.10** Schematic of a 3590-member braced space steel frame [1]. (a) Three-dimensional view, (b) front view, (c) side view, (d) stories 1–12 (Section 1), (e) stories 13–18 (Section 2), and (f) stories 19–30 (Section 3)

**Table 19.4** Number of design variables and their allocation to structural parts for each DVC of the 3590-member braced steel space frame

DVC	Member type	Number of member groups	Total	Remarks
C <sub>0</sub>	Columns	(3 × 15) + 9 = 54	124	Corner columns, inner columns, and outer columns grouped every two stories. Inner columns in braced frames are collected in separate groups
	Beams	2 × 30 = 60		Outer beams and inner beams are grouped every story
	Bracings	1 × 10 = 10		Bracings are grouped every three stories
C <sub>1</sub>	Columns	(3 × 7) + 4 = 25	60	Corner columns, inner columns, and outer columns in all stories and inner columns in braced frames in Sections 1 and 2 are grouped every four stories. Six above stories are collected in same groups
	Beams	2 × 15 = 30		Outer beams and inner beams grouped every two stories
	Bracings	1 × 5 = 5		Bracings grouped every six stories
C <sub>2</sub>	Columns	(3 × 2) + 1 = 7	14	Corner columns, inner columns, and outer columns grouped in 16 below stories and 14 above stories. All inner columns in braced frames are collected in one group
	Beams	2 × 3 = 6		Outer beams and inner beams are grouped in every ten stories
	Bracings	1		All bracings are collected in one group

respectively. In the first loading condition, gravity loads are applied together with wind loads acting along x-axis (1.0 GL + 1.0WL-x), whereas in the second one, these are applied with wind loads acting along y-axis (1.0 GL +1.0WL-y).

An optimum design weight of 6496.41 klb has been reached for the frame with cascade ECBO. Figure 19.11 shows the design history graph for 50,000 analyses obtained for this example. Non-cascade ECBO attained optimum design weight of 9937.94 klb that is 52.98 % heavier than the cascade method design. The mean and standard deviation of the independent runs for cascade optimization procedure are 6505.90 klb and 21.80 klb, respectively, whereas the values of these parameters for non-cascade procedure are 9955.93 klb and 35.67 klb, respectively. Figure 19.12 shows the ratio of story drift over the maximum allowable drift of the story for the final design of two mentioned methods. Maximum values of stress ratios for members of each design group for ECBO and cascade ECBO are also compared in Fig. 19.13.

**Table 19.5** Wind loading on the 3590-member braced space steel frame

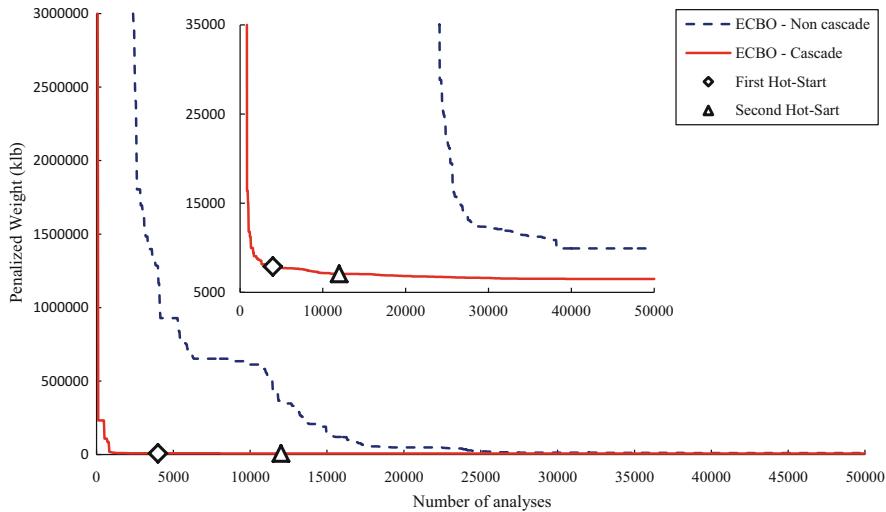
Floor	Windward kN/m (lb/ft)	Leeward kN/m (lb/ft)
1	1.60 (109.41)	2.47 (169.56)
2	1.83 (125.14)	2.47 (169.56)
3	2.05 (140.51)	2.47 (169.56)
4	2.23 (152.55)	2.47 (169.56)
5	2.37 (162.59)	2.47 (169.56)
6	2.50 (171.28)	2.47 (169.56)
7	2.61 (179.00)	2.47 (169.56)
8	2.71 (185.96)	2.47 (169.56)
9	2.81 (192.32)	2.47 (169.56)
10	2.89 (198.20)	2.47 (169.56)
11	2.97 (203.67)	2.47 (169.56)
12	3.05 (208.80)	2.47 (169.56)
13	3.12 (213.63)	2.47 (169.56)
14	3.18 (218.20)	2.47 (169.56)
15	3.25 (222.54)	2.47 (169.56)
16	3.31 (226.68)	2.47 (169.56)
17	3.37 (230.64)	2.47 (169.56)
18	3.42 (234.44)	2.47 (169.56)
19	3.47 (238.09)	2.47 (169.56)
20	3.53 (241.61)	2.47 (169.56)
21	3.56 (245.00)	2.47 (169.56)
22	3.62 (248.28)	2.47 (169.56)
23	3.67 (251.45)	2.47 (169.56)
24	3.71 (254.53)	2.47 (169.56)
25	3.76 (257.51)	2.47 (169.56)
26	3.80 (260.41)	2.47 (169.56)
27	3.84 (263.24)	2.47 (169.56)
28	3.88 (265.99)	2.47 (169.56)
29	3.92 (268.67)	2.47 (169.56)
30	1.98 (135.64)	84.78

**Table 19.6** Gravity loading on the beams of the 3590-member braced steel space frame

Beam type	Uniformly distributed load, kN/m (lb/ft)		
	Dead load	Live load	Snow load
Roof beams	10.53 (721.56)	N/A	4.38 (300.00)
Floor beams	22.42 (1536.66)	8.76 (600.00)	N/A

### 19.5.3 A 3328-Member Steel Space Frame

The last design example is a 3328-member irregular moment resisting steel space frame structure with setbacks and cross bracings of Fig. 19.14 which consist of



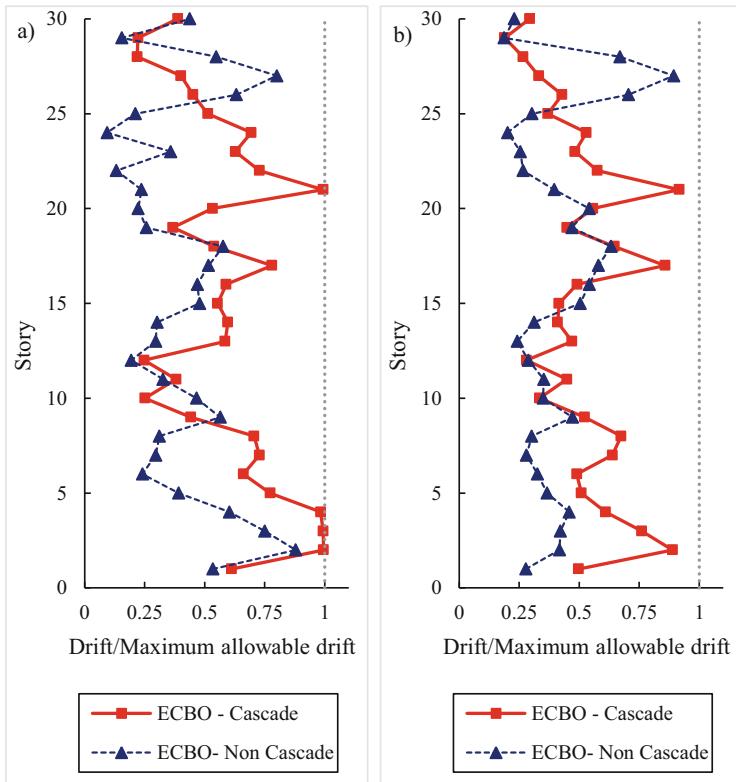
**Fig. 19.11** Convergence curves for the 3590-member steel space frame

1384 joints. The space frame is clamped to the ground and is subjected to vertical dead and live loads (Table 19.7) and horizontal wind loads (Table 19.8).

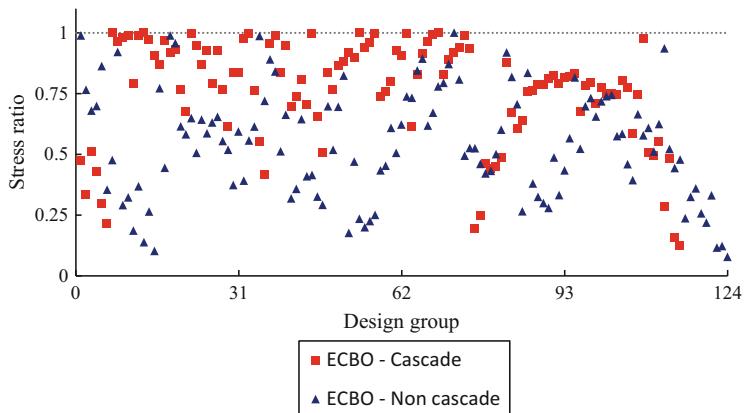
The frame is divided in the vertical direction into three 12-story sections, as shown in Fig. 19.14. Similar structure has been considered by Sarma and Adeli [28]. For this example also two-load combination is defined. In the first loading condition, gravity loads are applied together with wind loads acting along x-axis (1.0 GL + 1.0WL-x), whereas in the second one, they are applied with wind loads acting along y-axis (1.0 GL + 1.0WL-y). Table 19.9 tabulates member groups for the sizing optimization procedure and two other defined DVCs for cascade ECBO method. The following member types of the space frame are distinguished:

- Four types of columns: corner columns, outer columns, inner columns in unbraced frames, and inner columns in braced frames
- Three types of beams for the two lower Sections 1 and 2: outer beams, inner beams in unbraced frames, and inner beams in braced frames
- Two types of beams for the upper Section 3: outer and inner beams
- Two groups of bracings in the longitudinal and transverse directions

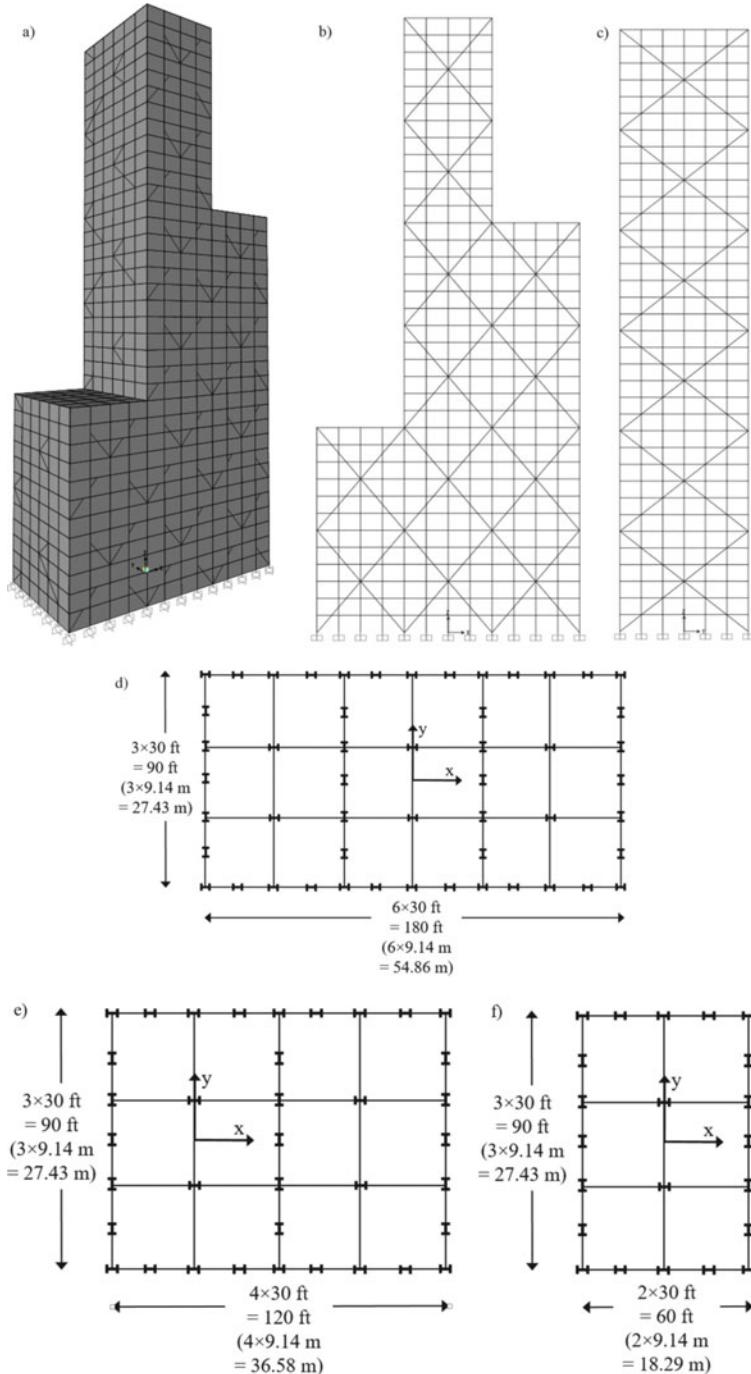
Figure 19.15 shows the convergence histories obtained for the proposed cascade ECBO and non-cascade ECBO optimization methods. In this example, 70,000 analyses were executed for cascade and non-cascade ECBO. Non-cascade ECBO attained optimum design weight of 12724.97 klb that is 28.31 % heavier than the cascade method design which attained optimum design weight of 9917.51 klb. The mean and standard deviation of the independent runs for cascade optimization procedure are 9948.72 klb and 43.03 klb, respectively, whereas the values of these parameters for non-cascade procedure are 12772.24 klb and 54.29 klb, respectively. Figure 19.16 shows the ratio of story drift versus the maximum



**Fig. 19.12** Drift divided by the maximum allowable drift ratio in final design of the 3590-member steel space frame. (a) x-direction, (b) y-direction



**Fig. 19.13** Maximum stress ratio of members in each design group in final design of the 1860-member steel space frame



**Fig. 19.14** Schematic of a 3328-member braced space steel frame [1]. (a) Three-dimensional view, (b) front view, (c) side view, (d) stories 1–12 (Section 1), (e) stories 13–24 (Section 2), and (f) stories 25–36 (Section 3)

**Table 19.7** Wind loading on the 3228-member braced space steel frame

Floor	Windward kN/m (lb/ft)	Leeward kN/m (lb/ft)
1	1.60 (109.41)	2.55 (174.91)
2	1.81 (124.18)	2.55 (174.91)
3	2.03 (139.43)	2.55 (174.91)
4	2.21 (151.37)	2.55 (174.91)
5	2.35 (161.34)	2.55 (174.91)
6	2.48 (169.96)	2.55 (174.91)
7	2.59 (177.62)	2.55 (174.91)
8	2.69 (184.52)	2.55 (174.91)
9	2.78 (190.84)	2.55 (174.91)
10	2.87 (196.67)	2.55 (174.91)
11	2.95 (202.10)	2.55 (174.91)
12	3.02 (207.19)	2.55 (174.91)
13	3.09 (211.98)	2.55 (174.91)
14	3.16 (216.52)	2.55 (174.91)
15	3.22 (220.83)	2.55 (174.91)
16	3.28 (224.94)	2.55 (174.91)
17	3.34 (228.87)	2.55 (174.91)
18	3.39 (232.64)	2.55 (174.91)
19	3.45 (236.26)	2.55 (174.91)
20	3.50 (239.75)	2.55 (174.91)
21	3.55 (243.11)	2.55 (174.91)
22	3.60 (246.36)	2.55 (174.91)
23	3.64 (249.51)	2.55 (174.91)
24	3.69 (252.57)	2.55 (174.91)
25	3.73 (255.53)	2.55 (174.91)
26	3.77 (258.41)	2.55 (174.91)
27	3.81 (261.21)	2.55 (174.91)
28	3.85 (263.94)	2.55 (174.91)
29	3.89 (266.6)	2.55 (174.91)
30	3.93 (269.19)	2.55 (174.91)
31	3.97 (271.73)	2.55 (174.91)
32	4.00 (274.2)	2.55 (174.91)
33	4.04 (276.62)	2.55 (174.91)
34	4.07 (278.99)	2.55 (174.91)
35	4.11 (281.31)	2.55 (174.91)
36	2.07 (141.80)	87.46)

**Table 19.8** Gravity loading on the beams of the 3228-member braced steel space frame

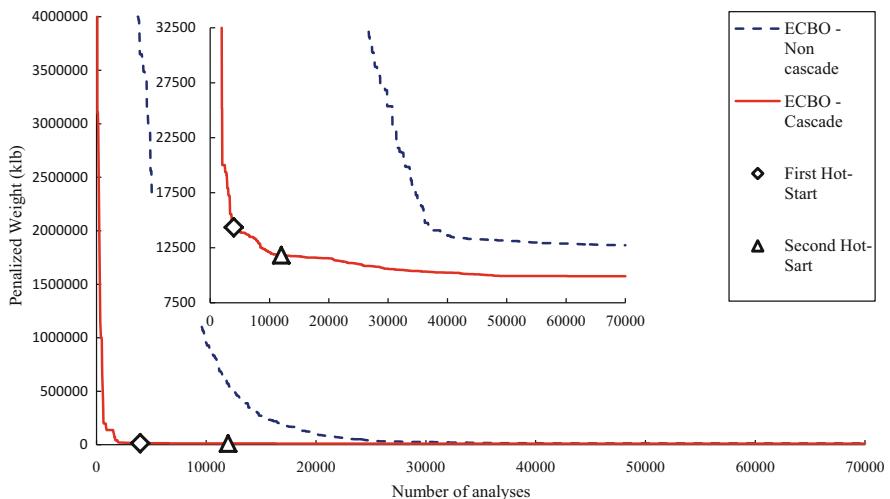
Beam type	Uniformly distributed load, kN/m (lb/ft)		
	Dead load	Live load	Snow load
Roof beams	11.12 (761.74)	N/A	3.65 (250.00)
Floor beams	13.26 (908.40)	8.17 (560.00)	N/A

**Table 19.9** Number of design variables and their allocation to structural parts for each DVC of the 3228-member braced steel space frame

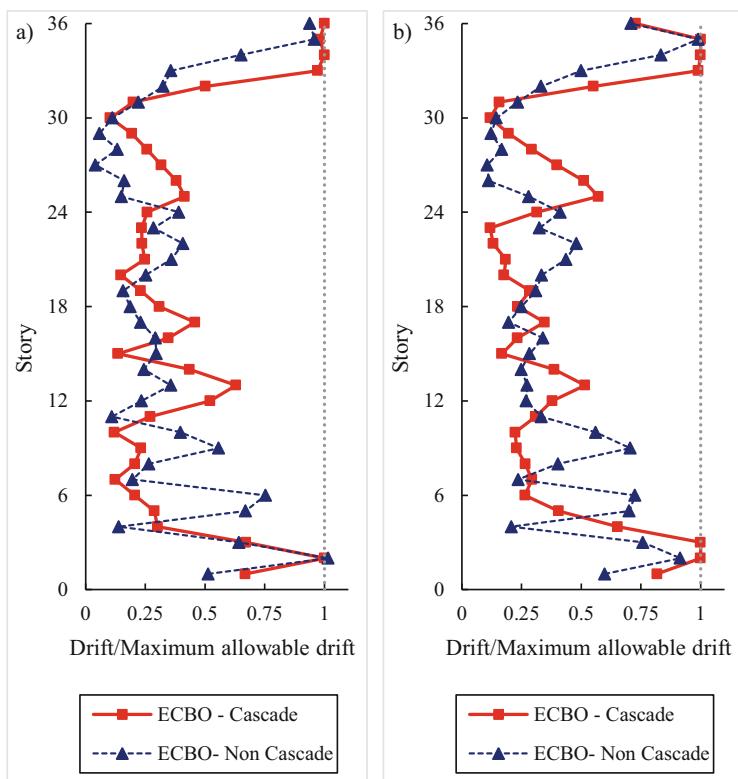
DVC	Member type	Number of member groups				Total	Remarks
		Section 1	Section 2	Section 3	All sections		
C <sub>0</sub>	Columns	4 × 6 = 24	4 × 6 = 24	3 × 6 = 18	66	186	Grouped every two stories
	Beams	3 × 12 = 36	3 × 12 = 36	2 × 12 = 24	96		Grouped every story
	Bracings	2 × 4 = 36	2 × 4 = 8	2 × 4 = 8	24		Grouped every three stories
C <sub>1</sub>	Columns	4 × 3 = 12	4 × 3 = 12	3 × 3 = 9	33	81	Grouped every four stories
	Beams	3 × 3 = 9	3 × 3 = 9	2 × 3 = 6	24		Grouped every four stories
	Bracings	2 × 4 = 8	2 × 4 = 8	2 × 4 = 8	24		Grouped every three stories
C <sub>2</sub>	Columns	4 × 1 = 4	4 × 1 = 4	3 × 1 = 3	11	25	Grouped every 12 stories
	Beams	3 × 1 = 3	3 × 1 = 3	2 × 1 = 2	8		Grouped every 12 stories
	Bracings	2 × 1 = 2	2 × 1 = 2	2 × 1 = 2	6		Grouped every 12 stories

allowable drift of the story for final design of two mentioned methods for this example. Maximum values of stress ratios for the members of each design group for ECBO and cascade ECBO, for this problem, are also compared in Fig. 19.17.

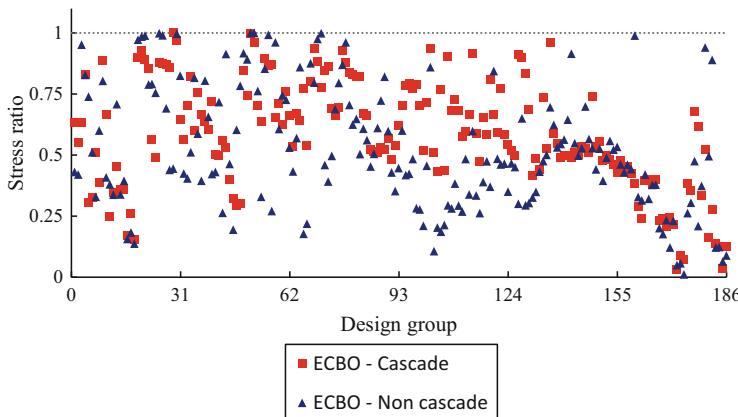
Predefining of the most critical constraint in optimal design of these problems is almost impossible. In the first example, which is a high-rise regular space frame, drifts in the middle stories are more critical (as expected), but in the second problem, the stress ratios are more critical than other constraints, and in the last example, no specific trends are observed.



**Fig. 19.15** Convergence curves for the 3228-member steel space frame



**Fig. 19.16** Drift divided by the maximum allowable drift ratio in final design of the 3590-member steel space frame. (a) X direction, (b) y direction



**Fig. 19.17** Maximum stress ratio of members in each design group in final design of the 3228-member steel space

## 19.6 Concluding Remarks

Large-scale steel space frame optimization with design code constraints is a difficult, complex, highly nonlinear, and non-convex problem. In this chapter the performance of multi-DVC cascade and non-cascade ECBO is compared through three large-scale steel space frames. In all examples, in addition to multi-DVC cascade procedure that resulted in better optimized design, the required number of analyses for achieving the best design of non-cascade procedure by the proposed method is decreased. This issue is important due to large computational cost of structural analyses in large-scale steel space frames.

Although ECBO algorithm has shown its efficiency in finding near-optimum solution, however, in problems of this chapter, it did not provide good results. Large number of design variables and nonlinear constraints caused the optimizer to be trapped in local optima. In this condition, optimizer could not find near-optimum solution. Because, finding a variable that its change leads to improvement of the solution, among large number of variables, is a big challenge, and optimizer can hardly cope with this challenge.

## References

- Kaveh A, Bolandgerami A (2016) Optimal design of large scale space steel frames using cascade enhanced colliding body optimization. *Struct Multidiscip Optim*. [10.1007/s00158-016-1494-2](https://doi.org/10.1007/s00158-016-1494-2)
- Schulz V, Book HG (1997) Partially reduced sqp methods for large-scale nonlinear optimization problems. *Nonlinear Anal Theory Methods Appl* 30:4723–4734
- Dreyer T, Maar B, Schulz V (2000) Multigrid optimization in applications. *J Comput Appl Math* 120:67–84

4. Wang Q, Arora JS (2007) Optimization of large-scale truss structures using sparse SAND formulations. *Int J Numer Methods Eng* 69:390–407
5. Yang Z, Tang K, Yao X (2008) Large scale evolutionary optimization using cooperative coevolution. *Inf Sci* 178:2985–2999
6. Hsieh S-T, Sun T-Y, Liu C-C, Tsai S-J (2008) Solving large scale global optimization using improved particle swarm optimizer. In: Evolutionary computation, 2008. CEC 2008, IEEE World Congress on Computational Intelligence, pp 1777–1784
7. Fister I, Fister Jr I, Zumer JB (2012) Memetic artificial bee colony algorithm for large-scale global optimization. In: IEEE Congress on Evolutionary Computation (CEC), pp 1–8
8. Singh D, Agrawal S (2016) Self organizing migrating algorithm with quadratic interpolation for solving large scale global optimization problems. *Appl Soft Comput* 38:1040–1048
9. Kaveh A, Talatahari S (2011) Optimization of large-scale truss structures using charged system search. *Int J Optim Civ Eng* 1(1):15–28
10. Lagaros ND (2013) A general purpose real-world structural design optimization computing platform. *Struct Multidiscip Optim* 49:1047–1066
11. Talatahari S, Kaveh A (2015) Improved Bat Algorithm for Optimum Design of Large-Scale Truss Structures. *Int J Optim Civil Eng* 5:241–254
12. Aydoğdu İ, Akin A, Saka MP (2016) Design optimization of real world steel space frames using artificial bee colony algorithm with Levy flight distribution. *Adv Eng Softw* 92:1–14
13. Papadrakakis M, Lagaros ND, Tsompanakis Y (1998) Structural optimization using evolution strategies and neural networks. *Comput Methods Appl Mech Eng* 156:309–333
14. Sobiesczanski-Sobieski J, James BB, Dovi AR (1985) Structural optimization by multilevel decomposition. *AIAA J* 23:1775–1782
15. Sobiesczanski-Sobieski J, James BB, Riley MF (1987) Structural sizing by generalized, multilevel optimization. *AIAA J* 25:139–145
16. Charmpis DC, Lagaros ND, Papadrakakis M (2005) Multi-database exploration of large design spaces in the framework of cascade evolutionary structural sizing optimization. *Comput Methods Appl Mech Eng* 194:3315–3330
17. Kaveh A, Ilchi Ghazaan M (2015) Optimal design of dome truss structures with dynamic frequency constraints. *Struct Multidiscip Optim.* [10.1007/s00158-015-1357-2](https://doi.org/10.1007/s00158-015-1357-2)
18. Kaveh A, Ilchi Ghazaan M (2014) Enhanced colliding bodies optimization for design problems with continuous and discrete variables. *Adv Eng Softw* 77:66–75
19. Kaveh A, Mahdavi VR (2014) Colliding Bodies Optimization method for optimum design of truss structures with continuous variables. *Adv Eng Softw* 70:1–12
20. American Institute of Steel Construction (1989) Manual of steel construction: allowable stress design. American Institute of Steel Construction
21. Dumontel P (1992) Simple equations for effective length factors. *Eng J AISC* 29(3):111–115
22. Hellesland J (1996) Improved frame stability analysis with effective lengths. *J Struct Eng* 122 (11):1275–1283
23. Specification A (2005) Specification for structural steel buildings. ANSI/AISC
24. Patnik SN, Coroneos RM, Hopkins DA (1997) A cascade optimization strategy for solution of difficult design problems. *Int J Numer Methods Eng* 40:2257–2266
25. The MathWorks (2013) MATLAB, Natick, Massachusetts, USA
26. Mazzoni S, McKenna F, Scott M (2006) OpenSees command language manual
27. Saka MP, Hasancebi O (2009) Adaptive harmony search algorithm for design code optimization of steel structures. Springer, Berlin
28. Sarma KC, Adeli H (2002) Life-cycle cost optimization of steel structures. *Int J Numer Methods Eng* 55:1451–1462

# Chapter 20

## Multi-Objective Optimization of Truss Structures

### 20.1 Introduction

In this chapter a multi-objective optimization (MOP) is presented that uses the main concepts of charged system search algorithm (Kaveh and Massoudi [1]).

In the last two decades, many efficient mono-objective optimization algorithms are developed [2–7]. These algorithms search through possible feasible solutions and ultimately identify the best results. Multi-objective optimization techniques play an important role in engineering design, resource optimization, and many other fields. Their main purpose is to find a set of best solutions from which a designer or a decision maker can choose a solution to derive maximum benefit from the available resources. Various objectives of a multi-objective optimization problem often conflict and/or compete with one another. In multi-criteria decision making (DM), no single solution can be termed as the optimum solution to the multiple conflicting objectives, as a multi-objective optimization problem is amenable to a number of trade-off optimal solutions. For this purpose, multi-objective optimization generates a Pareto front, which is a set of non-dominated solutions for problems with more than one objective. The major goal of a multi-objective optimization algorithm is to generate a well-distributed true Pareto optimal front or surface.

Over the past decade, a number of multi-objective meta-heuristic algorithms (MOEAs) have been developed, such as non-dominated sorting genetic algorithm (NSGA)-II [8], strength Pareto evolutionary algorithm 2 (SPEA2) [9], Pareto archive evolution strategy (PAES) [10], multi-objective particle swarm optimization (MOPSO) [11], and hybrid multi-objective optimization comprising CSS and PSO [12].

In this chapter a new multi-objective optimization approach, based purely on charged system search (CSS) algorithm, is introduced. The CSS is a population-based metaheuristic optimization algorithm which has been proposed recently by Kaveh and Talatahari [5, 13]. In the CSS each solution candidate is considered as a

charged sphere called a charged particle (CP). The electrical load of a CP is determined considering its fitness. Each CP exerts an electric force on all the others according to Coulomb's and Gauss's laws from electrostatics. Then, the new positions of all the CPs are calculated utilizing Newtonian mechanics, based on the acceleration produced by the electric force, the previous velocity, and the previous position of each CP. Many different structural optimization problems have been successfully solved by CSS [13].

In the present work, after a brief description of multi-objective optimization (MOP), the main concepts of charged system search algorithm are provided. For better understanding of the MOPs, readers can refer to [14]. Then, the multi-objective charged system search algorithm is presented. A simple multi-criteria decision-making process is also presented. Numerical examples are prepared to show the efficiency and accuracy of the proposed method. Finally, the concluding remarks are provided.

## 20.2 Multi-Objective Optimization Concepts

*Definition 1 Multi-objective optimization problem.* A multi-objective optimization problem can be defined as:

$$MOP = \left\{ \begin{array}{l} \min F(x) = (f_1(x), f_2(x), \dots, f_n(x)) \\ \text{s.c. } x \in S \end{array} \right. \quad (20.1)$$

where  $n \geq 2$  is the number of objectives,  $x = (x_1, x_2, \dots, x_k)$  is the vector representing the decision variables,  $S$  represents the set of feasible solutions associated with equality and inequality constraints and explicit bounds, and  $F(x) = (f_1(x), f_2(x), \dots, f_n(x))$  is the vector of objectives to be optimized.

*Definition 2 Pareto dominance.* An objective vector  $u = (u_1, u_2, \dots, u_n)$  is said to dominate  $v = (v_1, v_2, \dots, v_n)$ , denoted by  $u \prec v$ , if and only if no component of  $v$  is smaller than the corresponding component of  $u$  and at least one component of  $u$  is strictly smaller, that is,

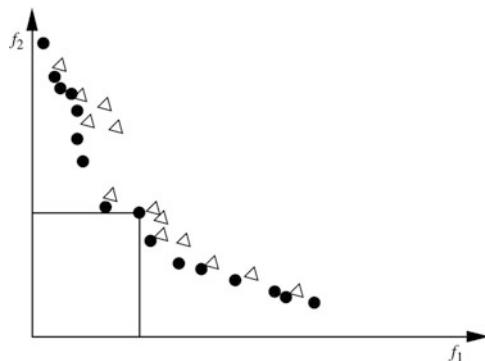
$$\forall i \in \{1, \dots, n\} : u_i \leq v_i \wedge \exists i \in \{1, \dots, n\} : u_i < v_i$$

*Definition 3 Pareto optimality.* A solution  $x^* \in S$  is Pareto optimal if for every  $x \in S$ ,  $F(x)$  does not dominate  $F(x^*)$ , that is,  $F(x) \not\prec F(x^*)$ .

Graphically, a solution  $x^*$  is Pareto optimal if there is no other solution  $x$  such that the point  $F(x)$  is in the dominance cone of  $F(x^*)$  that is the box defined by  $F(x)$ , with its projections on the axes and origin, Fig. 20.1.

*Definition 4 Pareto optimal set.* For a given  $MOP(F, S)$ , the Pareto optimal set is defined as  $P^* = \{x \in S / \nexists x' \in S, F(x') \prec F(x)\}$ .

**Fig. 20.1** Filled circles for Pareto solution and open triangles for dominated solution



**Definition 5 Pareto front.** For a given  $MOP(F, S)$  and its Pareto optimal set, the Pareto front is defined as  $PF^* = \{F(x), x \in P^*\}$ .

The Pareto front is the image of the Pareto optimal set in the objective space. Obtaining the Pareto front of a MOP is the main goal of a multi-objective optimization. The Pareto front should have two desirable properties consisting of good convergence and diversity.

## 20.3 Charged System Search Algorithm

The charged system search contains a number of charged particles (CPs), where each CP is treated as a charged sphere and can exert an electric force on the others. The magnitude of this force for a CP located inside the sphere is proportional to the separation distance between the CPs and for a CP located outside the sphere is inversely proportional to the square of the separation distance between the particles. The resultant forces persuade the CPs to move toward new locations according to the motion laws of Newtonian mechanics. In the new positions, the magnitude and the direction of the forces are reformed, and this successive action is repeated until a terminating condition is satisfied. The pseudo code for the CSS algorithm is summarized as follows:

### Level 1: Initialization

- **Step 1: Initialization.** The magnitude of charge for each CP is defined as:

$$q_i = \frac{fit(i) - fitworst}{fibest - fitworst} \quad i = 1, 2, \dots, N \quad (20.2)$$

where  $fibest$  and  $fitworst$  are the best and the worst fitness of all the particles, respectively,  $fit(i)$  represents 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:

$$r_{ij} = \frac{\|X_i - X_j\|}{\|(X_i + X_j)/2 - X_{best}\| + \epsilon} \quad (20.3)$$

where  $\mathbf{X}_i$  and  $\mathbf{X}_j$  are the positions of the  $i$ th and  $j$ th CPs, respectively,  $\mathbf{X}_{best}$  is the position of the best current CP, and  $\epsilon$  is a small positive number. The initial positions of CPs are determined randomly.

- **Step 2: CP ranking.** Considering the values of the fitness function, sort the CPs in an increasing order.
- **Step 3: CM creation.** Store a number of the first CPs and the values of their corresponding fitness functions in the charged memory (CM).

### Level 2: Search

- **Step 1: Moving probability determination.** Determine the probability of moving each CP toward the others using the following probability function:

$$p_{ij} = \begin{cases} 1 & \frac{fit(i) - fit_{best}}{fit(j) - fit(i)} > rand \vee fit(j) > fit(i) \\ 0 & otherwise \end{cases} \quad (20.4)$$

- **Step 2: Force determination.** Calculate the resultant force vector for each CP as

$$F_j = q_j \sum_{i, i \neq j} \left( \frac{q_i}{a^3 r_{ij}} \cdot i_1 + \frac{q_i}{r_{ij}^2} \cdot i_2 \right) ar_{ij} p_{ij} (X_i - X_j), \quad \begin{cases} j = 1, 2, \dots, N \\ i_1 = 1, i_2 = 0 \Leftrightarrow r_{ij} < a \\ i_1 = 0, i_2 = 1 \Leftrightarrow r_{ij} \geq a \end{cases} \quad (20.5)$$

where  $F_j$  is the resultant force acting on the  $j$ th CP and  $ar_{ij}$  is a new parameter, which determines the type of the force, where +1 and -1 correspond to attracting force and repelling force, respectively:

$$ar_{ij} = \begin{cases} +1 & w.p. k_t \\ -1 & w.p. 1 - k_t \end{cases} \quad (20.6)$$

where “w.p.” stands for “with the probability.” In this algorithm, each CP is considered as a “charged sphere” with radius  $a$ , which has a uniform volume charge density.

- **Step 3: Solution construction.** Move each CP to the new position and find the velocities as:

$$X_{j,new} = rand_{j1} \cdot k_a \cdot \frac{F_j}{m_j} \cdot \Delta t^2 + rand_{j2} \cdot k_v \cdot V_{j,old} \cdot \Delta t + X_{j,old} \quad (20.7)$$

$$V_{j,new} = \frac{X_{j,new} - X_{j,old}}{\Delta t} \quad (20.8)$$

where  $k_a$  and  $k_v$  are the acceleration and the velocity coefficients, respectively. These can be obtained as follows: If  $rand_{j1}$  and  $rand_{j2}$  are two random numbers uniformly distributed in the range [0,1], then:

$$k_a = 0.5(1 + iter/iter_{max}), \quad k_v = 0.5(1 - iter/iter_{max}) \quad (20.9)$$

- **Step 4: CP position correction.** If each CP leaves the predefined bounds, correct its position using the harmony search-based handling approach as described in Kaveh and Talatahari [15].
- **Step 5: CP ranking.** Considering the values of the fitness function, sort the CPs in an ascending order.
- **Step 6: CM updating.** Include the better new vectors in the CM, and exclude the worst ones from the CM.

### Level 3: Terminating Criterion Controlling

Repeat the search level steps until a terminating criterion is satisfied.

Figure 20.2 shows the flowchart of the CSS algorithm.

## 20.4 Multi-Objective Charged System Search Optimization Algorithm

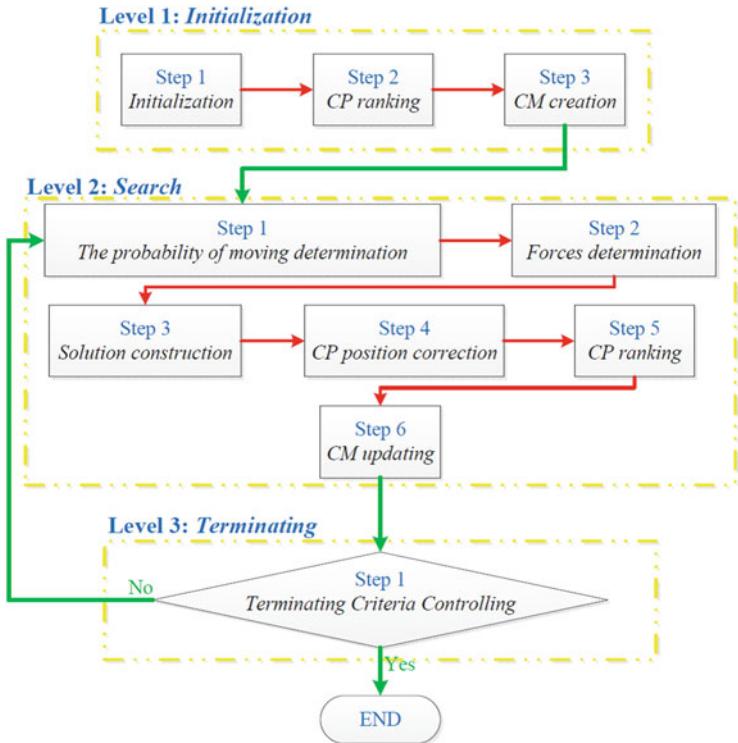
This algorithm is based on pure charged system search (CSS) algorithm. By using this algorithm in a multi-objective optimization procedure, some changes are made and some additional steps are considered.

### 20.4.1 Algorithm

This algorithm consists of the following steps:

**Step 1:** Initialize the charged particles' (CPs) magnitudes randomly. Initial speed of each particle is considered as zero.

**Step 2:** Determine the magnitude of charge for each CP. For this purpose, the vector of objectives for each CP is calculated. Then, dominance rank of each CP is obtained. Dominance rank of a solution is related to the number of solutions in the population that dominates the considered solution. Figure 20.3 represents the procedure for determining the dominance rank of some solutions.



**Fig. 20.2** Summarized flowchart of the CSS

Diversity loss is observable in many metaheuristics. In order to face the drawback related to the stagnation of a population, diversity must be maintained in the population. In general, diversification method deteriorates solutions that have a high density in their neighborhoods. For a solution  $i$ , the distances  $d_{ij}$  between  $i$  and other solutions of the population  $j$  are computed. The magnitude of charge for a solution  $i$ ,  $q(i)$ , is calculated as:

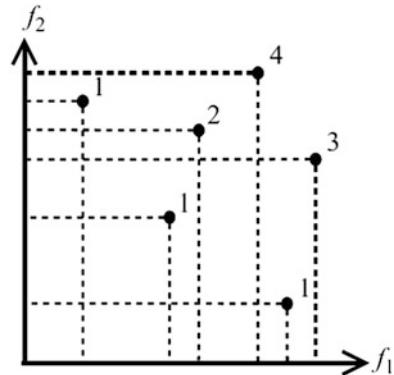
$$q(i) = \frac{1}{DR_i \times m_i} \quad i \in [1, 2, \dots, N] \quad (20.10)$$

where  $DR_i$  is the dominance rank of solution  $i$  and  $m_i = \sum_{j \in pop} sh(d_{ij})$ .

Sharing function,  $sh(d_{ij})$ , is defined as follows:

$$sh(d_{ij}) = \begin{cases} 1 - \frac{d_{ij}}{\sigma} & \text{if } d_{ij} < \sigma \\ 0 & \text{otherwise} \end{cases} \quad (20.11)$$

**Fig. 20.3** Dominance rank determination



The constant  $\sigma$  represents the non-similarity threshold. The effectiveness of the sharing principle depends mainly on these two parameters that must be set carefully. Indeed, diversification becomes inefficient with a low value of  $\sigma$ , but the convergence speed of the front becomes too small when this value is too high.

**Step 3:** Now, CM should be created. For this purpose, the particles with dominance rank equal to 1 are selected as CM.

**Step 4:** While  $\text{iter} \leq \text{itermax}$ , in other words, since a terminating criterion is not satisfied, repeat the following steps:

- (a) Determine the CM particle and CP particle. This means that the location of all the particles in the population and archive should be determined. It should be noted that the objective space is divided into  $z$  parts.

The space division method employed here is the same as the formulation which is introduced in [16]. According to this method, for each particle with  $F(x) = (f_1(x), f_2(x))$ , a value  $\sigma_i$  is defined as:

$$\sigma = \frac{f_1^2 - f_2^2}{f_1^2 + f_2^2} \quad (20.12)$$

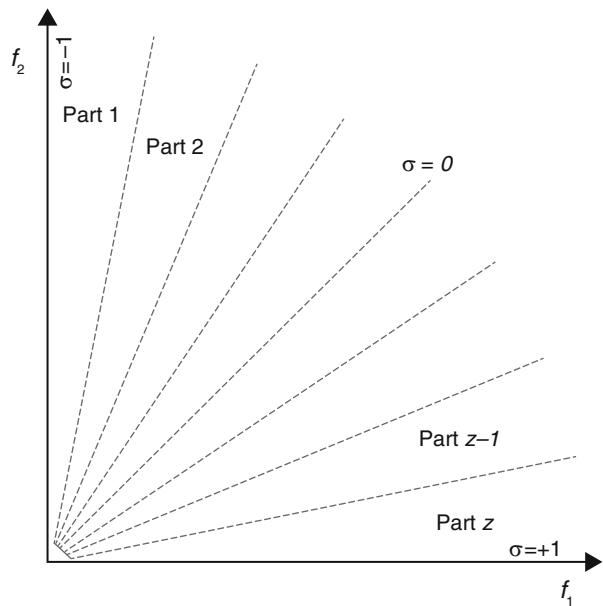
In case the objectives are not in the same range, for a two-objective optimization problem,  $\sigma$  can be calculated as below:

$$\sigma = \frac{m_1^2 - m_2^2}{m_1^2 + m_2^2}, \quad m_1 = \frac{f_1 - f_{\min 1}}{f_{\max 1} + f_{\min 1}}, \quad m_2 = \frac{f_2 - f_{\min 2}}{f_{\max 2} - f_{\min 2}} \quad (20.13)$$

where  $f_{\max 1}(f_{\min 1}), f_{\max 2}(f_{\min 2})$  are the maximum (minimum) values of the first and second objective of the particles in the population or archive, respectively. The schematic demonstration of different parts is shown in Fig. 20.4.

- (b) Calculate the resultant force vector for each CP or CM particles as:

**Fig. 20.4** Division of the objective space by assigning parameter  $\sigma$  to each particle



$$F_j = q_j \sum_{i, i \neq j} \left( \frac{q_i}{a^3} r_{ij} \cdot i_1 + \frac{q_i}{r_{ij}^2} \cdot i_2 \right) ar_{ij} p_{ij} (X_i - X_j), \quad \begin{cases} i_1 = 1, i_2 = 0 \Leftrightarrow r_{ij} < a \\ i_1 = 0, i_2 = 1 \Leftrightarrow r_{ij} \geq a \end{cases} \quad (20.14)$$

where the probability of moving,  $p_{ij}$ , can be calculated according to Eq. (20.4);  $F_j$  is the resultant force acting on the  $j$ th particle; and  $ar_{ij}$  determines the type of the force which is explained in the previous sections. This parameter can be calculated as follows:

$$\text{if } i, j \in CP \text{ or } i, j \in CM \Rightarrow ar_{ij} = \begin{cases} +1 & \text{if } rand \leq k_t \\ -1 & \text{if } rand < 1 - k_t \end{cases}$$

$$\text{if } i \in CP \text{ and } j \in CM \Rightarrow ar_{ij} = -1$$

*This means : ACM particle is repelled by all CP particles.*

$$\text{if } i \in CM \text{ and } j \in CP \Rightarrow ar_{ij} = +1$$

*This means : ACM particle attracts all CP particles.*

- (c) Compute the new position and velocity of each particle using Eqs. (20.7) and (20.8). When the current position of a particle is obtained, the following control should be performed:

*if  $j \in CP \Rightarrow \text{Part}(X_{newj})$  should be the same as  $\text{Part}(X_{oldj})$ ; otherwise,  $X_{newj} = X_{oldj}$*

This means that each particle of CP should remain in its initial part up to the end the optimization procedure, but CM particles can be moved to other parts.

- (d) Update the magnitude of each particle of CP and CM. Calculate the dominance rank of them and select the new members of CM. This means that all particles which have dominance rank equal to one should be selected as new CM.
- (e) In this step, each particle of CM is compared with other particles of CM. In other words, the Euclidean distance between the objective vectors of all the particles in the CM is calculated, and if this value is smaller than a positive predefined value, one of them is eliminated. By this approach, crowded region cannot be generated in the objective space.

## 20.5 Multi-Criteria Decision Making

The aim of solving multi-objective optimization problems is to help a decision maker (DM) to find a Pareto solution that copes with his preferences. One of the fundamental questions in MOPs resolution is related to the interaction between the problem-solver and the decision maker. Indeed, the Pareto optimal solutions cannot be ranked globally. The role of the decision maker is to specify some extra information to select his favorite solution.

Many different approaches can be used for the decision-making process [17]. A simple method for multi-criteria decision-making problem, so-called multi-criteria tournament decision-making method (MTDM), is due to [18]. This method provides the ranking of alternatives from best to worst, according to the preferences of a human decision maker. It has another positive aspect that involves few input parameters, just the importance of weight of each criterion. This method introduces a function  $R$ , capable of reflecting the DM global interests. In order to find this function, first each possible solution is compared to the others, considering only the  $i$ th criterion. The pairwise comparisons are performed through the tournament function  $T_i(a, A)$ , which counts the ratio of times the alternative  $a$  wins the tournament against other solution  $b$  from  $A$ . Hence, considering that  $a$  is a non-dominated point in the objective space,  $T_i(a, A)$  can be stated as

$$T_i(a, A) = \sum_{\forall b \in A, a \neq b} \frac{t_i(a, b)}{(|A| - 1)} \quad (20.15)$$

where

$$t_i(a, b) = \begin{cases} 1 & \text{if } f_i(b) - f_i(a) > 0 \\ 0 & \text{otherwise.} \end{cases} \quad (20.16)$$

The tournament function  $T_i(a, A)$  assigns a score to each solution in Pareto front. The assigned score works as a performance measure, which provides a distinct ordering of the elements of  $A$  for each criterion. In order to generate the global ranking, taking into account all criteria and their respective weights  $w_i$  (priority factors), the scores are aggregated into the global ranking function  $R$ . The weighted geometric mean which is utilized by many different researchers is considered as the aggregation function in this study as follows:

$$R(a) = \left( \prod_{i=1}^n T_i(a, A)^{w_i} \right)^{\frac{1}{n}} \quad (20.17)$$

where  $n$  is the number of objective functions. The priority weights must be specified by the DM in accordance with the following conditions:

$$w_i > 0 \text{ \& } \sum_{i=1}^n w_i = 1 \quad (20.18)$$

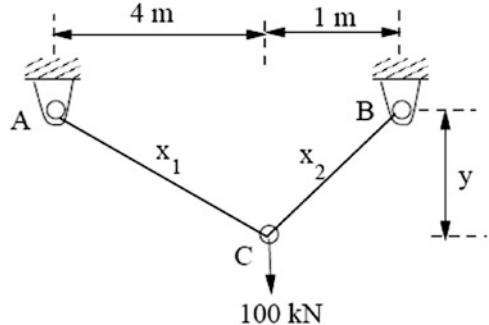
The ranking index  $R(a)$  gives an idea of how much each alternative is preferred to the others. In other words:

if  $R(a) > R(b)$ , then  $a$  is preferred to  $b$  and when  $R(a) = R(b)$ , then  $a$  is indifferent to  $b$ .

## 20.6 Numerical Examples

In this section, some numerical results are presented in order to show the performance of the pure CSS algorithm in multi-objective optimization problems. The algorithms are coded in MATLAB, and in order to handle the constraints, a penalty approach is utilized. When the constraints are in the range of allowable limits, the penalty is zero; otherwise, the amount of penalty is obtained by dividing the violation of allowable limit to the limit itself. For the examples presented in this chapter, the CSS algorithm parameters are set as follows:  $k_a = 2$ ,  $k_v = 2$ , the number of agents is taken as 100, the maximum number of iterations is set to 100,  $a = 1$ ,  $\Delta T = 1$ , and  $k_t = 0.5$ . The algorithm is run with an archive size of 100. In this chapter, a real-coded NSGA-II is utilized with a population size of 100, a crossover probability of 0.9 ( $p_c = 0.9$ ), a tournament selection, and a mutation rate of  $1/u$  (where  $u$  is the number of decision variables), and distribution indexes for crossover and mutation operators are taken as  $\eta_c = 20$  and  $\eta_m = 20$ , respectively (as recommended in [8]). MOPSO uses a population of 100 particles, an archive size of 100 particles, a mutation rate of 0.5, and 30 divisions for the adaptive grid

**Fig. 20.5** Schematic of the 2-bar truss problem



[11]. Also, s-MOPSO is run with a population of 100 particles, an archive size of 100 particles, and a mutation probability of 0.05 [16]. And the parameters which are considered for CSS–MOPSO consist of  $C1 = 1$ ,  $C2 = 2$ ,  $R = 15$ ,  $rld = 0.01$ ,  $rud = 0.05$ , mutation probability of 0.1, archive size of 100, and a population of 50 particles [12]. For all examples presented in this chapter, the number of fitness function evaluation (structural analysis) in multi-objective optimization phase is restricted to 30,000.

The results obtained by CSS are compared to the original MOPSO [11], s-MOPSO [16], NSGA [8], and MOCHS [21].

### 20.6.1 Design of a 2-Bar Truss

This problem was originally studied using  $\epsilon$ -constraint method [19]. As shown in Fig. 20.5, the truss has to carry a certain load without elastic failure.

Thus, in addition to the objective of designing the truss for minimum volume, there are additional objectives of minimizing stresses in each of the two members AC and BC. The two-objective optimization problem for three variables  $y$  (vertical distance between B and C in m),  $x_1$  (length of AC in m), and  $x_2$  (length of BC in m) is constructed as follows:

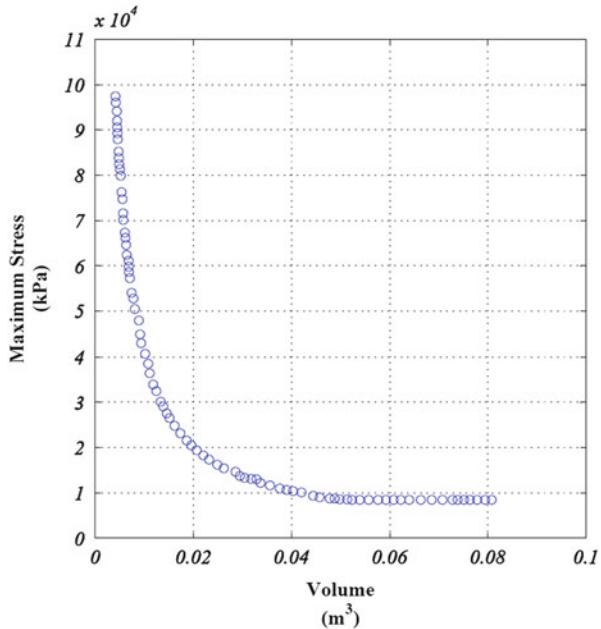
$$\begin{aligned} & \text{minimize } f_1(x) = x_1\sqrt{16+y^2} + x_2\sqrt{1+y^2} \\ & \text{minimize } f_2(x) = \max(\sigma_{AC}, \sigma_{BC}) \end{aligned}$$

$$S.T \quad \left\{ \begin{array}{l} \max(\sigma_{AC}, \sigma_{BC}) \leq 10^5 \\ 1 \leq y \leq 3 \\ x \geq 0 \end{array} \right.$$

$$\text{where } \sigma_{AC} = \frac{20\sqrt{16+y^2}}{yx_1} \text{ and } \sigma_{BC} = \frac{80\sqrt{1+y^2}}{yx_2}.$$

Figure 20.6 shows the Pareto front obtained using the CSS method. Also, the two extreme objective values obtained by various algorithms are compared in Table 20.1.

**Fig. 20.6** Pareto optimal front obtained using CSS method for 2-bar truss design problem



**Table 20.1** Comparison of the results for a 2-bar truss design problem

Optimization method	EM-MOPSO [20]	NSGA-II [8]	MOCHS [21]	CSS (present work)
Extreme obtained values ( $m^3$ , kN)	[0.004026, 99,996] [0.05273, 8434.493]	[0.00407, 99,755] [0.05304, 8439]	[0.00375, 99,847] [0.0537, 7685]	[0.00412, 99,457] [0.08078, 8434.23]

### 20.6.2 Design of an I-Beam

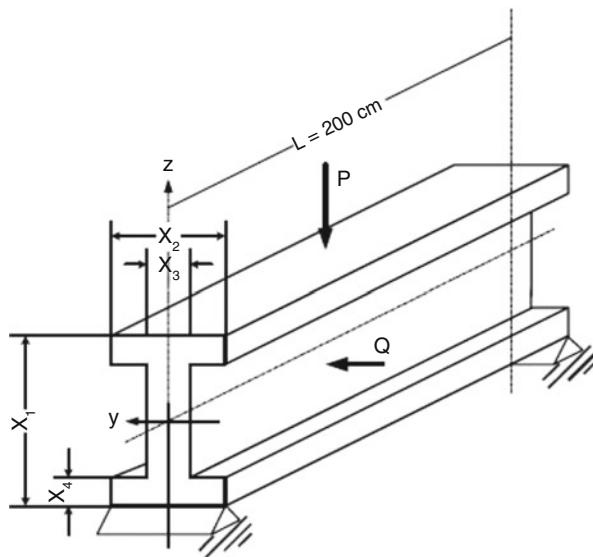
The second design problem is taken from [20]. The problem is to find the dimension of the beam shown in Fig. 20.7. In this design problem, the dimensions of the geometric and strength constraints should be satisfied, and at the same time, the cross-sectional area of the beam and the static deflection of the beam should be minimized under a force P. The mathematical formulation of the problem is as follows:

$$\begin{aligned} & \text{minimize cross-sectional area } (cm^2) : f_1 = 2x_2x_4 + x_3(x_1 - 2x_4) \\ & \text{minimize displacement } (cm) : f_2 = \frac{PL^3}{48EI} \end{aligned}$$

where

$$I = \frac{1}{12} \left\{ x_3(x_1 - 2x_4)^3 + 2x_2x_4[4x_4^2 + 3x_1(x_1 - 2x_4)] \right\}.$$

**Fig. 20.7** Schematic of an I-beam design problem



$$x_i, \quad i = 1, 2, 3, 4 \quad (\text{Find})$$

Subject to:

$$g(x) = \sigma_a - \left( \frac{M_y}{Z_y} + \frac{M_z}{Z_z} \right) \geq 0 \quad \text{and} \quad \begin{cases} 10 \leq x_1 \leq 80 \\ 10 \leq x_2 \leq 50 \\ 0.9 \leq x_3 \leq 5 \\ 0.9 \leq x_4 \leq 5 \end{cases} \quad \text{where}$$

$$M_y = \frac{P}{2} \times \frac{L}{2}, \quad M_z = \frac{Q}{2} \times \frac{L}{2},$$

$$Z_y = \frac{1}{6x_1} \left\{ x_3(x_1 - x_4)^3 + 2x_2x_4[4x_4^2 + 3x_1(x_1 - 2x_4)] \right\},$$

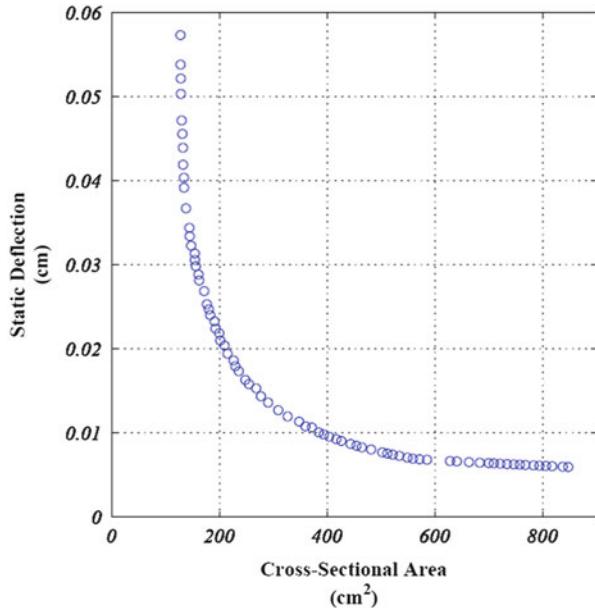
$$Z_z = \frac{1}{6x_1} \left\{ x_3^3(x_1 - x_4) + 2x_2^3x_4 \right\},$$

$$E = 2 \times 10^4 \text{ kNm}^{-2},$$

$$\sigma_a = 16 \text{ kNm}^{-2}, \quad P = 600 \text{ kN}, \quad Q = 50 \text{ kN}, \quad \text{and} \quad L = 200 \text{ cm}$$

Figure 20.8 shows the Pareto front obtained after 100 iterations. The CSS obtained the minimal cross-sectional area of 127.8201 units for a deflection of 0.0573, and for the minimal deflection of 0.0059 units, the cross-sectional area is 847.5709 units. EM-MOPSO obtained the minimal cross-sectional area of 127.9508 units for a deflection of 0.05368 units, and for the minimal deflection of 0.005961 units, the cross-sectional area was 829.5748 units. NSGA-II obtained a

**Fig. 20.8** Pareto optimal front obtained using the CSS method for the I-beam design



minimal cross-sectional area of 127.2341 units with deflection of 0.0654 units and a minimal deflection of 0.0060 units with cross-sectional area of 829.8684 units.

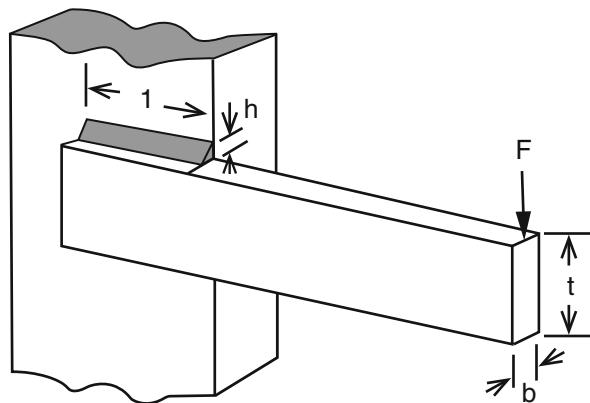
### 20.6.3 Design of a Welded Beam

The third design problem was studied by [20]. A beam needs to be welded on to another beam and must carry a certain load, Fig. 20.9.

The overhang has a length of 14 in. and a force  $F$  of 6000 lb is applied at the end of the beam. The objective of the design is to minimize the cost of fabrication and the end deflection. The mathematical formulation of the two-objective optimization problem is as follows:

$$\begin{aligned} \text{minimize } & \begin{cases} f_1(x) = 1.10471h^2l + 0.04811tb(14 + l) \\ f_2(x) = \delta(x) = \frac{2.1952}{t^3b} \end{cases} \\ \times \text{subject to } & \begin{cases} g_1(x)^1 = 13,600 - \tau(x) \geq 0 \\ g_2(x)^2 = 30,000 - \sigma(x) \geq 0 \\ g_3(x) = b - h \geq 0 \\ g_4(x) = P_c(x) - 6000 \geq 0. \end{cases} \end{aligned}$$

**Fig. 20.9** Schematic of a welded beam design



The first constraint ensures that the shear stress developed at the support location of the beam is less than the allowable shear strength of the material (13,600 psi). The second one ensures that the normal stress developed at the support location of the beam is less than the allowable yield strength of the material (30,000 psi). The third ensures that the thickness of the beam is not less than the weld thickness from a practical standpoint. The fourth one ensures that the allowable buckling load of the beam (along the  $t$  direction) is greater than the applied load  $F$ . The stress and buckling terms are as follows:

$$\tau(x) = \sqrt{(\tau')^2 + (\tau'')^2 + \frac{l\tau'\tau''}{\sqrt{0.25(l^2 + (h+t)^2)}}}$$

$$\tau' = \frac{6000}{\sqrt{2}hl}$$

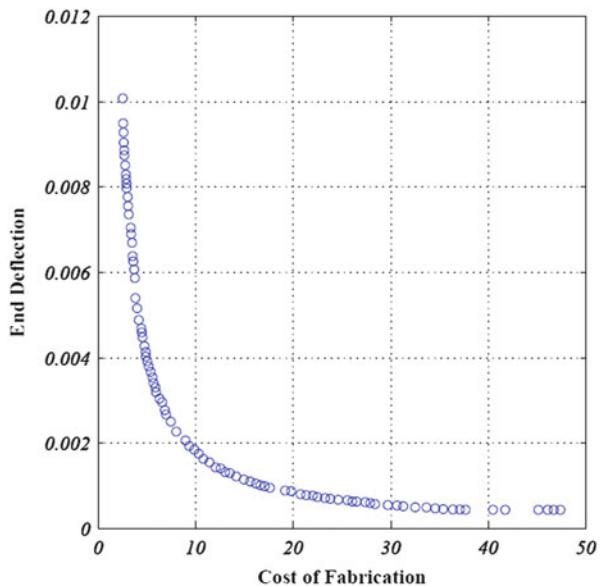
$$\tau'' = \frac{6000(14 + 0.5l)\sqrt{0.25(l^2 + (h+t)^2)}}{2\left\{0.707hl\left(\frac{l^2}{12} + 0.25(h+t)^2\right)\right\}}$$

$$\sigma(x) = \frac{504,000}{t^2b}$$

$$P_c(x) = 64,746.022(1 - 0.0282346t)tb^3$$

Figure 20.10 shows the optimized non-dominated solutions obtained using the CSS algorithm. EM-MOPSO found the minimal cost solution as 2.382 units with deflection 0.0157 in. and the minimal deflection as 0.000439 with a cost of 36.4836 units. For NSGA-II, the minimal cost was 3.443 units for deflection of 0.0101 units, and the minimal deflection was 0.004 with a cost of 36.9121 units. For the CSS, the minimal cost is 2.5112 units for deflection of 0.000439 units, and the minimal deflection is 0.0108 with a cost of 47.3722 units.

**Fig. 20.10** Pareto optimal front obtained using the CSS method for the welded beam design



#### 20.6.4 Design of a 25-Bar Truss

Another famous 25-bar truss is considered as shown in Fig. 20.11 [12]. Again, the problem is to find the cross-sectional area of members such that the total structural weight and the displacement in y direction at node 1 are minimized concurrently. 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)  $A_{22}-A_{25}$ .

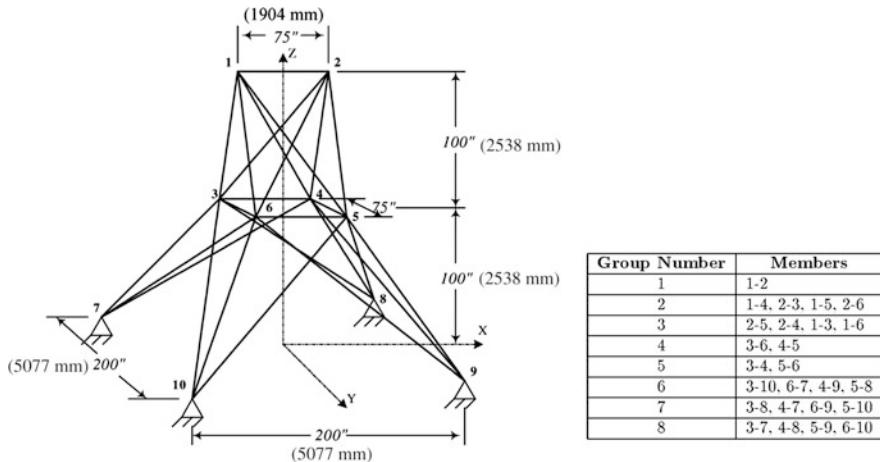
The applied load to this structure is:

$$F_{X(1)} = 4.45 \text{ (kN)}, F_{Y(1)} = -44.5 \text{ (kN)}, F_{Z(1)} = -44.5 \text{ (kN)}, F_{Y(2)} = -44.5 \text{ (kN)}, \\ F_{Z(2)} = -44.5 \text{ (kN)}, F_{X(3)} = 2.25 \text{ (kN)}, F_{X(6)} = 2.67 \text{ (kN)}.$$

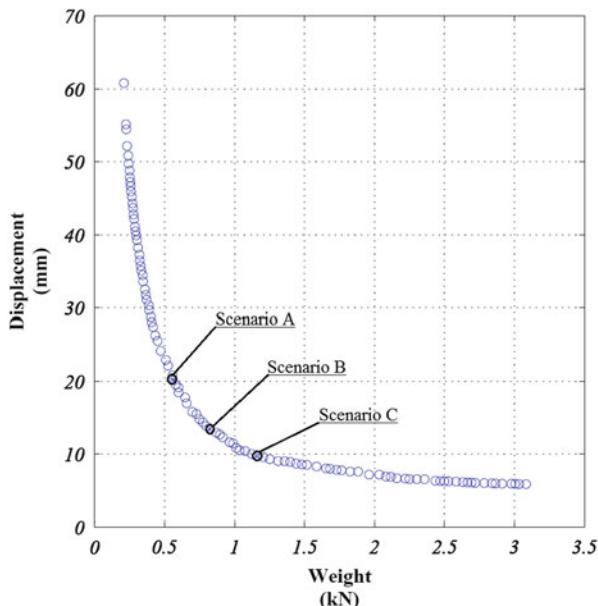
The upper and lower bounds for the cross sections of each truss element are  $64.45 \text{ mm}^2$  ( $0.1 \text{ in}^2$ ) and  $2191.47 \text{ mm}^2$  ( $3.4 \text{ in}^2$ ), respectively. The modulus of elasticity is taken as  $E = 68.97 \text{ kN/mm}^2$  ( $1 \times 10^4 \text{ ksi}$ ) and the weight density as  $\rho = 2.714E - 8 \text{ kN/mm}^2$  ( $0.1 \text{ lb/in}^2$ ). Constraints on the truss limit the principal stress  $\sigma_j$  in each element to a maximum allowable stress value of  $\sigma_j = \pm 0.27584 \text{ kN/mm}^2$  ( $\pm 40 \text{ ksi}$ ).

The Pareto front obtained by the CSS algorithm is shown in Fig. 20.12. Also, the two extreme objective values obtained in ten runs of algorithms are shown in Table 20.2.

In this example, after finding the Pareto front, the next step is to ask DMs to notify their preferences by considering all information that is integrated in the



**Fig. 20.11** Schematic of the 25-bar space truss structures and its member grouping



**Fig. 20.12** The Pareto front of a 25-bar truss structure and the best solutions according to three different scenarios

Pareto front. Many different scenarios are possible for a considered problem. For example, these scenarios can be as follows:

Scenario A: The first criterion (objective) is more important: e.g.,  $(w_1, w_2) = (0.6, 0.4)$ .

**Table 20.2** Comparison of the extreme values obtained by different methods for a 2-bar truss design problem

Optimization method	Extreme values obtained (mm, kN)
CSS–MOPSO [12]	[5.8437, 4.8111] [62.9807, 0.3440]
s-MOPSO [16]	[5.8437, 4.8917] [62.7832, 0.3239]
MOPSO [11]	[5.8791, 4.4836] [60.3942, 0.3642]
NSGA-II [8]	[5.8437, 4.8297] [64.5579, 0.3141]
CSS (present work)	[5.8697, 4.7989] [63.6643, 0.2176]

Scenario B: The first criterion (objective) is as important as the second criterion: e.g.,  $(w_1, w_2) = (0.5, 0.5)$ .

Scenario C: The second criterion (objective) is more important: e.g.,  $(w_1, w_2) = (0.4, 0.6)$ .

The selected solutions corresponding to each considered scenario are indicated in Fig. 20.11, and in Table 20.3 the best solutions for different scenarios are presented and compared to those of Ref. [12]. By calculating the index,  $R_i = \sqrt{f_1^{w_1} \times f_2^{w_2}}$ , for the results obtained by CSS and [12], the efficiency of the proposed algorithm is clarified.

### 20.6.5 Design of a 56-Bar

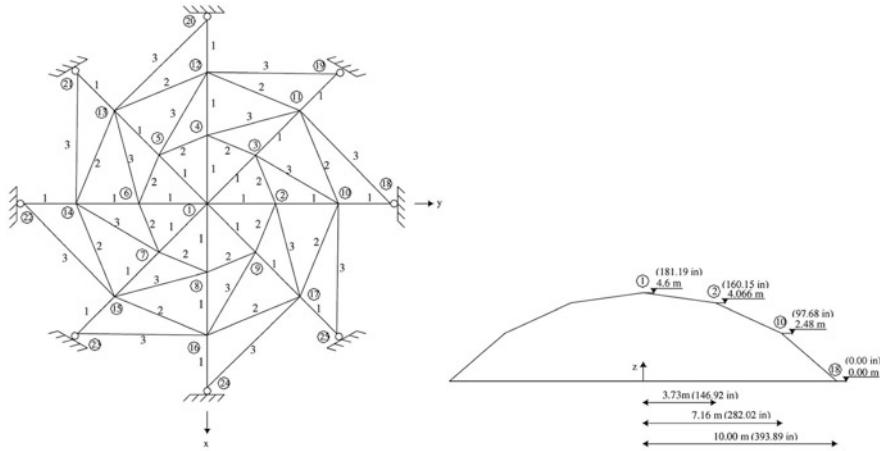
This example is a 56-bar space truss studied in [22] with members categorized in three groups as shown in Fig. 20.13. Joint 1 is loaded with 4 kN (899.24 lb) in the y direction and 30 kN (6744.267 lb) in the z direction, while the remaining free nodes are loaded with 4 kN (899.24 lb) in the y direction and 10 kN (2248.09 lb) in the z direction.

The vertical displacements of joint 4, 5, 6, 12, 13, and 14 are restricted to 40 mm (0.158 in), while the displacement of joint 8 in the y direction is limited to 20 mm (0.079 in). The modulus of elasticity and the minimum and the maximum member cross-sectional areas are taken as  $210 \text{ kN/mm}^2$  ( $3.05 \times 10^4 \text{ ksi}$ ),  $200 \text{ mm}^2$  ( $0.31 \text{ in}^2$ ), and  $2000 \text{ mm}^2$  ( $3.1 \text{ in}^2$ ), respectively. The total structural volume,  $F_1(x)$ , and the displacement at node 1,  $F_2(x)$ , have to be minimized simultaneously. Objective functions are:

$$\min \begin{cases} F_1(x) = \sum_{i=1}^{56} A_i l_i \\ F_2(x) = \sqrt{\delta_{1X}^2 + \delta_{1Y}^2 + \delta_{1Z}^2} \end{cases}$$

Table 20.3 Best selected solutions for a 2-bar truss design problem

	Scenario A			Scenario B			Scenario C		
	$f_1$ (kN)	$f_2$ (mm)		$f_1$ (kN)	$f_2$ (mm)		$f_1$ (kN)	$f_2$ (mm)	
Algorithm	$w_1 = 0.6$	$w_2 = 0.4$	$R_i$	$w_1 = 0.5$	$w_2 = 0.5$	$R_i$	$w_1 = 0.4$	$w_2 = 0.6$	$R_i$
CSS (present work)	0.558	20.2810	1.5325	0.823	13.4183	1.8229	1.159	9.6868	2.0355
CSS-MOPSO [12]	1.189	16.7307	1.8504	1.548	12.7422	2.0962	2.036	9.6144	2.2732



**Fig. 20.13** A 56-bar space truss structure

**Table 20.4** Comparison of the extreme values obtained by different methods for the 56-bar truss

Optimization method	Extreme obtained values (mm, mm <sup>3</sup> )
CSS-MOPSO [12]	[2,2148, 402,923,368.6] [7,5495, 120,812,690.1]
s-MOPSO [16]	[2,2137, 402,417,631.6] [7,4721, 120,151,168.8]
MOPSO [11]	[2,2154, 403,070,300.4] [7,0825, 123,191,518.1]
NSGA-II [8]	[2,2137, 402,403,612.4] [7,4883, 119,960,278.7]
CSS (present work)	[1.1061, 478,422,670.1] [10.4342, 50,644,453.1]

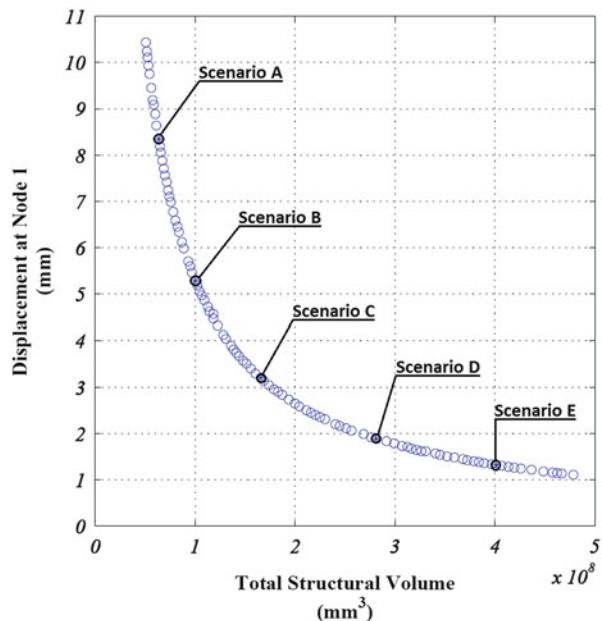
The two extreme objective values, obtained in ten runs of various algorithms and the proposed method, are compared in Table 20.4. In addition, the Pareto front obtained via the CSS algorithm is shown in Fig. 20.14.

The process of decision making and finding the best solution is performed identical to the previous example. In this example, in order to show the wide range of possible solutions, five different scenarios are considered. The results are aggregated in Table 20.5. The selected solutions corresponding to each considered scenario are provided in Fig. 20.14.

## 20.6.6 Design of a 272-Bar Transmission Tower

The fifth test example is the transmission tower, depicted in Fig. 20.15, together with its geometric characteristics. This example is generated by the author and his

**Fig. 20.14** The Pareto front of a 56-bar space truss structure and the best solutions according to five different scenarios



colleagues. The nodal coordinates and end nodes of each member are provided in Tables 20.6 and 20.7, respectively. The design variables considered are cross-sectional area of members, divided into 28 groups as shown in Table 20.8.

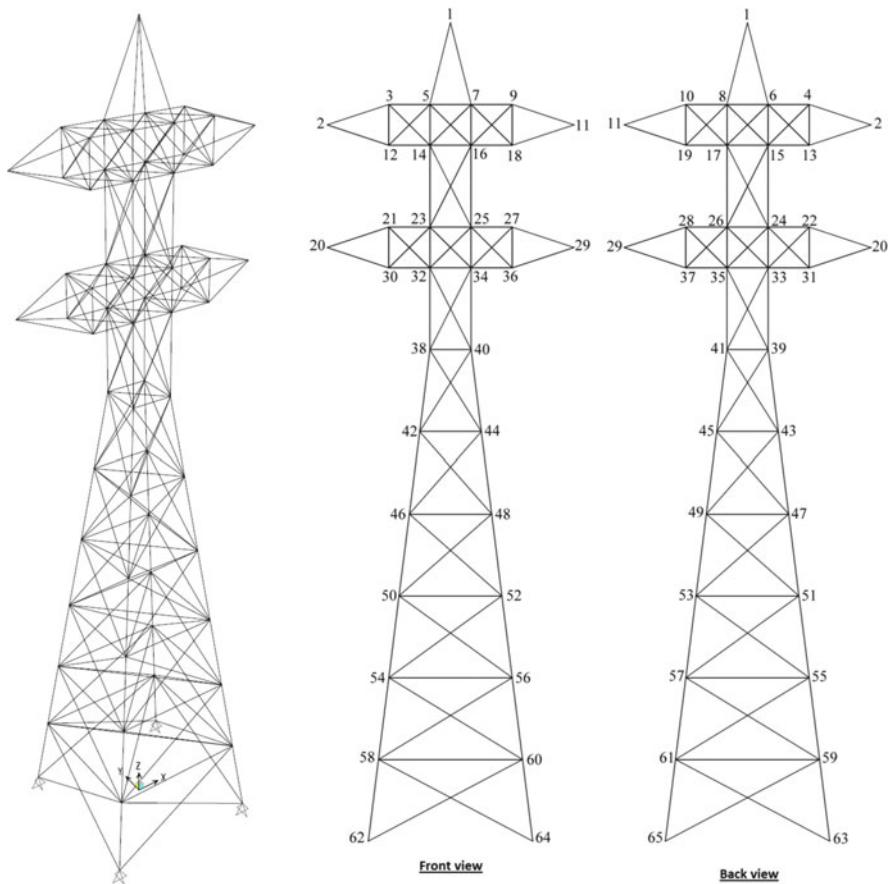
Joints 1, 2, 11, 20, and 29 are loaded with  $20\text{ kN}$  in the x and y directions and  $-40\text{ kN}$  in the z direction, while the remaining free nodes are loaded with  $5\text{ kN}$  in the x and y directions. The vertical displacement of joints 2, 11, 20, and 29 is restricted to  $20\text{ mm}$ , while the displacements in the x and y directions are limited to  $100\text{ mm}$ . The modulus of elasticity and the minimum and the maximum member cross-sectional areas are taken as  $2 \times 10^8 \text{ kN/m}^2$  ( $3.05 \times 10^4 \text{ ksi}$ ),  $1000\text{ mm}^2$ , and  $16,000\text{ mm}^2$ , respectively. The principal stress  $\sigma_j$  in each element is restricted to the maximum allowable stress,  $\sigma_j = \pm 275000\text{ kN/m}^2$ . The total structural volume,  $F_1(x)$ , and the displacement at node 1,  $F_2(x)$ , have to be minimized simultaneously. Objective functions are:

$$\min \begin{cases} F_1(x) = \sum_{i=1}^{272} A_i l_i \\ F_2(x) = \sqrt{\delta_{1X}^2 + \delta_{1Y}^2 + \delta_{1Z}^2} \end{cases}$$

The Pareto front obtained via CSS algorithm is shown in Fig. 20.16. The process of decision making and finding the best solution is performed completely similar to those of the previous examples. In this example, in order to show the wide range of possible solutions, nine different scenarios are considered. The results are

**Table 20.5** Different possible scenarios for the 56-bar truss with corresponding solutions

Scenario	Importance of criteria	Possible priority weights	Selected solution by MTDM (mm, mm <sup>3</sup> )					
			CSS-MOPSO [12]		CSS (present work)			
			$f_1$ (mm)	$f_2$ (mm <sup>3</sup> )	$R_i$	$f_1$ (mm)	$f_2$ (mm <sup>3</sup> )	$R_i$
A	$c_1 > c_2$	[0.9,0.1]	6.1144	134,900,811.2	<b>5.7592</b>	8.3478	63,593,336.6	<b>6.3808</b>
B	$c_1 > c_2$	[0.7,0.3]	4.4740	166,100,766.2	<b>28.8934</b>	5.2836	100,114,327.7	<b>28.3857</b>
C	$c_1 \sim c_2$	[0.5,0.5]	3.2959	208,951,138.7	<b>161.9961</b>	3.2022	165,628,602.5	<b>151.7560</b>
D	$c_1 < c_2$	[0.3,0.7]	2.5679	267,415,709.3	<b>1025.5389</b>	1.8809	281,480,915.0	<b>996.4649</b>
E	$c_1 << c_2$	[0.1,0.9]	2.2709	353,607,163.4	<b>7322.2470</b>	1.3168	401,089,550.8	<b>7541.0982</b>



**Fig. 20.15** Schematic of a 272-bar transmission tower

aggregated in Table 20.9. The selected solutions corresponding to each considered scenario are provided in Fig. 20.16.

## 20.7 Concluding Remarks

Optimization problems encountered in practice are seldom mono-objective. In general, there are many conflicting objectives to handle. An efficient procedure for solving multi-objective optimization problems using charged system search algorithm is presented in this chapter. The algorithm is also applied to six engineering design problems to demonstrate its applicability in practical problems. The results obtained amply demonstrate that the presented approach is efficient in converging to the true Pareto fronts and finding a diverse set of solutions along

**Table 20.6** Nodal coordinates of the transmission tower

Node	X (m)	Y (m)	Z (m)	Node X (m)	Node Y (m)	Node Z (m)	Node X (m)	Node Y (m)	Node Z (m)	Node X (m)	Node Y (m)	Node Z (m)
<b>1</b>	0	0	20	<b>14</b>	-0.5	-0.5	<b>27</b>	1.5	-0.5	<b>40</b>	0.5	-0.5
<b>2</b>	-3	0	17.5	<b>15</b>	-0.5	0.5	<b>28</b>	1.5	0.5	<b>41</b>	0.5	0.5
<b>3</b>	-1.5	-0.5	18	<b>16</b>	0.5	-0.5	<b>29</b>	3	0	<b>42</b>	-0.75	-0.75
<b>4</b>	-1.5	0.5	18	<b>17</b>	0.5	0.5	<b>30</b>	-1.5	-0.5	<b>43</b>	-0.75	0.75
<b>5</b>	-0.5	-0.5	18	<b>18</b>	1.5	-0.5	<b>31</b>	-1.5	0.5	<b>44</b>	0.75	-0.75
<b>6</b>	-0.5	0.5	18	<b>19</b>	1.5	0.5	<b>32</b>	-0.5	-0.5	<b>45</b>	0.75	0.75
<b>7</b>	0.5	-0.5	18	<b>20</b>	-3	0	<b>33</b>	14.5	0.5	<b>46</b>	-1	-1
<b>8</b>	0.5	0.5	18	<b>21</b>	-1.5	-0.5	<b>34</b>	0.5	-0.5	<b>47</b>	-1	1
<b>9</b>	1.5	-0.5	18	<b>22</b>	-1.5	0.5	<b>35</b>	0.5	0.5	<b>48</b>	1	-1
<b>10</b>	1.5	0.5	18	<b>23</b>	-0.5	1.5	<b>36</b>	1.5	-0.5	<b>49</b>	1	1
<b>11</b>	3	0	17.5	<b>24</b>	-0.5	0.5	<b>37</b>	1.5	0.5	<b>50</b>	-1.25	-1.25
<b>12</b>	-1.5	-0.5	17	<b>25</b>	0.5	-0.5	<b>38</b>	-0.5	-0.5	<b>51</b>	-1.25	1.25
<b>13</b>	-1.5	0.5	17	<b>26</b>	0.5	0.5	<b>39</b>	-0.5	0.5	<b>52</b>	1.25	-1.25
										<b>63</b>	-2	2
										<b>64</b>	2	-2
										<b>65</b>	2	0

**Table 20.7** End nodes of the members of a 272-bar transmission tower

Member	Start point	End point	Member point	Start point	End point	Member point	Start point	End point	Member point	Start point	End point	Member point	Start point	End point	Member point	Start point	End point	Member point	Start point	End point	Member point	Start point	End point	Member point
	1	5	35	10	19	69	34	32	103	4	15	137	9	19	171	38	42	205	47	48	239	57	56	56
1	6	36	8	17	70	35	37	104	6	13	138	10	18	172	40	44	206	46	49	240	56	54	54	
2	7	37	6	15	71	37	36	105	21	24	139	21	31	173	41	45	207	46	50	241	54	57	57	
3	8	38	4	13	72	36	34	106	22	23	140	22	30	174	39	43	208	48	52	242	55	56	56	
4	3	39	12	13	73	21	30	107	24	25	141	23	33	175	38	43	209	49	53	243	54	58	58	
5	4	40	13	15	74	23	32	108	23	26	142	32	24	176	39	42	210	47	51	244	56	60	60	
6	12	41	15	14	75	25	34	109	26	27	143	25	35	177	40	45	211	46	52	245	57	61	61	
7	13	42	14	12	76	27	36	110	25	28	144	26	34	178	41	44	212	48	50	246	55	59	59	
8	9	43	15	17	77	28	37	111	31	32	145	27	37	179	38	44	213	49	51	247	54	60	60	
9	10	44	17	16	78	26	35	112	30	33	146	28	36	180	40	42	214	47	53	248	56	58	58	
10	11	45	16	14	79	24	33	113	33	34	147	14	23	181	41	43	215	46	51	249	57	59	59	
11	19	46	17	19	80	22	31	114	32	35	148	16	25	182	39	45	216	47	50	250	55	61	61	
12	20	47	19	18	81	4	5	115	35	36	149	17	26	183	42	43	217	48	53	251	54	59	59	
13	21	48	18	16	82	3	6	116	34	37	150	15	24	184	43	45	218	49	52	252	55	58	58	
14	20	49	38	39	83	6	7	117	21	32	151	32	38	185	45	44	219	50	51	253	56	61	61	
15	20	50	39	41	84	5	8	118	30	23	152	34	40	186	42	44	220	51	53	254	57	60	60	
16	29	51	41	40	85	8	9	119	23	34	153	35	41	187	43	44	221	53	52	255	58	59	59	
17	29	52	40	38	86	7	10	120	32	25	154	33	39	188	42	45	222	52	50	256	59	61	61	
18	29	53	21	22	87	12	15	121	25	36	155	14	25	189	42	46	223	51	52	257	61	60	60	
19	29	54	22	24	88	13	14	122	34	27	156	16	23	190	44	48	224	50	53	258	60	58	58	
20	3	55	24	23	89	15	16	123	28	35	157	17	24	191	45	49	225	50	54	259	59	60	60	
21	4	56	23	21	90	14	17	124	26	37	158	15	26	192	43	47	226	52	56	260	58	61	61	
22	6	57	24	26	91	17	18	125	26	33	159	14	24	193	42	48	227	53	57	261	58	62	62	
23	6	58	26	25	92	16	19	126	24	35	160	15	23	194	44	46	228	51	55	262	60	64	64	
24	6	59	25	23	93	3	14	127	24	31	161	16	26	195	45	47	229	50	56	263	61	65	65	
25	8	7	60	26	28	94	5	12	128	22	33	162	17	25	196	43	49	230	52	54	264	59	63	63
26	7	5	61	28	27	95	5	16	129	23	39	163	32	40	197	42	47	231	53	55	265	58	64	64
27	8	10	62	27	25	96	7	14	130	38	41	164	38	34	198	43	46	232	51	57	266	60	62	62

---

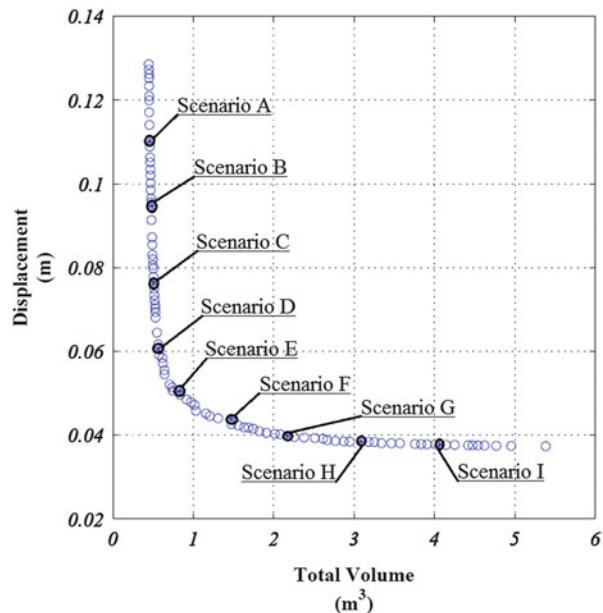
(continued)

Table 20.7 (continued)

Member	Start point	End point	Member point	Start point	End point	Member point	Start point	End point	Member point	Start point	End point	Member point	Start point	End point	Member point	Start point	End point	Member point	Start point	End point	Member point	Start point	End point	Member point
<b>29</b>	10	9	<b>63</b>	30	31	<b>97</b>	7	18	<b>131</b>	3	13	<b>165</b>	35	39	<b>199</b>	44	49	<b>233</b>	50	55	<b>267</b>	61	63	
<b>30</b>	9	7	<b>64</b>	31	33	<b>98</b>	9	16	<b>132</b>	12	4	<b>166</b>	33	41	<b>200</b>	45	48	<b>234</b>	51	54	<b>268</b>	59	65	
<b>31</b>	3	12	<b>65</b>	33	32	<b>99</b>	10	17	<b>133</b>	5	15	<b>167</b>	32	39	<b>201</b>	46	47	<b>235</b>	52	57	<b>269</b>	58	63	
<b>32</b>	5	14	<b>66</b>	32	30	<b>100</b>	8	19	<b>134</b>	14	6	<b>168</b>	33	38	<b>202</b>	47	49	<b>236</b>	53	56	<b>270</b>	59	62	
<b>33</b>	7	16	<b>67</b>	33	35	<b>101</b>	8	15	<b>135</b>	7	17	<b>169</b>	34	41	<b>203</b>	49	48	<b>237</b>	54	55	<b>271</b>	60	65	
<b>34</b>	9	18	<b>68</b>	35	34	<b>102</b>	6	17	<b>136</b>	8	16	<b>170</b>	35	40	<b>204</b>	48	46	<b>238</b>	55	57	<b>272</b>	61	64	

**Table 20.8** Member grouping of the 272-bar transmission tower

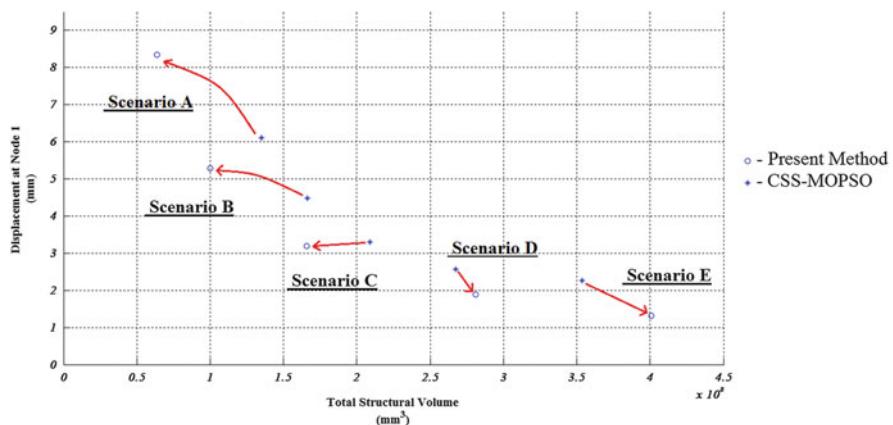
Group number	Members	Group number	Members	Group number	Members	Group number	Members
<b>1</b>	[M1–M4]	<b>8</b>	[M175–M182]	<b>15</b>	[M207–M210]	<b>22</b>	[M241–M242]
<b>2</b>	[M5–M20]	<b>9</b>	[M183–M186]	<b>16</b>	[M211–M218]	<b>23</b>	[M243–M246]
<b>3</b>	[M21–M80]	<b>10</b>	[M187–M188]	<b>17</b>	[M219–M222]	<b>24</b>	[M247–M254]
<b>4</b>	[M81–146 M]	<b>11</b>	[M189–M192]	<b>18</b>	[M223–M224]	<b>25</b>	[M255–M258]
<b>5</b>	[M147–M154]	<b>12</b>	[M193–M200]	<b>19</b>	[M225–M228]	<b>26</b>	[M259–M260]
<b>6</b>	[M155–M170]	<b>13</b>	[M201–M204]	<b>20</b>	[M229–M236]	<b>27</b>	[M261–M264]
<b>7</b>	[M171–M174]	<b>14</b>	[M205–M206]	<b>21</b>	[M237–M240]	<b>28</b>	[M265–M272]

**Fig. 20.16** The Pareto front of the transmission tower and the best solutions according to nine different scenarios

the Pareto front. After computing the Pareto front, the engineers involved in the design who should make a decision express their preferences about different criteria (objectives or other independent criteria). By aggregating different ideas, the final solution is selected by an algorithm called MTDM. For instance, comparison between best solutions according to five different scenarios obtained by the present work and CSS-MOPSO is shown in Fig. 20.17 for Example 5.

**Table 20.9** Different possible scenarios for the transmission tower with corresponding solutions

Scenario	Possible priority weights	Selected solutions by MTDM (m, m <sup>3</sup> )	$R_i = \sqrt{f_1^{w_1} \times f_2^{w_2}}$
A	[0.9,0.1]	[0.4571, 0.1102]	0.6296
B	[0.8,0.2]	[0.4793, 0.0947]	0.5887
C	[0.7,0.3]	[0.5067, 0.0763]	0.5359
D	[0.6,0.4]	[0.5646, 0.0606]	0.4809
E	[0.5,0.5]	[0.8278, 0.0504]	0.4520
F	[0.4,0.6]	[1.4683, 0.0439]	0.4228
G	[0.3,0.7]	[2.1702, 0.0398]	0.3633
H	[0.2,0.8]	[3.0858, 0.0384]	0.3039
I	[0.1,0.9]	[4.0674, 0.0377]	0.2453

**Fig. 20.17** Comparison between best solutions according to five different scenarios obtained by the present work and CSS–MOPSO for Example 5

## References

1. Kaveh A, Massoudi MS (2014) Multi-objective optimization using charged system search. *Scientia Iranica* 21(6):1845–1860
2. Dorigo M, Maniezzo V, Colomi A (1996) The ant system: optimization by a colony of cooperating agents. *IEEE Trans Syst Man Cybern* 26:29–41
3. Erol OK, Eksin I (2006) New optimization method: Big Bang–Big Crunch. *Adv Eng Softw* 37:106–111
4. Lee KS, Geem ZW (2004) A new structural optimization method based on the harmony search algorithm. *Comput Struct* 82:781–798
5. Kaveh A, Talatahari S (2010) A novel heuristic optimization method: charged system search. *Acta Mech* 213:267–289
6. Kaveh A, Khayatiazad M (2012) A new metaheuristic method: ray optimization. *Comput Struct* 112–113:283–294

7. Kaveh A, Farhoudi N (2013) A new optimization method: dolphin echolocation. *Adv Eng Softw* 59:53–70
8. 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
9. Zitzler E, Laumanns M, Thiele L (2001) SPEA2: improving the strength Pareto evolutionary algorithm. Swiss Federal Institute Technology, Zurich
10. Knowles JD, Corne DW (2000) Approximating the non-dominated front using the Pareto archived evolution strategy. *Evol Comput* 8:149–172
11. Coello Coello C, Lechuga M (2002) MOPSO: a proposal for multiple objective particle swarm optimization. *Proc Congr Evol Comput* 1:1051–1056
12. Kaveh A, Laknejadi K (2011) A hybrid multi-objective optimization and decision making procedure for optimal design of truss structures. *Iran J Sci Technol Trans Civil Eng* 35:137–154
13. Kaveh A, Talatahari S (2012) Charged system search for optimal design of planar frame structures. *Appl Soft Comput* 12:382–393
14. Deb K (2001) Multi objective optimization using evolutionary algorithms. Wiley, Chichester
15. 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
16. Mostaghim S, Teich J (2003) Strategies for finding good local guides in multi objective particle swarm optimization (MOPSO). In: Proceedings of the IEEE swarm intelligence symposium, pp 26–33
17. Fishburn PC (1970) Utility theory for decision making. Wiley, New York
18. Parreiras RO, Maciel JHRD, Vasconcelos JA (2005) Decision making in multi-objective optimization problems. ISE book series on real word multi-objective system engineering. pp 29–52
19. Palli N, Azram S, McCluskey P, Sundararajan R (1999) An interactive multistage  $\epsilon$ -inequality constraint method for multiple objectives decision making. *ASME J Mech Des* 4:678–686
20. Janga Reddy M, Nagesh Kumar D (2007) An efficient multi-objective optimization algorithm based on swarm intelligence for engineering design. *Eng Optim* 39:49–68
21. El-Santawy MF, Ahmed AN (2012) A multi-objective chaotic harmony search for structural optimization. *Int J Comput Sci* 3:33–39
22. Kelesoglu O (2007) Fuzzy multi-objective optimization of truss-structures using genetic algorithm. *Adv Eng Softw* 38:717–721