Studies in Computational Intelligence 585

Xin-She Yang *Editor*

Recent Advances in Swarm Intelligence and Evolutionary Computation

Springer

Studies in Computational Intelligence

Volume 585

Series editor

Janusz Kacprzyk, Polish Academy of Sciences, Warsaw, Poland
e-mail: kacprzyk@ibspan.waw.pl

About this Series

The series "Studies in Computational Intelligence" (SCI) publishes new developments and advances in the various areas of computational intelligence—quickly and with a high quality. The intent is to cover the theory, applications, and design methods of computational intelligence, as embedded in the fields of engineering, computer science, physics and life sciences, as well as the methodologies behind them. The series contains monographs, lecture notes and edited volumes in computational intelligence spanning the areas of neural networks, connectionist systems, genetic algorithms, evolutionary computation, artificial intelligence, cellular automata, self-organizing systems, soft computing, fuzzy systems, and hybrid intelligent systems. Of particular value to both the contributors and the readership are the short publication timeframe and the world-wide distribution, which enable both wide and rapid dissemination of research output.

More information about this series at http://www.springer.com/series/7092

Xin-She Yang
Editor

Recent Advances in Swarm Intelligence and Evolutionary Computation

Springer

Editor
Xin-She Yang
School of Science and Technology
Middlesex University
London
UK

ISSN 1860-949X ISSN 1860-9503 (electronic)
Studies in Computational Intelligence
ISBN 978-3-319-13825-1 ISBN 978-3-319-13826-8 (eBook)
DOI 10.1007/978-3-319-13826-8

Library of Congress Control Number: 2014956560

Springer Cham Heidelberg New York Dordrecht London

Printed on acid-free paper

Springer International Publishing AG Switzerland is part of Springer Science+Business Media (www.springer.com)

Preface

In many applications, we have to deal with complex optimization problems with complicated constraints. Such problems can be challenging to solve, due to their complexity, nonlinearity and potentially high-dimensionality. These problems can even be NP-hard, and thus require alternative solution methods because conventional methods usually cannot deal with such complex problems. In recent years, nature-inspired metaheuristic algorithms have gained huge popularity because they have demonstrated some promising results in solve tough optimization problems. These metaheuristic algorithms include ant colony optimization, particle swarm optimization, cuckoo search, firefly algorithm, bat algorithm, flower pollination algorithm, bee algorithms and others. There are many reasons for such popularity. In general, these algorithms tend to be flexible, efficient and highly adaptable, and yet easy to implement. The high efficiency of these algorithms makes it possible to apply them to a wide range of problems in diverse applications.

Swarm intelligence is quite a general concept in that multiple agents interact and exchange information, following simple rules. Rather surprisingly, such simple systems can show complex, self-organized behaviour. Though the characteristics of agent interactions may be drawn from different sources of inspiration in nature, algorithmic procedures can be quite simple and flexible, and yet efficient in practice. On the other hand, evolutionary computation is traditionally considered as part of computational intelligence, which concerns optimization with continuous, combinatorial or mixed problems. Algorithms such as genetic algorithms and evolutionary strategy are good examples of evolutionary computation. However, evolutionary computation has broadened its scope and extended to include many areas. Loosely speaking, swarm intelligence is part of the evolutionary computation paradigm, but the interests in swarm intelligence are so overwhelming that swarm intelligence has almost become a field by itself. Here, we will not debate on what the right terminology or fields should be. We discuss both swarm intelligence and evolutionary computation in the broadest sense in this book.

The rapid advances in swarm intelligence and evolutionary computation have resulted in a much richer literature. This timely review volume summarizes the state-of-the-art developments in nature-inspired algorithms and applications with

the emphasis on swarm intelligence and bio-inspired computation. Topics include the analysis and overview of swarm intelligence and evolutionary computation, hybrid metaheuristic algorithms, bat algorithm, discrete cuckoo search, firefly algorithm, particle swarm optimization, and harmony search as well as convergent hybridization. Application case studies have focused on the feature selection by the binary flower pollination algorithm, dehydration of fruits and vegetables by the firefly algorithm and goal programming, job shop scheduling, single row facility layout optimization, training of feed-forward neural networks, damage and stiffness identification, synthesis of cross-ambiguity functions by the bat algorithm, web document clustering, truss analysis, water distribution networks, sustainable building designs and others.

As a timely review, this book can serve as an ideal reference for graduates, lecturers, engineers and researchers in computer science, evolutionary computing, artificial intelligence, machine learning, computational intelligence, data mining, engineering optimization and designs.

I would like to thank our editors, Drs. Thomas Ditzinger and Holger Schaepe, and the staff at Springer for their help and professionalism. Last but not least, I thank my family for the help and support.

London, September 2014 Xin-She Yang

Contents

Contributors

Belaïd Ahiod LRIT, Associated Unit to the CNRST (URAC 29), Mohammed V-Agdal University, Rabat, Morocco

Adis Alihodzic University of Sarajevo, Sarajevo, Bosnia and Herzegovina

Somayeh Asadi Department of Architectural Engineering, Pennsylvania State University, University Park, PA, USA

M. Tamer Ayvaz Department of Civil Engineering, Pamukkale University, Denizli, Turkey

Nebojsa Bacanin Megatrend University Belgrade, Belgrade, Serbia

Sinem Büyüksaatçı Faculty of Engineering, Department of Industrial Engineering, Istanbul University, İstanbul, Turkey

E.F. Campana CNR-INSEAN, Rome, Italy

Sara Casciati Department of Civil Engineering and Architecture, University of Catania, Siracusa, Italy

Shi Cheng Division of Computer Science, The University of Nottingham, Ningbo, Zhejiang Province, China

André Nunes de Souza Department of Electrical Engineering, UNESP, Bauru, SP, Brazil

M. Diez CNR-INSEAN, Rome, Italy

Lorenzo Elia Department of Civil Engineering and Architecture, University of Pavia, Pavia, Italy

G. Fasano Department of Management, University Ca' Foscari of Venice, Venice, Italy

Zong Woo Geem Department of Energy IT, Gachon University, Seongnam, South Korea

M. Geetha Department of Mathematics, Kamaraj College of Engineering and Technology, Virudhunagar, Tamilnadu, India

Şerife Gökçe Department of Civil Engineering, Afyon Kocatepe University, Afyonkarahisar, Turkey

Xingshi He College of Science, Xi'an Polytechnic University, Xi'an, People's Republic of China

Kaizhu Huang Department of Electrical and Electronic Engineering, Xi'an Jiaotong-Liverpool University, Suzhou, Jiangsu Province, China

U. Iemma Department of Engineering, Roma Tre University, Rome, Italy

Raha Imanirad Technology and Operations Management, Harvard Business School, Boston, MA, USA

Momin Jamil Automotive Division, Harman International, Karlsbad, Germany; Blekinge Institute of Technology, Karlskrona, Sweden

M.K. Marichelvam Department of Mechanical Engineering, Mepco Schlenk Engineering College, Sivakasi, Tamilnadu, India

Ei Ei Mon University of Technology (Yatanarpon Cyber City), Pyin Oo Lwin, Myanmar

Aziz Ouaarab LRIT, Associated Unit to the CNRST (URAC 29), Mohammed V-Agdal University, Rabat, Morocco

João Paulo Papa Department of Computing, UNESP, Bauru, SP, Brazil

D. Peri CNR-IAC, Rome, Italy

T. Prabaharan Department of Mechanical Engineering, Mepco Schlenk Engineering College, Sivakasi, Tamilnadu, India

Douglas Rodrigues Department of Computing, UNESP, Bauru, SP, Brazil

A. Serani CNR-INSEAN, Rome, Italy; Department of Engineering, Roma Tre University, Rome, Italy

Rasim Temür Department of Civil Engineering, Faculty of Engineering, Istanbul University, Istanbul, Turkey

T.O. Ting Department of Electrical and Electronic Engineering, Xi'an Jiaotong-Liverpool University, Suzhou, Jiangsu Province, China

Yusuf Cengiz Toklu Department of Civil Engineering, Faculty of Engineering, Bilecik Seyh Edebali University, Bilecik, Turkey

Milan Tuba Megatrend University Belgrade, Belgrade, Serbia

Yusuf Sait Türkan Open and Distance Education Faculty, Istanbul University, Istanbul, Turkey

Xin-She Yang School of Science and Technology, Middlesex University, London, UK

Julian Scott Yeomans OMIS Area, Schulich School of Business, York University, Toronto, ON, Canada

Moe Moe Zaw University of Technology (Yatanarpon Cyber City), Pyin Oo Lwin, Myanmar

Hans-Jürgen Zepernick Blekinge Institute of Technology, Karlskrona, Sweden

Swarm Intelligence and Evolutionary Computation: Overview and Analysis

Xin-She Yang and Xingshi He

Abstract In many applications, the complexity and nonlinearity of the problems require novel and alternative approaches to problem solving. In recent years, nature-inspired algorithms, especially those based on swarm intelligence, have become popular, due to the simplicity and flexibility of such algorithms. Here, we review briefly some recent algorithms and then outline the self-tuning framework for parameter tuning. We also discuss some convergence properties of the cuckoo search and the bat algorithm. Finally, we present some open problems as further research topics.

Keywords Algorithm · Adaptation · Bat algorithm · Cuckoo search · Diversity · Firefly algorithm · Metaheuristic · Nature-inspired algorithm · Optimization · Parameter tuning · Swarm intelligence

1 Introduction

In many applications, we have to deal with complex optimization problems with complicated constraints. Such problems can be challenging to solve, due to their complexity, nonlinearity and potentially high-dimensionality. These problems can even be NP-hard, and thus require alternative methods because conventional methods usually cannot cope such complex problems. In recent years, nature-inspired metaheuristic algorithms have gained huge popularity because they have demonstrated some promising results in solve tough optimization problems. These metaheuristic algorithms include ant colony optimization, particle swarm optimization, cuckoo

X.-S. Yang (✉)
School of Science and Technology, Middlesex University, London NW4 4BT, UK
e-mail: x.yang@mdx.ac.uk; xy227@cam.ac.uk

X. He
College of Science, Xi'an Polytechnic University, No. 19 Jinhua South Road, Xi'an, People's Republic of China

X.-S. Yang (ed.), *Recent Advances in Swarm Intelligence and Evolutionary Computation*, Studies in Computational Intelligence 585, DOI 10.1007/978-3-319-13826-8_1

search, firefly algorithm, bat algorithm, bee algorithms and others [1–4]. There are many reasons for such popularity. From the algorithm analysis point of view, these algorithms tend to be flexible, efficient and highly adaptable, and yet easy to implement. The high efficiency of these algorithms makes it possible to apply them to a wide range of problems in diverse applications.

Swarm intelligence is quite a general concept that multiple agents interact and exchange information, following simple rules. Rather surprisingly, such simple systems can show complex, self-organized behaviour. Though the characteristics of agent interactions may be drawn from different sources of inspiration in nature [4], algorithmic procedures can be quite simple and flexible, and yet efficient in practice. On the other hand, evolutionary computation is traditionally considered as part of computational intelligence, which concerns optimization with continuous, combinatorial or mixed problems. Algorithms such as genetic algorithms and evolutionary strategy are good examples of evolutionary computation. However, evolutionary computation has broaden its scope and extended to include many areas. Loosely speaking, swarm intelligence is part of the evolutionary computation paradigm, but the interests in swarm intelligence are so overwhelming that swarm intelligence has almost become a field of itself. Here, we will not debate what the right terminology or fields should be. We will discuss both swarm intelligence and evolutionary computation in the most broad sense.

The main purpose of this chapter is to provide an overview and the recent advances concerning swarm intelligence and evolutionary computation. Therefore, the chapter is organized as follows. Section 2 outlines some recent nature-inspired algorithms, followed by the analysis and discussions of adaptation and diversity in these algorithms in Sect. 3. Section 4 discusses the self-tuning framework for parameter tuning and control, while Sect. 5 outlines the convergence analysis of the cuckoo search and the bat algorithm. Finally some discussions and open problems are presented in Sect. 6.

2 Swarm Intelligence, Adaptation and Diversity

Extensive research activities have resulted in significant developments in swarm intelligence (SI) in recent years. It is not possible to cover even a good fraction of the extensive literature in this brief review and thus we have to focus on the most recent algorithms, especially those in the last few years.

2.1 The Essence of an Algorithm

Different disciplines may view an algorithm differently, and the point of view all depends on the perspective. In essence, algorithm A is an iterative process, which

aims to generate a new and better solution $\boldsymbol{x}^{t+1}$ to a given problem from the current solution $\boldsymbol{x}^t$ at iteration or (pseudo)time t. It can be written as

$$\boldsymbol{x}^{t+1} = A(\boldsymbol{x}^t, p), \tag{1}$$

where p is an algorithm-dependent parameter. For example, the Newton-Raphson method to find the optimal value of $f(\boldsymbol{x})$ is equivalent to finding the critical points or roots of $f'(\boldsymbol{x}) = 0$ in a d-dimensional space. That is,

$$\boldsymbol{x}^{t+1} = \boldsymbol{x}^t - \frac{f'(\boldsymbol{x}^t)}{f''(\boldsymbol{x}^t)} = A(\boldsymbol{x}^t). \tag{2}$$

Obviously, the convergence rate may become very slow near the optimal point where $f'(x) \to 0$. Sometimes, the true convergence rate may not be as quick as it should be. A simple way to improve the convergence is to modify the above formula slightly by introducing a parameter p as follows:

$$\boldsymbol{x}^{t+1} = \boldsymbol{x}^t - p\frac{f'(\boldsymbol{x}^t)}{f''(\boldsymbol{x}^t)}, \quad p = \frac{1}{1 - A'(\boldsymbol{x}_*)}. \tag{3}$$

Here, $\boldsymbol{x}_*$ is the optimal solution, or a fixed point of the iterative formula. For simplicity, we can treat p as a step size and this essentially becomes the modified Newton-Raphson method.

The above formula is for a trajectory-based, single agent system. For population-based algorithms with a swarm of n solutions $(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n)$, we can extend the above iterative formula to a more general form

$$\begin{pmatrix} \boldsymbol{x}_1 \\ x_2 \\ \vdots \\ \boldsymbol{x}_n \end{pmatrix}^{t+1} = A\left((\boldsymbol{x}_1^t, \boldsymbol{x}_2^t, \ldots, \boldsymbol{x}_n^t); (p_1, p_2, \ldots, p_k); (\epsilon_1, \epsilon_2, \ldots, \epsilon_m)\right) \begin{pmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \\ \vdots \\ \boldsymbol{x}_n \end{pmatrix}^t, \tag{4}$$

where $p_1, \ldots, p_k$ are k algorithm-dependent parameters and $\epsilon_1, \ldots, \epsilon_m$ are m random variables. An algorithm can be viewed as a dynamical system, Markov chains and iterative maps [4], and it can also be viewed as a self-organized system [5]. Here, the introduction of m random variables captures the essence of all contemporary evolutionary algorithms because they all use some sort of randomization techniques.

Whatever the perspective may be, the aim of such an iterative process is to let the system evolve and converge into some stable optimality. In this case, it has strong similarity to a self-organizing system. Such an iterative, self-organizing system can evolve, according to a set of rules or mathematical equations. As a result, such a complex system can interact and self-organize into certain converged states, showing some emergent characteristics of self-organization. In this sense, the

proper design of an efficient optimization algorithm is equivalent to finding efficient ways to mimic the evolution of a self-organizing system [5]. In practice, all nature-inspired algorithms try to mimic some successful characteristics of biological, physical or chemical systems in nature [1, 4, 6].

Among all evolutionary algorithms, algorithms based on swarm intelligence (SI) dominate the landscape. There are many reasons for this dominance, though three obvious reasons are: (1) swarm intelligence uses multiple agents as an evolving, interacting population, and thus provides good ways to mimic natural systems. (2) Population-based approaches allow parallelization and vectorization implementations in practice, and are thus straightforward to implement. (3) SI-based algorithms are simple and easy to implement, and they are flexible and yet sufficiently efficient. As a result, these algorithms can deal with a relatively wide range of problems in applications.

2.2 Particle Swarm Optimization

Among the swarm intelligence (SI) based algorithms, particle swarm optimization (PSO) is among the first. PSO was developed by Kennedy and Eberhart in 1995 [1], based on the swarm behaviour of fish or bird schooling in nature. Each particle updates its position $\boldsymbol{x}_i$ and velocity $\boldsymbol{v}_i$ by

$$\boldsymbol{v}_i^{t+1} = \boldsymbol{v}_i^t + \alpha\,\boldsymbol{\epsilon}_1[\boldsymbol{g}^* - \boldsymbol{x}_i^t] + \beta\boldsymbol{\epsilon}_2[\boldsymbol{x}_i^* - \boldsymbol{x}_i^t], \tag{5}$$

$$\boldsymbol{x}_i^{t+1} = \boldsymbol{x}_i^t + \boldsymbol{v}_i^{t+1}, \tag{6}$$

where $\boldsymbol{\epsilon}_1$ and $\boldsymbol{\epsilon}_2$ are two random vectors, drawn from a uniform distribution between 0 and 1. Here, α and β are the learning parameters with typical values of $\alpha \approx \beta \approx 2$.

The literature of PSO is vast, with thousands of papers and dozens of books. Therefore, we will not provide detailed literature review here, and readers can find such literature quite easily.

2.3 Some Recent SI-Based Algorithms

In this section, we will focus on the SI-based optimization algorithms that have been developed in recent years and these new algorithms have attracted much attention in the last few years.

2.3.1 Firefly Algorithm

The firefly algorithm (FA) is simple, flexible and easy to implement. FA was developed by Yang in 2008 [2], which was based on the flashing patterns and behaviour of tropical fireflies. FA can naturally deal with nonlinear multimodal optimization problems.

The movement of a firefly i is attracted to another more attractive (brighter) firefly j is determined by

$$\boldsymbol{x}_i^{t+1} = \boldsymbol{x}_i^t + \beta_0 e^{-\gamma r_{ij}^2}(\boldsymbol{x}_j^t - \boldsymbol{x}_i^t) + \alpha\, \boldsymbol{\epsilon}_i^t, \tag{7}$$

where the second term is due to the attraction of fireflies, and β_0 is the attractiveness at $r = 0$. The third term is randomization with α being the randomization parameter, and $\boldsymbol{\epsilon}_i^t$ is a vector of random numbers drawn from a Gaussian distribution at time t. Other studies also use the randomization in terms of ϵ_i^t that can easily be extended to other distributions such as Lévy flights. A comprehensive review of the firefly algorithm and its variants has been carried out by Fister et al. [7–9].

One novel feature of FA is that distance-related attraction is used, and this is the first of its kind in any SI-based algorithms. Since local attraction is stronger than long-distance attraction, the population in FA can automatically subdivide into multiple subgroups, and each group can potentially swarm around a local mode. Among all the local modes, there is always a global best solution which is the true optimality of the problem. Thus, FA can deal with multimodal problems naturally and efficiently [3].

2.3.2 Cuckoo Search

The cuckoo search (CS) was developed in 2009 by Yang and Deb [10]. CS is based on the brood parasitism of some cuckoo species. In addition, this algorithm is enhanced by the so-called Lévy flights [11], rather than by simple isotropic random walks. Recent studies show that CS is potentially far more efficient than PSO and genetic algorithms [12–14].

In essence, CS uses a balanced combination of a local random walk and the global explorative random walk, controlled by a switching parameter p_a. The local random walk can be written as

$$\boldsymbol{x}_i^{t+1} = \boldsymbol{x}_i^t + \alpha s \otimes H(p_a - \epsilon) \otimes (\boldsymbol{x}_j^t - \boldsymbol{x}_k^t), \tag{8}$$

where $\boldsymbol{x}_j^t$ and $\boldsymbol{x}_k^t$ are two different solutions selected randomly by random permutation, $H(u)$ is a Heaviside function, ϵ is a random number drawn from a uniform distribution, and s is the step size. On the other hand, the global random walk is carried out by using Lévy flights

$$\boldsymbol{x}_i^{t+1} = \boldsymbol{x}_i^t + \alpha L(s, \lambda), \tag{9}$$

where

$$L(s, \lambda) = \frac{\lambda \Gamma(\lambda) \sin(\pi\lambda/2)}{\pi} \frac{1}{s^{1+\lambda}}, \quad (s \gg s_0 > 0). \tag{10}$$

Here $\alpha > 0$ is the step size scaling factor, which should be related to the scales of the problem of interest.

CS has two distinct advantages over other algorithms such as GA and SA, and these advantages are: efficient random walks and balanced mixing. Since Lévy flights are usually far more efficient than any other random-walk-based randomization techniques, CS can be efficient in global search [3, 11]. In fact, recent studies show that CS can have guaranteed global convergence [4]. In addition, the similarity between eggs can produce better new solutions, which is essentially fitness-proportional generation with a good mixing ability. In other words, CS has varying mutation realized by Lévy flights, and the fitness-proportional generation of new solutions based on similarity provides a subtle form of crossover. In addition, simulations also show that CS can have autozooming ability in the sense that new solutions can automatically zoom into the region where the promising global optimality is located.

In addition, Eq. (9) is essentially simulated annealing in the framework of Markov chains. In Eq. (8), if $p_a = 1$ and $\alpha s \in [0, 1]$, CS can degenerate into a variant of differentia evolution. Furthermore, if we replace $\boldsymbol{x}_j^t$ by the current best solution $\boldsymbol{g}^*$, then (8) can further degenerate into accelerated particle swarm optimization (APSO) [15]. This means that SA, DE and APSO are special cases of CS, and this explains why CS is so efficient [3].

2.3.3 Bat Algorithm

The bat algorithm (BA) was developed by Yang in 2010 [16]. It was inspired by the echolocation behavior of microbats. It is the first algorithm of its kind to use frequency tuning. Each bat is associated with a velocity $\boldsymbol{v}_i^t$ and a location $\boldsymbol{x}_i^t$, at iteration t, in a d-dimensional search or solution space. Among all the bats, there exists a current best solution $\boldsymbol{x}_*$. Therefore, the updating equations for $\boldsymbol{x}_i^t$ and velocities $\boldsymbol{v}_i^t$ can be written as

$$f_i = f_{\min} + (f_{\max} - f_{\min})\beta, \tag{11}$$

$$\boldsymbol{v}_i^t = \boldsymbol{v}_i^{t-1} + (\boldsymbol{x}_i^{t-1} - \boldsymbol{x}_*)f_i, \tag{12}$$

$$\boldsymbol{x}_i^t = \boldsymbol{x}_i^{t-1} + \boldsymbol{v}_i^t, \tag{13}$$

where $\beta \in [0, 1]$ is a random vector drawn from a uniform distribution.

The loudness and pulse emission rates are regulated by the following equations:

$$A_i^{t+1} = \alpha A_i^t, \tag{14}$$

and

$$r_i^{t+1} = r_i^0[1 - \exp(-\gamma t)], \tag{15}$$

where $0 < \alpha < 1$ and $\gamma > 0$ are constants. In essence, here α is similar to the cooling factor of a cooling schedule in simulated annealing.

BA has been extended to multiobjective bat algorithm (MOBA) by Yang [17], and Fister et al. have extended to a hybrid bat algorithm [18]. The preliminary results suggested that they are very efficient [19, 20].

2.3.4 Flower Algorithm

Flower pollination algorithm (FPA) was developed by Yang in 2012 [21], inspired by the flower pollination process of flowering plants. It has been extended to multiobjective optimization problems and found to be very efficient [22, 23]. For simplicity, we use the following four rules:

1. Biotic and cross-pollination can be considered as a process of global pollination process, and pollen-carrying pollinators move in a way which obeys Lévy flights (Rule 1).
2. For local pollination, abiotic pollination and self-pollination are used (Rule 2).
3. Pollinators such as insects can develop flower constancy, which is equivalent to a reproduction probability that is proportional to the similarity of two flowers involved (Rule 3).
4. The interaction or switching of local pollination and global pollination can be controlled by a switch probability $p \in [0, 1]$, with a slight bias towards local pollination (Rule 4).

In order to formulate updating formulae, we have to convert the above rules into updating equations. For example, in the global pollination step, flower pollen gametes are carried by pollinators such as insects, and pollen can travel over a long distance because insects can often fly and move in a much longer range. Therefore, Rule 1 and flower constancy can be represented mathematically as

$$\boldsymbol{x}_i^{t+1} = \boldsymbol{x}_i^t + \gamma L(\lambda)(\boldsymbol{g}_* - \boldsymbol{x}_i^t), \tag{16}$$

where $\boldsymbol{x}_i^t$ is the pollen i or solution vector $\boldsymbol{x}_i$ at iteration t, and $\boldsymbol{g}_*$ is the current best solution found among all solutions at the current generation/iteration. Here γ is a scaling factor to control the step size.

Here $L(\lambda)$ is the parameter that corresponds to the strength of the pollination, which essentially is also a step size. Since insects may move over a long distance with various distance steps, we can use a Lévy flight to mimic this characteristic efficiently. That is, we draw $L > 0$ from a Levy distribution

$$L \sim \frac{\lambda \Gamma(\lambda) \sin(\pi\lambda/2)}{\pi} \frac{1}{s^{1+\lambda}}, \quad (s \gg s_0 > 0). \tag{17}$$

Here $\Gamma(\lambda)$ is the standard gamma function, and this distribution is valid for large steps $s > 0$. This step is essentially a global mutation step, which enables to explore the search space more efficiently.

For the local pollination, both Rule 2 and Rule 3 can be represented as

$$\boldsymbol{x}_i^{t+1} = \boldsymbol{x}_i^t + \epsilon(\boldsymbol{x}_j^t - \boldsymbol{x}_k^t), \tag{18}$$

where $\boldsymbol{x}_j^t$ and $\boldsymbol{x}_k^t$ are pollen from different flowers of the same plant species. This essentially mimics the flower constancy in a limited neighborhood. Mathematically, if $\boldsymbol{x}_j^t$ and $\boldsymbol{x}_k^t$ comes from the same species or selected from the same population, this equivalently becomes a local random walk if we draw ϵ from a uniform distribution in [0, 1]. In essence, this is a local mutation and mixing step, which can help to converge in a subspace.

In principle, flower pollination activities can occur at all scales, both local and global. But in reality, adjacent flower patches or flowers in the not-so-far-away neighborhood are more likely to be pollinated by local flower pollen than those far away. In order to mimic this feature, we can effectively use a switch probability (Rule 4) or proximity probability p to switch between common global pollination to intensive local pollination. To start with, we can use a naive value of $p = 0.5$ as an initially value. A preliminary parametric showed that $p = 0.8$ may work better for most applications.

Recent studies suggested that flower pollination algorithm is very efficient for multiobjective optimization [23, 24].

2.4 Other Evolutionary Algorithms

There are quite a few other evolutionary algorithms, though they are not swarm intelligence based algorithms. So we also briefly introduce them here.

2.4.1 Differential Evolution

Differential evolution (DE) was developed by Storn and Price in 1996 and 1997 [25, 26]. In fact, modern differential evolution (DE) has strong similarity to the traditional mutation operator in the traditional pattern search. In essence, the mutation in DE can be viewed as the generalized pattern search in any random direction $(\boldsymbol{x}_p - \boldsymbol{x}_q)$ by

$$\boldsymbol{x}_i = \boldsymbol{x}_r + F(\boldsymbol{x}_p - \boldsymbol{x}_q), \tag{19}$$

where F is the differential weight in the range of [0,2]. Here, r, p, q, i are four different integers generated by random permutation.

In addition, DE also has a crossover operator which is controlled by a crossover probability $C_r \in [0,\ 1]$ and the actual crossover can be carried out in two ways: binomial and exponential. Selection is essentially the same as that used in genetic algorithms. It is to select the most fittest, and for the minimization problem, the minimum objective value. Therefore, we have

$$\boldsymbol{x}_i^{t+1} = \begin{cases} \boldsymbol{u}_i^{t+1} & \text{if } f(\boldsymbol{u}_i^{t+1}) \le f(\boldsymbol{x}_i^t), \\ \boldsymbol{x}_i^t & \text{otherwise.} \end{cases} \tag{20}$$

Most studies have focused on the choice of F, C_r, and the population size n as well as the modification of the mutation scheme. In addition, it can be clearly seen that selection is also used when the condition in the above equation is checked. Almost all variants of DE use crossover, mutation and selection, and the main differences are in the step of mutation and crossover. For example, DE/Rand/1/Bin use the three vectors for mutation, and binomial crossover. There are more than 10 different variants [27].

2.4.2 Harmony Search

Harmony search (HS) is a music-inspired algorithm, developed by Geem et al. in 2001 [28]. It is not swarm-intelligence-based, but it is a metaheuristic algorithm. In the standard HS, solutions are represented in terms of a population of harmonies, using the following three choices/rules (of a musician playing a piece of music): (1) play any famous piece of music from memory, (2) play something similar to a known piece (pitch adjustment), and (3) compose new or random notes. HS uses mainly mutation and selection, while crossover is not explicitly used. The first rule corresponds to selection or elitism, and the second and third rules are mutation.

Mutation can be local and global in HS. For example, the pitch adjustment (the second rule) uses the following equation

$$\boldsymbol{x}_{new} = \boldsymbol{x}_{old} + b_w \varepsilon, \tag{21}$$

where b_w is the bandwidth of the pitch adjustment, while ε is a random number drawn from $[-1, 1]$. This is a local random walk, and the distance of the random walk is controlled by the bandwidth. This part can be considered as a local mutation action with an equivalent mutation rate of 0.1–0.3.

The third rule is essentially mutation on a larger scale, which is essentially equivalent to random walks. The selection is controlled by the probability of choosing a harmony from harmony memory. Similar to genetic algorithms, this choice of harmonies from the population is high with a typical value of 0.9, which enables the system to converge in a subspace. However, this may be at the expense of reduced probability of finding the global optimality in some highly nonlinear problems.

2.4.3 Other Algorithms

Many other algorithms have appeared in the literature, which may require more extensive analysis and comparisons [4]. However, as this is not the main focus of this chapter, we will not go into more details about these algorithms.

One thing we may notice by analyzing these algorithms is that mutation and selection are always used, while crossover are not used in most of these algorithms. This may raise the question and further need to analyze what exactly the role of crossover is. Therefore, there is a strong need to investigate further how two-stage eagle strategy and co-evolutionary methods can work better. In addition, systematic tuning of parameters in algorithms and careful control of these algorithm-dependent parameters may be very useful to understand how these algorithms behave and how to improve them in practice.

3 Adaptation and Diversity

The effectiveness of these algorithms can be attributed to two important characteristics: adaptation and diversity of nature-inspired optimization algorithms.

Adaptation in nature-inspired algorithms can take many forms. For example, the ways to balance exploration and exploitation are the key form of adaptation [29]. As diversity can be intrinsically linked with adaptation, it is better not to discuss these two features separately. If exploitation is strong, the search process will use problem-specific information (or landscape-specific information) obtained during the iterative process to guide the new search moves, this may lead to the focused search and thus reduce the diversity of the population, which may help to speed up the convergence of the search procedure. However, if exploitation is too strong, it can result in the quick loss of diversity in the population and thus may lead to the

premature convergence. On the other hand, if new search moves are not guided by local landscape information, it can typically increase the exploration capability and generate new solutions with higher diversity. However, too much diversity and exploration may result in meandered search paths, thus lead to the slow convergence. Therefore, adaptation of search moves so as to balance exploration and exploitation is crucial. Consequently, to maintain a balanced diversity in population is also important.

Adaptation can also be in terms of the representations of solutions of a problem. In genetic algorithms, representations of solutions are usually in binary or real-valued strings [29, 30], while in swarm-intelligence-based algorithms, representations mostly use real number solution vectors. For example, the population size used in an algorithm can be fixed or varying. Adaptation in this case may mean to vary the population size so as to maximize the overall performance.

For a given algorithm, adaptation can also occur to adjust its algorithm-dependent parameters. As the performance of an algorithm can largely depend on its parameters, the choice of these parameter values can be very important. Values can be varied so as to adapt the landscape type of the problem and thus may lead to better search efficiency. Such parameter tuning is in essence parameter adaptation. Once parameters are tuned, they can remain fixed. However, there is no particular reason why parameters should be fixed. In fact, adaptation in parameter can be extended to parameter control. That is to control the parameter values in such a way that their values vary during the iterations so that optimal performance of the algorithm can be achieved.

Diversity in metaheuristic algorithms can also take many forms. The simplest diversity is to allow the variations of solutions in the population by randomization. For example, solution diversity in genetic algorithms is mainly controlled by mutation rates and crossover mechanisms, while in simulated annealing, diversity is achieved by random walks. In most swarm-intelligence-based algorithms, new solutions are generated according to a set of deterministic equations, which also include some random variables. Diversity is represented by the variation, often in terms of population variance. Once the population variance is getting smaller (approaching zero), diversity also decreases, leading to converged solution sets. However, if diversity is reduced too quickly, premature convergence may occur. Therefore, a right amount of randomness and the right form of randomization can be crucial.

In addition, adaptation and diversity can also be related to the selection of solutions among the population and the replacement of the old population. If the selection is based on the fitness, parent solutions with higher fitness will be more likely to pass onto the next generation. In the extreme case, only the best solutions can be selected, which is a kind of elitism. If the replacement of worst solutions by new (hopefully better) solutions, this will ensure that better solutions will remain in the population. The balance of what to replace and what to pass on can be tricky, which requires good adaptation so as to maintain good diversity in the population.

4 Self-Tuning Algorithms

From a mathematical point of view, an algorithm A tends to generate a new and better solution $\boldsymbol{x}^{t+1}$ to a given problem from the current solution $\boldsymbol{x}^t$ at iteration or time t. In modern metaheuristic algorithms, randomization is often used in an algorithm, and in many cases, randomization appears in the form of a set of m random variables $\varepsilon = (\varepsilon_1, \ldots, \varepsilon_m)$ in an algorithm. For example, in simulated annealing, there is one random variable, while in particle swarm optimization, there are two random variables. In addition, there are often a set of k parameters in an algorithm. For example, in particle swarm optimization, there are 4 parameters (two learning parameters, one inertia weight, and the population size). In general, we can have a vector of parameters $\boldsymbol{p} = (p_1, \ldots, p_k)$. Mathematically speaking, we can write an algorithm with k parameters and m random variables as

$$\boldsymbol{x}^{t+1} = \boldsymbol{A}(\boldsymbol{x}^t, \boldsymbol{p}(t), \varepsilon(t)), \tag{22}$$

where $\boldsymbol{A}$ is a nonlinear mapping from a given solution (a d-dimensional vector $\boldsymbol{x}^t$) to a new solution vector $\boldsymbol{x}^{t+1}$.

Representation (22) gives rise to two types of optimality: optimality of a problem and optimality of an algorithm [4, 31]. For an optimization problem such as $\min f(\boldsymbol{x})$, there is a global optimal solution whatever the algorithmic tool we may use to find this optimality. This is the optimality for the optimization problem. On the other hand, for a given problem Φ with an objective function $f(\boldsymbol{x})$, there are many algorithms that can solve it. Some algorithms may require less computational effort than others. There may be the best algorithm with the least computing cost, though this may not be unique. However, this is not our concern here. Once we have chosen an algorithm A to solve a problem Φ, there is an optimal parameter setting for this algorithm so that it can achieve the best performance. This optimality depends on both the algorithm itself and the problem it solves. In the rest of this chapter, we will focus on this type of optimality.

That is, the optimality to be achieved is

$$\text{Maximize the performance of } \xi = A(\Phi, \boldsymbol{p}, \varepsilon), \tag{23}$$

for a given problem Φ and a chosen algorithm $A(., \boldsymbol{p}, \varepsilon)$. We will denote this optimality as $\xi_* = A_*(\Phi, \boldsymbol{p}_*) = \xi(\Phi, \boldsymbol{p}_*)$ where $\boldsymbol{p}_*$ is the optimal parameter setting for this algorithm so that its performance is the best. Here, we have used a fact that ε is a random vector can be drawn from some known probability distributions, thus the randomness vector should not be related to the algorithm optimality.

It is worth pointing out that there is another potential optimality. That is, for a given problem, a chosen algorithm with the best parameter setting $\boldsymbol{p}_*$, we can still use different random numbers drawn from various probability distributions and even chaotic maps, so that even better performance may be achieved. Strictly speaking, if an algorithm $A(., ., \varepsilon)$ has a random vector ε that is drawn from a

uniform distribution $\varepsilon_1 \sim U(0,1)$ or from a Gaussian $\varepsilon_2 \sim N(0,1)$, it becomes two algorithms $A_1 = A(.,.,\ \varepsilon_1)$ and $A_2 = A(.,.,\ \varepsilon_2)$. Technically speaking, we should treat them as different algorithms. Since our emphasis here is about parameter tuning so as to find the optimal setting of parameters, we will omit the effect of randomness vectors, and thus focus on

$$\text{Maximize } \xi = A(\Phi, \boldsymbol{p}). \tag{24}$$

In essence, tuning algorithm involves in tuning its algorithm-dependent parameters. Therefore, parameter tuning is equivalent to algorithm tuning in the present context.

4.1 Parameter Tuning

In order to tune $A(\Phi, \boldsymbol{p})$ so as to achieve its best performance, a parameter-tuning tool, i.e., a tuner, is needed. Like tuning a high-precision machinery, sophisticated tools are required. For tuning parameters in an algorithm, what tool can we use? One way is to use a better, existing tool (say, algorithm B) to tune an algorithm A. Now the question may become: how do you know B is better? Is B well-tuned? If yes, how do you tune B in the first place? Naively, if we say, we use another tool (say, algorithm C) to tune B. Now again the question becomes how algorithm C has been tuned? This can go on and on, until the end of a long chain, say, algorithm Q. In the end, we need some tool/algorithm to tune this Q, which again comes back to the original question: how to tune an algorithm A so that it can perform best?

It is worth pointing out that even if we have good tools to tune an algorithm, the best parameter setting and thus performance all depend on the performance measures used in the tuning. Ideally, these parameters should be robust enough to minor parameter changes, random seeds, and even problem instance. However, in practice, they may not be achievable. According to Eiben [32], parameter tuning can be divided into iterative and non-iterative tuners, single-stage and multi-stage tuners. The meaning of these terminologies is self-explanatory. In terms of the actual tuning methods, existing methods include sampling methods, screening methods, model-based methods, and metaheuristic methods. Their success and effectiveness can vary, and thus there are no well-established methods for universal parameter tuning.

4.2 Framework for Self-Tuning Algorithms

From our earlier observations and discussions, it is clear that parameter tuning is the process of optimizing the optimization algorithm; therefore, it is a hyper-optimization problem. In essence, a tuner is a meta-optimization tool for tuning algorithms [31].

For a standard unconstrained optimization problem, the aim is to find the global minimum f_* of a function $f(\boldsymbol{x})$ in a d-dimensional space. That is,

$$\text{Minimize } f(\boldsymbol{x}), \quad \boldsymbol{x} = (x_1, x_2, \ldots, x_d). \tag{25}$$

Once we choose an algorithm A to solve this optimization problem, the algorithm will find a minimum solution $f_{\min}$ which may be close to the true global minimum f_*. For a given tolerance δ, this may requires t_δ iterations to achieve $|f_{\min} - f_*| \leq \delta$. Obviously, the actual t_δ will largely depend on both the problem objective $f(\boldsymbol{x})$ and the parameters $\boldsymbol{p}$ of the algorithm used.

The main aim of algorithm-tuning is to find the best parameter setting $\boldsymbol{p}_*$ so that the computational cost or the number of iterations t_δ is the minimum. Thus, parameter tuning as a hyper-optimization problem can be written as

$$\text{Minimize } t_\delta = A(f(\boldsymbol{x}), \boldsymbol{p}), \tag{26}$$

whose optimality is $\boldsymbol{p}_*$.

Ideally, the parameter vector $\boldsymbol{p}_*$ should be sufficiently robust. For different types of problems, any slight variation in $\boldsymbol{p}_*$ should not affect the performance of A much, which means that $\boldsymbol{p}_*$ should lie in a flat range, rather than at a sharp peak in the parameter landscape.

4.3 A Multiobjective View

If we look the algorithm tuning process from a different perspective, it is possible to construct it as a multi-objective optimization problem with two objectives: one objective $f(\boldsymbol{x})$ for the problem Φ and one objective t_δ for the algorithm. That is

$$\text{Minimize } f(\boldsymbol{x}) \text{ and Minimize } t_\delta = A(f(\boldsymbol{x}), \boldsymbol{p}), \tag{27}$$

where t_δ is the (average) number of iterations needed to achieve a given tolerance δ so that the found minimum $f_{\min}$ is close enough to the true global minimum f_*, satisfying $|f_{\min} - f_*| \leq \delta$.

This means that for a given tolerance δ, there will be a set of best parameter settings with a minimum t_δ. As a result, the bi-objectives will form a Pareto front. In principle, this bi-objective optimization problem (27) can be solved by any methods that are suitable for multiobjective optimization. But as δ is usually given, a natural way to solve this problem is to use the so-called ϵ-constraint or δ-constraint methods. The naming may be dependent on the notations; however, we will use δ-constraints.

For a given $\delta \geq 0$, we change one of the objectives (i.e., $f(\boldsymbol{x})$) into a constraint, and thus the above problem (27) becomes a single-objective optimization problem with a constraint. That is

$$\text{Minimize } t_\delta = A(f(\boldsymbol{x}), \boldsymbol{p}), \tag{28}$$

subject to

$$f(\boldsymbol{x}) \leq \delta. \tag{29}$$

In the rest of this chapter, we will set $\delta = 10^{-5}$.

The important thing is that we still need an algorithm to solve this optimization problem. However, the main difference from a common single objective problem is that the present problem contains an algorithm A. Ideally, an algorithm should be independent of the problem, which treats the objective to be solved as a black box. Thus we have $A(., \boldsymbol{p}, \varepsilon)$, however, in reality, an algorithm will be used to solve a particular problem Φ with an objective $f(\boldsymbol{x})$. Therefore, both notations $A(., \boldsymbol{p})$ and $A(f(\boldsymbol{x}), \boldsymbol{p})$ will be used here.

4.4 Self-Tuning Framework

This framework has been proposed by Yang et al. in 2013 [31]. In principle, we can solve (28) by any efficient or well-tuned algorithm. Now a natural question is: Can we solve this algorithm-tuning problem by the algorithm A itself? There is no reason why we cannot. In fact, if we solve (28) by using A, we have a self-tuning algorithm. That is, the algorithm automatically tunes itself for a given problem objective to be optimized. This essentially provides a framework for a self-tuning algorithm as shown in Fig. 1.

This framework is generic in the sense that any algorithm can be tuned this way, and any problem can be solved within this framework. This essentially achieves two goals simultaneously: parameter tuning and optimality finding.

4.5 Self-Tuning Firefly Algorithm

Now let us use the framework outlined earlier to tune the firefly algorithm (FA). As we have seen earlier, FA has the following updating equation:

$$\boldsymbol{x}_i^{t+1} = \boldsymbol{x}_i^t + \beta_0 e^{-\gamma r_{ij}^2}(\boldsymbol{x}_j^t - \boldsymbol{x}_i^t) + \alpha \boldsymbol{\epsilon}_i^t, \tag{30}$$

which contains four parameters: α, β_0, γ and the population size n. For simplicity for parameter tuning, we set $\beta_0 = 1$ and $n = 20$, and therefore the two parameters to be tuned are: $\gamma > 0$ and $\alpha > 0$. It is worth pointing out that γ controls the scaling, while α controls the randomness. For this algorithm to convergence properly, randomness

Implement an algorithm $A(., \boldsymbol{p}, \boldsymbol{\varepsilon})$
  with parameters $\boldsymbol{p} = [p_1, ..., p_K]$ and random vector $\boldsymbol{\varepsilon} = [\varepsilon_1, ..., \varepsilon_m]$;
Define a tolerance (e.g., $\delta = 10^{-5}$);
  Algorithm objective $t_\delta(f(\boldsymbol{x}), \boldsymbol{p}, \boldsymbol{\varepsilon})$;
    Problem objective function $f(\boldsymbol{x})$;
    Find the optimality solution $f_{\min}$ within δ;
    Output the number of iterations t_δ needed to find $f_{\min}$;
  Solve $\min t_\delta(f(\boldsymbol{x}), \boldsymbol{p})$ using $A(., \boldsymbol{p}, \boldsymbol{\varepsilon})$ to get the best parameters;
Output the tuned algorithm with the best parameter setting $\boldsymbol{p}_*$.

Fig. 1 A framework for a self-tuning algorithm

should be gradually reduced, and one way to achieve such randomness reduction is to use

$$\alpha = \alpha_0 \theta^t, \quad \theta \in (0, 1), \tag{31}$$

where t is the index of iterations/generations. Here α_0 is the initial randomness factor, and we can set $\alpha_0 = 1$ without losing generality. Therefore, the two parameters to be tuned become γ and θ. This framework works wells as shown by Yang et al. [31].

5 Convergence Analysis

There some solid mathematical analysis of nature-inspired algorithms such as differential evolution, genetic algorithms [33], simulated annealing and particle swarm optimization. In recent years, more theoretical results appear, and in the rest of this section, we summarize some of the recent advances.

5.1 Global Convergence of Cuckoo Search

Wang et al. provided a mathematical proof of global convergence for the standard cuckoo search, and their approach is based on the Markov chain theory [34]. Their proof can be outlined as follows:

As there are two branches in the updating formulas, the local search step only contributes mainly to local refinements, while the main mobility or exploration is carried out by the global search step. In order to simplify the analysis and also to emphasize the global search capability, we now use a simplified version of cuckoo search. That is, we use only the global branch with a random number $r \in [0, 1]$, compared with a discovery/switching probability p_a. Now we have

$$\begin{cases} \boldsymbol{x}_i^{(t+1)} \leftarrow x_i^{(t)} & \text{if } r < p_a, \\ \boldsymbol{x}_i^{(t+1)} \leftarrow \boldsymbol{x}_i^{(t)} + \alpha \otimes L(\lambda) & \text{if } r > p_a. \end{cases} \tag{32}$$

As our cuckoo search algorithm is a stochastic search algorithm, we can summarize it as the following key steps:

1. Randomly generate an initial population of n nests at the positions, $\boldsymbol{X} = \{\boldsymbol{x}_1^0, \boldsymbol{x}_2^0, \ldots, \boldsymbol{x}_n^0\}$, then evaluate their objective values so as to find the current global best $\boldsymbol{g}_t^0$.
2. Update the new solutions/positions by

$$\boldsymbol{x}_i^{(t+1)} = \boldsymbol{x}_i^{(t)} + \alpha \otimes L(\lambda). \tag{33}$$

3. Draw a random number r from a uniform distribution [0,1]. Update $\boldsymbol{x}_i^{(t+1)}$ if $r > p_a$. Then, evaluate the new solutions so as to find the new, global best $\boldsymbol{g}_t^*$.
4. If the stopping criterion is met, then $\boldsymbol{g}_t^*$ is the best global solution found so far. Otherwise, return to step (2).

The global convergence of an algorithm. If f is measurable and the feasible solution space Ω is a measurable subset on $\Re^n$, algorithm A satisfies the above two conditions with the search sequence $\{x_k\}_{k=0}^{\infty}$, then

$$\lim_{k \to \infty} P(x_k \in R_{\epsilon,M}) = 1. \tag{34}$$

That is, algorithm A can converge globally with a probability of one. Here $P(x_k \in R_{\epsilon,M})$ is the probability measure of the kth solution on $R_{\epsilon,M}$ at the kth iteration.

The state and state space. The positions of a cuckoo/nest and its global best solution g in the search history forms the states of cuckoos: $y = (x, g)$, where $x, g \in \Omega$ and $f(g) \leq f(x)$. The set of all the possible states forms the state space, denoted by

$$Y = \{y = (x, g) | x, g \in \Omega, f(g) \leq f(x)\}. \tag{35}$$

The states and state space of the cuckoo group/population. The states of all n cuckoos/nests form the states of the group, denoted by $q = (y_1, y_2, \ldots, y_n)$. All the states of all the cuckoos form a state space for the group, denoted by

$$Q = \{q = (y_1, y_2, \ldots, y_n), y_i \in Y, 1 \leq i \leq n\}. \tag{36}$$

Obviously, Q contains the historical global best solution g^* for the whole population and all individual best solutions $g_i (1 \leq i \leq n)$ in history. In addition, the global best solution of the whole population is the best among all g_i, so that $f(g^*) = \min(f(g_i)),\ 1 \leq i \leq n$.

The transition probability from state y_1 to y_2 in cuckoo search is

$$P(T_y(y_1) = y_2) = P(x_1 \to x_1')P(g_1 \to g_1')P(x_1' \to x_2)P(g_1' \to g_2), \tag{37}$$

where $P(x_1 \to x_1')$ is the transition probability at Step 2 in cuckoo search, and $P(g_1 \to g_1')$ is the transition probability for the historical global best at this step. $P(x_1' \to x_2)$ is the transition probability at Step 3, while $P(g_1' \to g_2)$ is the transition probability of the historical global best.

For globally optimal solution g_b for an optimization problem $<\Omega, f>$, the optimal state set is defined as $R = \{y = (x, g) | f(g) = f(g_b), y \in Y\}$.

For the globally optimal solution g_b to an optimization problem $<\Omega, f>$, the optimal group state set can be defined as

$$H = \{q = (y_1, y_2, \ldots, y_n) | \exists y_i \in R, 1 \le i \le n\}. \tag{38}$$

All these will ensure that the convergence conditions are met. Further detailed mathematical analysis proves that when the number of iteration approaches sufficiently large [34], the group state sequence will converge to the optimal state/solution set H. Therefore, the cuckoo search has guaranteed global convergence.

5.2 Convergence of the Bat Algorithm

Huang et al. have carried out a detailed convergence analysis for the bat algorithm using the finite Markov process theory [35].

In theory, an algorithm with an order-m reducible stochastic matrix P can be rewritten as

$$P = \begin{pmatrix} S \ldots 0 \\ R \ldots T \end{pmatrix}, \tag{39}$$

where $R \neq 0$, $T \neq 0$, and S is order-q stochastic matrix (with $q < m$). Then, we have

$$\begin{aligned} P^{\infty} &= \lim_{k \to \infty} P^k \\ &= \lim_{k \to \infty} \begin{pmatrix} S^k & \cdots & 0 \\ \sum_{i=1}^{k-1} T^i R S^{k-i} & \cdots & T^k \end{pmatrix} = \begin{pmatrix} S^{\infty} \ldots 0 \\ R^{\infty} \ldots T \end{pmatrix}, \end{aligned} \tag{40}$$

which is a stable stochastic matrix and independent of the initial distribution. In addition, we also have

$$P^{\infty} = [p_{ij}]_{m\times m}, \quad \begin{cases} p_{ij} > 0, & (1 \le i \le m, 1 \le j \le q), \\ p_{ij} = 0, & (1 \le i \le m, q < j \le m). \end{cases} \tag{41}$$

The search algorithm will converge with almost probability one to the global optimality, starting from any initial random states, if the transition probability p to a better solution/state is $p > 0$. Conversely, if the transition probability p to a worse state is greater, then the algorithm will not converge.

With this main result, it has been proved that PSO will not converge to the global optimality [36], while the bat algorithm will converge to the true global optimality [35].

Huang et al. concluded that for unconstrained function optimization, the bat algorithm satisfies all the conditions for guaranteed global convergence. For nonlinear constrained problems, the bat algorithm will converge with additional initialization of orthogonal Latin squares, and has guaranteed global convergence to the true global optimality. They further concluded that

$$S^{\infty} = (1), \quad R^{\infty} = (1, 1, \ldots, 1)^T, \tag{42}$$

and

$$P^{\infty} = \begin{pmatrix} 1 & 0 & \ldots & 0 \\ 1 & 0 & \ldots & 0 \\ \vdots & \vdots & & \vdots \\ 1 & 0 & \ldots & 0 \end{pmatrix}, \tag{43}$$

which leads to

$$\lim_{t\to\infty} p\{f(\boldsymbol{x}) \to f(\boldsymbol{x}_*)\} = 1. \tag{44}$$

That is, the global convergence is guaranteed.

Huang et al. also proposed a BA variant, called modified bat algorithm (MBA) [35], which can further improve the convergence rate with guaranteed global optimality. They also showed that this variant is suitable for large-scale, global optimization.

For the moment, most convergence studies can provide some results in terms of the long term behaviour of an algorithm during iterations; however, there are not enough results about the convergence rates to indicate how quickly an algorithm can converge and under what conditions. Obviously, more theoretical are highly needed to analyze these algorithms further.

6 Discussions and Open Problems

Despite the huge success of nature-inspired algorithms, there are still some challenging, open problems that need to be addressed. These open problems include the balance of exploration and exploitation, selection mechanisms, right amount of randomization, parameter tuning as well as parameter control, scalability and others.

- **Mathematical Framework**: It still lacks a general mathematical framework for analyzing the convergence and stability of metaheuristic algorithms. There are some good results using Markov chains, dynamic systems and self-organization theory, but a systematic framework is yet to be developed.
- **Exploration and exploitation**: A key problem is how to balance of exploration and exploitation in an algorithm so that it can deal with a vast range of problems efficiently [37]. In reality, the amount of exploration and exploitation may depend on the type of problem, and therefore, some a *priori* knowledge of the problem to be solved can help to determine such a balance. However, it is not known how to incorporate such knowledge effectively. For example, gradient/derivative information obtained from the objective function can be very useful for exploitation, but if such exploitation is too strong, it can cause the system to be trapped in a local optimum, thus sacrificing the possibility of finding the true global optimality. There may not exist such optimal balance for all problems [38].
- **Selection Mechanism**: Selection mechanism is also very important and it is not known what selection is most effective. A proper selection pressure is crucial to maintain a healthy population. For example, when many solutions have similar fitness, numerically speaking, their fitness values may almost be the same, thus how to select certain solutions becomes tricky. Typical approaches include re-scaled fitness values, ranking of solutions, and adaptive elitism. However, it is not clear if they can work for all algorithms and if there is other better ways to handle selection.
- **Right Amount of Randomness**: In order to balance exploration and exploitation, a right amount of randomness is needed. However, no one knows what amount is the right amount. At one extreme, if there is no randomness, an algorithm becomes a deterministic algorithm, and thus loses the ability to explore. At the other extreme, if the search is dominated by high randomness, the algorithm becomes a random search, and thus significantly reduces its ability to exploit the landscape information. In fact, it is not known how to control randomness properly so as to balance exploration and exploitation most effectively.
- **Parameter Tuning and Control**: As the performance of almost any algorithm will depend on its parameter settings, how to tune these parameters to achieve the best performance is a higher level optimization problem. In fact, this is the optimization of an optimization algorithm. It is still an open question. Similarly,

how to control the parameters by varying their values to achieve the best overall performance is also a key challenging issue.

- **Dynamic Landscape**: The problems that have been solved in the current literature usually have fixed landscape. That is, once the problem is defined, its landscape in the search space remain unchanged. However, for dynamic problems and problems with noise, the search landscape can change with time. In such cases, adaptation can be more sophisticated and challenging. It is not clear if most current methods can still work well in such time-dependent, noisy environments.
- **Scalability**: How to solve high-dimensional, large-scale problems effectively? At the moment, most case studies using metaheuristic algorithms are small-scale problems. It is not clear if these algorithms are scalable to deal with large-scale problems effectively.

Therefore, in-depth understanding and theoretical results are needed. Possible research routes may require a combination of mathematical analysis, numerical simulations, empirical observations as well as other tools such as dynamical system theories, Markov theory, self-organization theory and probability. It may even require a paradigm shift in analyzing metaheuristic algorithms. There is no doubt that any theoretical results will provide tremendous insight into understanding metaheursitic algorithms.

All these challenges can present golden opportunities for further research in analyzing adaptation and diversity in metaheuristic algorithms. It can be expected that more efficient tools may be developed to solve more complex, real-world problems with a diverse range of applications.

References

1. Kennedy, J., Eberhart, R.C.: Particle swarm optimization. In: Proceedings of the IEEE International Conference on Neural Networks, pp. 1942–1948. Piscataway, NJ (1995)
2. Yang, X.S.: Nature-Inspired Metaheuristic Algorithms. Luniver Press, UK (2008)
3. Yang, X.S.: Cuckoo Search and Firefly Algorithm: Theory and Applications, Studies in Computational Intelligence, vol. 516, Springer, Berlin (2014)
4. Yang, X.S.: Nature-Inspired Optimization Algorithms. Elsevier, Amsterdam (2014)
5. Ashby, W.R.: Princinples of the self-organizing sysem. In: Von Foerster, H., Zopf, G.W., Jr. (eds.) Pricinples of Self-Organization: Transactions of the University of Illinois Symposium, pp. 255–278. Pergamon Press, London (1962)
6. Dorigo, M., Di Caro, G., Gambardella, L.M.: Ant algorithms for discrite optimization. Artif. Life **5**(2), 137–172 (1999)
7. Fister, I., Fister Jr, I., Yang, X.S., Brest, J.: A comprehensive review of firefly algorithms. Swarm Evol. Comput. **13**(1), 34–46 (2013)
8. Fister, I., Yang, X.S., Brest, J., Fister Jr, I.: Modified firefly algorithm using quaternion representation. Expert Syst. Appl. **40**(18), 7220–7230 (2013)
9. Fister, I., Mernik, M., Filipic, B.: Graph 3-coloring with a hybrid self-adaptive evolutionary algorithm. Comput. Optim. Appl. **54**(3), 741–770 (2013)

10. Yang, X.S., Deb, S.: Cuckoo search via Lévy flights. In: Proceedings of World Congress on Nature and Biologically Inspired Computing (NaBIC 2009), pp. 210–214. IEEE Publications, USA (2009)
11. Pavlyukevich, I.: Lévy flights, non-local search and simulated annealing J. Comput. Phys. **226** (2), 1830–1844 (2007)
12. Yang, X.S., Deb, S.: Engineering optimization by cuckoo search. Int. J. Math. Model. Num. Optim. **1**(4), 330–343 (2010)
13. Yang, X.S., Deb, S.: Multiobjective cuckoo search for design optimization. Comput. Oper. Res. **40**(6), 1616–1624 (2013)
14. Yang, X.S., Deb, S.: Cuckoo search: recent advances and applications. Neural Comput. Appl. **24**(1), 169–174 (2014)
15. Yang, X.S., Deb, S., Fong, S.: Accelerated particle swarm optimization and support vector machine for business optimization and applications. In: Networked Digital Technologies, Communications in Computer and Information Science, vol. 136, pp. 53–66 (2011)
16. Yang, X.S.: A new metaheuristic bat-inspired algorithm. In: Nature Inspired Cooperative Strategies for Optimisation (NICSO 2010), Studies in Computational Intelligence, vol. 284, pp. 65–74. Springer, New York (2010)
17. Yang, X.S.: Bat algorithm for multi-objective optimisation. Int. J. Bio-Inspired Comput. **3**(5), 267–274 (2011)
18. Fister Jr, I., Fister, D., Yang, X.S.: A hybrid bat algorithm. Elektrotehniski Vestn. **80**(1–2), 1–7 (2013)
19. Yang, X.S., Gandomi, A.H.: Bat algorithm: a novel approach for global engineering optimization. Eng. Comput. **29**(5), 1–18 (2012)
20. Yang, X.S., He, X.S.: Bat algorithm: literature review and applications. Int. J. Bio-inspired Comput. **5**(3), 141–149 (2013)
21. Yang, X.S.: Flower pollination algorithm for global optimization. In: Unconventional Computation and Natural Computation, pp. 240–249. Springer, New York (2012)
22. Yang, X.S.: Flower pollination algorithm for global optimization. In: Unconventional Computation and Natural Computation, Lecture Notes in Computer Science, vol. 7445, pp. 240–249. Springer, New York (2012)
23. Yang, X.S., Karamanoglu, M., He, X.S.: Multi-objective flower algorithm for optimization. Procedia Comput. Sci. **18**(1), 861–868 (2013)
24. Yang, X.S., Karamanoglu, M., He, X.S.: Flower pollination algorithm: a novel approach for multiobjective optimization. Eng. Optim. **46**(9), 1222–1237 (2014)
25. Storn, R.: On the usage of differential evolution for function optimization. Biennial Conference of the North American Fuzzy Information Processing Society (NAFIPS), pp. 519–523. Berkeley, CA (1996)
26. Storn, R., Price, K.: Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. J. Global Optim. **11**(4), 341–359 (1997)
27. Price, K., Storn, R., Lampinen, J.: Differential Evolution: A Practical Approach to Global Optimization. Springer, Berlin (2005)
28. Geem, Z.W., Kim, J.H., Loganathan, G.V.: A new heuristic optimization: Harmony search. Simulation **76**(2), 60–68 (2001)
29. Booker, L., Forrest, S., Mitchell, M., Riolo, R.: Perspectives on Adaptation in Natural and Artificial Systems. Oxford University Press, Oxford (2005)
30. Holland, J.: Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Anbor (1975)
31. Yang, X.S., Deb, S., Loomes, M., Karamanoglu, M.: A framework for self-tuning optimization algorithm. Neural Comput. Appl. **23**(7–8), 2051–2057 (2013)
32. Eiben, A.E., Smit, S.K.: Parameter tuning for configuring and analyzing evolutionary algorithms. Swarm Evol. Comput. **1**(1), 19–31 (2011)
33. Belavkin, R.V.: Optimal measures and Markov transition kernels. J. Global Optim. **55**(2), 387–416 (2013)

34. Wang, F., He, X.S., Wang, Y., Yang, S.M.: Markov model and convergence analysis based on cuckoo search algorithm. Comput. Eng. **38**(11), 180–185 (2012). (in Chinese)
35. Huang, G.Q., Zhao, W.J., Lu, Q.Q.: Bat algorithm with global convergence for solving large-scale optimization problem. Appl. Res. Comput. **30**(5), 1323–1328 (2013). (in Chinese)
36. Ren, Z.H., Wang, J., Gao, Y.L.: The global convergence of particle swarm optimization based on Markov chain. Control Theory Appl. **2011**, 462–466 (2011). (in Chinese)
37. Blum, C., Roli, A.: Metaheuristics in combinatorial optimisation: overview and conceptural comparision. ACM Comput. Surv. **35**(2), 268–308 (2003)
38. Wolpert, D.H., Macready, W.G.: No free lunch theorems for optimization. IEEE Trans. Evol. Comput. **1**(1), 67–82 (1997)

Globally Convergent Hybridization of Particle Swarm Optimization Using Line Search-Based Derivative-Free Techniques

A. Serani, M. Diez, E.F. Campana, G. Fasano, D. Peri and U. Iemma

Abstract The hybrid use of exact and heuristic derivative-free methods for global unconstrained optimization problems is presented. Many real-world problems are modeled by computationally expensive functions, such as problems in simulation-based design of complex engineering systems. Objective-function values are often provided by systems of partial differential equations, solved by computationally expensive black-box tools. The objective-function is likely noisy and its derivatives are often not available. On the one hand, the use of exact optimization methods might be computationally too expensive, especially if asymptotic convergence properties are sought. On the other hand, heuristic methods do not guarantee the stationarity of their final solutions. Nevertheless, heuristic methods are usually able to provide an approximate solution at a reasonable computational cost, and have been widely applied to real-world simulation-based design optimization problems. Herein, an overall hybrid algorithm combining the appealing properties of both exact and heuristic methods is discussed, with focus on Particle Swarm Optimization (PSO) and line search-based derivative-free algorithms. The theoretical

A. Serani (✉) · M. Diez · E.F. Campana
CNR-INSEAN, Via di Vallerano 139, Rome, Italy
e-mail: andrea.serani@uniroma3.it

M. Diez
e-mail: matteo.diez@cnr.it

E.F. Campana
e-mail: emiliofortunato.campana@cnr.it

G. Fasano
Department of Management, University Ca' Foscari of Venice,
S. Giobbe, Cannaregio 873, Venice, Italy
e-mail: fasano@unive.it

D. Peri
CNR-IAC, Via dei Taurini 19, Rome, Italy
e-mail: daniele.peri@cnr.it

A. Serani · U. Iemma
Department of Engineering, Roma Tre University, Via Vito Volterra 62, Rome, Italy
e-mail: umberto.iemma@uniroma3.it

X.-S. Yang (ed.), *Recent Advances in Swarm Intelligence and Evolutionary Computation*,
Studies in Computational Intelligence 585, DOI 10.1007/978-3-319-13826-8_2

properties of the hybrid algorithm are detailed, in terms of limit points stationarity. Numerical results are presented for a specific test function and for two real-world optimization problems in ship hydrodynamics.

Keywords Derivative-free optimization · Global optimization · Particle swarm optimization · Line search algorithm · Hybrid optimization algorithm · Simulation-based design · Ship design

1 Introduction

There is plenty of challenging real applications in sciences where optimization is naturally involved, and sophisticated minimization techniques are definitely necessary in order to allocate resources. In particular, these scientific tough problems often involve a remarkably large computational cost, along with large time of computation and machine resources.

Up to 15–20 years ago, to a great extent the main interest of theoreticians in optimization was for methods based on the use of derivatives. This was basically due to the following three strong reasons:

- in several cases derivatives are available when solving computational problems. In particular, they are always 'analytically' available if the nonlinear functions involved are known in closed form (see for instance the work by Griewank in 2000 [14]), and they can be exactly computed (not simply approximated) at reasonable cost in small-medium scale problems [12, 22];
- strong theoretical results have been developed, both in terms of convergence and computational performance, for optimization methods where derivatives (say of first/second order) are available;
- the use of machine resources at a cheaper cost has allowed the solution of problems where derivatives can be suitably approximated by finite differences, using either coarse or fine techniques.

On the other hand, engineering design offers a huge number of real-world problems where scientists are continuously asked to apply robust methods, using the most recent theoretical advances. In particular, design problems often include functions which are not differentiable or where the use of derivatives is possibly discouraged. The following issues motivate the latter statement and give more precise guidelines for analyzing and improving optimization procedures not involving derivatives.

- For **large scale** problems, computing derivatives by finite differences might be prohibitively costly, and also Automatic Differentiation [14] might be of difficult application. Furthermore, the computation of derivatives by finite differences proved to be very harmful when the scale of the problem increases.

- Most of the codes for complex design problems are **parameter dependent**, and the parameters need to be 'properly' assessed. Their correct choice in practice implies that the overall performance of the code needs to be optimized with respect to those parameters. Thus, an implicit optimization problem with respect to these parameters requires a solution, and surely the derivatives of the functions involved are unavailable, being the output of a code *nondifferentiable*.
- Most of the design problems need solution procedures where expensive simulations are performed. Typically, simulations are affected by **noise**, systematic errors arise and stochastic parameters are used, so that derivatives are essentially unavailable or their use may lead to completely destroy the robustness of procedures.

The issues above contribute to motivate the use of efficient and effective derivative-free methods, in order to solve a wide range of challenging problems.

In this chapter we focus on a modification of the PSO algorithm (originally proposed by Kennedy and Eberhart in 1995 [18]), for the solution of the unconstrained *global optimization problem*

$$\min_{x \in \mathbb{R}^n} f(x), \qquad f : \mathbb{R}^n \to \mathbb{R}. \tag{1}$$

At present $f(x)$ is assumed to be a continuous nonlinear, non-convex and computationally expensive function. Observe that in (1) we aim at using a modified PSO algorithm in order to detect a global minimum of $f(x)$, i.e. a point $x^* \in \mathbb{R}^n$ such that $f(x^*) \leq f(x)$, for any $x \in \mathbb{R}^n$.

The reason for which we focus on a modified PSO in order to tackle (1) is that when the function $f(x)$ is computationally costly, exact methods may be definitely too expensive to solve (1). Moreover, some exact methods are possibly unable to provide a current satisfactory approximation of a solution. In the latter cases the use of heuristic approaches may be fruitful, in particular when the computational resources and/or the time allowed for the computation are severely limited, and asymptotically convergent procedures are unaffordable. On the basis of the latter observations, PSO proved to be both effective and efficient on several practical applications [8, 23, 24], so that it is often the heuristics of choice.

Recalling the above considerations, in the framework of derivative-free optimization, we think that combining heuristic procedures and exact methods could be amenable, provided that:

1. the overall hybridized scheme is *efficient*, i.e. it is possibly not too expensive. A legitimate expectation is that the overall computational cost of the combined scheme is in-between the cost of (not combined) PSO and the cost of the exact method;
2. the results provided by the combined procedure are endowed with some theoretical properties, which are guaranteed by an *effective* combination of PSO and the exact method. Typical theoretical properties characterize both the

convergence of sequences of points, and the stationarity of limit points of the sequences generated by the hybridized scheme.

Thus, we focus here on some modifications of PSO, where converging subsequences of iterates are generated. As a consequence, in the next section we are committed to provide clear conditions, under which PSO particles trajectories can be controlled. On the other hand, our modifications proposed for PSO guarantee that the generated sequences of iterates have subsequences converging to stationary points of the objective function (see also [15, 16, 29, 30]). In particular, since there are in the literature theoretical results for several exact derivative-free methods [7, 19], we decided to combine PSO with a line search-based derivative-free algorithm, which is to our knowledge still an unexplored issue, apart from the analysis by Campana et al. in 2009 [1]. We consider here also a numerical experience on a simplified method proposed by Campana et al. in 2009 [1], where the choice of the search directions is particularly 'intuitive', and preserves some relevant theoretical results.

Observe that the aim of this paper is to provide robust methods with a twofold purpose. First we would like to exploit the capability of PSO to provide a satisfactory approximation to a global solution, within a few iterations. Then, by combining PSO with an exact method, we want to force the convergence of subsequences of points toward a stationary point, which satisfies first order optimality conditions for $f(x)$. This paper is specifically concerned with both reporting some theoretical results and performing a valid numerical experience, to prove our theoretical conclusions.

As regards the symbols we adopt in this chapter, subscripts are used to identify the particles in a PSO scheme, whilst the superscripts indicate the iteration. By I we denote the identity matrix, and $\|\cdot\|$ represents the Euclidean norm of a vector/matrix. Finally, $B(c, r)$ is the real ball with center in the vector c and radius $r > 0$, i.e. $B(c, r) = \{y \in \mathbb{R}^n : \|y - c\| \leq r\}$. All the other symbols follow a very standard notation.

In Sects. 2–3 we discuss some issues related to the stability of PSO iteration. Then, the Sects. 4–5 introduce both the theory and the motivations for our modification of PSO iteration. In Sect. 6 we describe our proposal and we carry out the related convergence analysis. Section 7 is then devoted to report a numerical experience on a test case and real problems from ship design. Finally, Sect. 8 contains some conclusions.

2 Stable and Unstable Trajectories for PSO

The strategy of PSO for solving (1) is that of generating the P sequences $\{x_j^k\}, j = 1, \ldots, P$, of iterates in $\mathbb{R}^n$, each associated with the j-th particle of the swarm. The particles share information on the point p_g^k, at any iteration k, satisfying the condition

$$f(p_g^k) \leq f(x_j^h), \qquad \forall h \leq k, \ \ \forall j \in \{1, \ldots, P\}.$$

To our purposes we preliminarily refer to the following PSO iteration, for any $k \geq 0$:

$$\begin{aligned} v_j^{k+1} &= \chi_j \left[w_j^k v_j^k + c_j r_j \otimes (p_j^k - x_j^k) + c_g r_g \otimes (p_g^k - x_j^k) \right], \\ x_j^{k+1} &= x_j^k + v_j^{k+1}, \end{aligned} \tag{2}$$

where $j = 1,\ldots,P$ represents the j-th *particle* (i.e. the j-th sequence of iterates), P is finite, while v_j^k and x_j^k are n-real vectors, which respectively represent the *speed* (i.e. the search direction) and the *position* of the j-th particle at step k. The real bounded coefficients c_j and c_g are typically given at the outset of iteration $k = 0$, and are possibly not modified unless stagnation arises. On the other hand, with $r_j \otimes (p_j^k - x_j^k)$ (similarly with $r_g \otimes (p_g^k - x_j^k)$) we indicate that every entry of the vector $(p_j^k - x_j^k)$ is multiplied by a different value of r_j, which is a random parameter in the uniform distribution between 0 and 1. Finally, for a given $k \geq 0$, the n-real vectors $\{p_j^k\}$ satisfy the conditions

$$f(p_j^k) \leq f(x_j^\ell), \qquad \forall \ell \leq k, \ p_j^k \in \{x_j^\ell\}, \tag{3}$$

moreover, χ_j (*constriction coefficient*) and w_j^k (*inertia*) are positive bounded coefficients. In words the vector p_j^k represents the 'best position' in the j-th subsequence up to iteration k, while p_g^k is the 'best position' among all the vectors $\{p_1^k, \ldots, p_P^k\}$. A keynote issue in PSO is that an effective choice of the coefficients χ, w^k, c_j and c_g is often problem dependent, whereas several standard settings for them have been proposed in the literature (see [25]). Notwithstanding the latter fact, more precise rules for assessing the coefficients in (2) are still sought, with a specific reference to eventually avoid stagnation of PSO iteration.

In order to possibly generalize the recurrence (2), we can assume that the speed v_j^{k+1} depends on all the P vectors $(p_h^k - x_j^k)$ (see also [21]), $h = 1,\ldots,P$, and not only on the pair of vectors $(p_j^k - x_j^k)$, $(p_g^k - x_j^k)$. The resulting new iteration represents the so called Fully Informed PSO (FIPSO). The latter generalization is possibly unessential for our purposes, so that hereafter we limit our analysis to the more standard iteration (2).

Observe that in order to give rules, which ensure that PSO trajectories satisfy suitable conditions, we need to impose some restrictions to the coefficients in (2). In particular, after reviewing the literature we remark that the following (not mutually exclusive) conditions can be reasonably expected to hold for particles trajectories in PSO:

(1) the sequence $\{x_j^k\}$ converges to x_j^*, for any $j = 1,\ldots,P$, with $x_1^* = \cdots = x_P^*$;
(2) the sequence $\{x_j^k\}$ converges to x_j^*, for any $j = 1,\ldots,P$, but possibly $x_j^* \neq x_\ell^*$, with $1 \leq j \neq \ell \leq P$;
(3) the sequence $\{x_j^k\}$ is not diverging for any $j = 1,\ldots,P$, i.e. $\lim_{k\to\infty} \|x_j^k\| < +\infty$, for any $j = 1,\ldots,P$ and any $k \geq 0$.

We highlight that different bounds can be imposed on the coefficients χ, w^k, c_j and c_g in (2), in order to ensure that either of the three conditions (1)–(3) is fulfilled. It is also not difficult to realize that (1) implies (2) (but not *viceversa*) and (2) implies (3) (but not *viceversa*). Thus, the conditions on the coefficients of PSO ensuring (3), are expected to be both weak enough and sufficiently general to allow a wide exploration of the search space. For the latter reason, in this paper we prefer to study and analyze the case (3), while the interested reader can possibly refer to PSO literature (see also Sect. 4) for the analysis on the cases (1)–(2). Now, by (2) let us preliminarily consider the following assumption, in order to simplify our notation.

Assumption 1 We assume in (2) that $\chi_j = \chi > 0$, $c_j = c > 0$ and $r_j = r > 0$, for any $j = 1,\ldots,P$. Moreover, $c_g = \bar{c} > 0$, $r_g = \bar{r} > 0$ and $w_j^k = w$, for any $j = 1,\ldots,P$ and any $k \geq 0$.

Then (see also [2]), using Assumption 1 the iteration (2) is equivalent to the *discrete stationary (time-invariant) system*

$$X_j(k+1) = \begin{pmatrix} \chi w I & -\chi(cr + \bar{c}\bar{r})I \\ \chi w I & [1 - \chi(cr + \bar{c}\bar{r})]I \end{pmatrix} X_j(k) + \begin{pmatrix} \chi(crp_j^k + \bar{c}\bar{r}p_g^k) \\ \chi(crp_j^k + \bar{c}\bar{r}p_g^k) \end{pmatrix}, \tag{4}$$

where

$$X_j(k) = \begin{pmatrix} v_j^k \\ x_j^k \end{pmatrix} \in \mathbb{R}^{2n}, \qquad k \geq 0. \tag{5}$$

For a given j, the vectors $\{X_j(k)\}$ identify a sequence of points in $\mathbb{R}^{2n}$ and represent indeed the trajectory of the j-th particle in the state space $\mathbb{R}^{2n}$. By definition, since $X_j(k)$ represents a *state* vector, it can be split into the so called *free response* $X_{j\mathscr{L}}(k)$ and the *forced response* $X_{j\mathscr{F}}(k)$ (see also [27]), such that

$$X_j(k) = X_{j\mathscr{L}}(k) + X_{j\mathscr{F}}(k), \tag{6}$$

being

$$X_{j\mathscr{L}}(k) = \Phi_j(k)X_j(0), \qquad X_{j\mathscr{F}}(k) = \sum_{\tau=0}^{k-1} H_j(k-\tau)U_j(\tau), \tag{7}$$

and (with a little computation)

$$\Phi_j(k) = \begin{pmatrix} \chi w I & -\chi(cr + \bar{c}\bar{r})I \\ \chi w I & [1 - \chi(cr + \bar{c}\bar{r})]I \end{pmatrix}^k, \tag{8}$$

$$H_j(k-\tau) = \begin{pmatrix} \chi w I & -\chi(cr + \bar{c}\bar{r})I \\ \chi w I & [1 - \chi(cr + \bar{c}\bar{r})]I \end{pmatrix}^{k-\tau-1}, \tag{9}$$

$$U_j(\tau) = \begin{pmatrix} \chi(crp_j^\tau + \bar{c}\bar{r}p_g^\tau) \\ \chi(crp_j^\tau + \bar{c}\bar{r}p_g^\tau) \end{pmatrix}. \tag{10}$$

We highlight the important fact that the free response $X_{j\mathscr{L}}(k)$ in (6)–(7) only depends on the initial point $X_j(0)$, and is not affected by changes of the vectors p_j^τ, p_g^τ, $\tau \geq 0$. The latter observation is of great interest, in order to assess rules for the parameters χ, w, c, r, $\bar{c}$ and $\bar{r}$, as the next section shows.

3 Issues on Assessing Parameters in PSO

As described in the last section, the fruitful choice of the parameters in PSO is to a large extent guided by a couple of issues: the efficiency of the overall scheme and the necessity of guaranteeing at least non-diverging trajectories of the particles. As regards the first issue, the literature of PSO provides several suggestions which have proved to be effective in most cases (see for instance [4, 26]). Conversely, some more recent papers, concerned with studying the stability of particles trajectory, have detailed some restrictions on PSO parameters in order to dynamically control the trajectory of particles, and make them more accurately predictable. On this guideline, papers like Kadirkamanathan et al. in 2006 [17], and Gazi in 2012 [12] are advisable, since they contain clear indications on the latter issue.

Here we want to propose a unified approach for parameters assessment in PSO, using the reformulation (4)–(5). In practice (see also [27]), as long as Assumption 1 holds, our perspective is that of using classic analysis for discrete linear systems in order to deduce bounds on PSO parameters. On the other hand, we want to carry on our analysis as rigorously as possible, so that we will separately develop formal conditions for the following three purposes:

(a) define *necessary conditions* (possibly not sufficient) on PSO parameters, so that particles trajectories are not diverging, i.e. the quantities $\|X_j(k)\|$ are limited, for any $j = 1,\ldots,P$ and any $k \geq 0$;
(b) ensure *no stagnation* (see [13]) for a modified PSO scheme;

(c) possibly introduce simple modifications to PSO, so that the resulting scheme is *globally convergent* to stationary points, i.e. for any choice of the initial positions $x_1^0, \ldots, x_P^0$ of the particles, the scheme generates sequences of points like $\{y^k\}$, such that

$$\liminf_{k\to\infty} \left\|\nabla f(y^k)\right\| = 0.$$

Observe that the latter condition substantially guarantees that possibly 'true' minima for $f(x)$ are eventually outreached, even in case stagnation arises. We strongly remark that though we used the symbol $\nabla f(x)$, the latter relation will be proved *without using any information* on $\nabla f(x)$, so that a completely derivative-free method will be developed.

In the current section we consider the issue (a), while in the next section issues (b) and (c) will be analyzed in detail.

As well known, for a discrete linear system like (4)–(5) (see for instance [27]), if the j-th trajectory $\{X_j(k)\}$ in (6) is non-diverging, then

$$\lim_{k\to\infty} X_j(k) = \lim_{k\to\infty} X_{j\mathscr{F}}(k), \qquad j = 1, \ldots, P.$$

In other words, the free response $X_{j\mathscr{L}}(k)$ is bounded away from zero only for finite values of the index k, and

$$\lim_{k\to\infty} X_{j\mathscr{L}}(k) = 0. \tag{11}$$

This introduces a possible rule to address bounds for PSO parameters, since relation (11) imposes some restrictions to the eigenvalues of matrix $\Phi_j(k)$ in (8). Observing that $\Phi_j(k) = \Phi_j(1)^k$, for any $k \geq 0$, the $2n$ eigenvalues of the unsymmetric matrix $\Phi_j(1)$ are real. In particular, by setting for the sake of simplicity in (8)

$$a = \chi w, \qquad \omega = \chi(cr + \bar{c}\bar{r}), \tag{12}$$

we can prove that after some computation the matrix $\Phi_j(1)$ has two distinct eigenvalues λ_{j1} and λ_{j2} given by

$$\begin{aligned} \lambda_{j1} &= \frac{1 - \omega + a - \left[(1-\omega+a)^2 - 4a\right]^{1/2}}{2}, \\ \lambda_{j2} &= \frac{1 - \omega + a + \left[(1-\omega+a)^2 - 4a\right]^{1/2}}{2}, \end{aligned} \tag{13}$$

each of them with algebraic multiplicity n. Thus, a necessary (but in general not sufficient) condition for the j-th trajectory $\{X_j(k)\}$ to be non-diverging and satisfy

$\lim_{k\to\infty} X_{j\mathscr{L}}(k) = 0$, is the following, which yields the required conditions on the coefficients of PSO iteration.

Lemma 1 *Consider the PSO iteration* (2). *Let Assumption 1 hold. Let the eigenvalues* λ_{j1} *and* λ_{j2} *in* (13) *satisfy the conditions*

$$|\lambda_{j1}| < 1, \qquad |\lambda_{j2}| < 1, \tag{14}$$

for any $j = 1,\ldots,P$. *Then, the sequence* $\{X_{j\mathscr{L}}(k)\}$ *satisfies* $\lim_{k\to\infty} X_{j\mathscr{L}}(k) = 0$, *and condition* (14) *is a necessary condition for the trajectory* $\{X_j(k)\}$ *to be non-diverging*.

Note that most of the typical settings for PSO parameters proposed in the literature (see e.g. [4, 5, 26]), satisfy the condition (14). Moreover, Lemma 1 does not guarantee that the sequence $\{X_j(k)\}$, for a given j, is converging, and indeed it possibly does not admit even limit points. This means that condition (14) only provides a result for item (a), but is definitely inadequate to treat items (b) and (c). This also implies that for instance if (14) holds, then possibly the trajectory $\{X_j(k)\}$ is not converging, or some of its subsequences converge to the point x_j^* which is not a minimum and does not satisfy the property

$$f(x_j^*) \le f(x), \qquad \forall x \in B(x_j^*, \varepsilon),\ \varepsilon > 0.$$

In the next sections we focus on a rigorous analysis of the latter issue. I.e., under mild assumptions we propose a modified PSO scheme such that, if the function $f(x)$ is continuously differentiable, the sequence $\{x_1^1, \ldots, x_P^1, \ldots, x_1^k, \ldots, x_P^k\}$ admits *stationary limit points* for $f(x)$, so that

$$\liminf_{k\to\infty} \left\|\nabla f(x_j^k)\right\| = 0 \qquad \text{or} \qquad \lim_{k\to\infty} \left\|\nabla f(x_j^k)\right\| = 0. \tag{15}$$

As the reader can expect, there is a theoretical and computational evidence that fulfilling condition (15) may be met (in several real applications) at the expense of a reasonably larger computational cost, with respect to the standard PSO iteration (2).

4 PSO and Stationarity

In Sect. 1 we have detailed some motivations, to explain why in the last two decades design optimization and simulation-based optimization have required effective and robust derivative-free optimization methods. Exploiting also the recent advances on parallel computing, the last two decades have seen in particular the blow up of a remarkably effective class of optimization methods, endowed with complete convergence analysis and competitive performance: namely *direct search methods*. The latter class (see [19]) counts several optimization methods, which do not use derivatives but basically rely on "*the ranks of a countable set of function*

values" [19], i.e. on comparing the objective function values in specific points of the search space.

Among direct search methods we focus here on a subclass of iterative techniques, which is usually addressed in the literature as *Generating Set Search* (GSS). In the latter class, the main idea is that of decreasing the objective function at each iteration, on a *cone* in $\mathbb{R}^n$ generated by suitable search directions. *Pattern search methods* are in the GSS class, and have the distinguishing feature of enforcing, at each iteration, a simple decrease of the objective function. Conversely, also *line search-based* derivative-free methods are iterative schemes in GSS class, however they impose at each iteration a so called *sufficient reduction* of $f(x)$. We want to show that global convergence properties of a modified PSO scheme may be obtained by properly combining PSO with a line search-based derivative-free method, so that convergence to stationary points can be forced at a reasonable cost (see also items (b) and (c) of Sect. 3). On this guideline, there is plenty of examples where evolutionary strategies are combined with GSS schemes and yield globally convergent algorithms (see for instance [15, 31, 32]). In particular, in the last reference PSO is hybridized within a pattern search framework, and a resulting method converging to stationary points is given.

Observe that in the literature of derivative-free methods we can also find PSO-based approaches combined with a trust-region framework (see [31, 32]), in order to provide again globally convergent methods to stationary points.

In this section we consider the solution of the problem (1), and we focus on a modified PSO scheme, combined with a line search-based derivative-free algorithm. We study in particular the nature of limit points of the sequences $\{x_j^k\}$, $j = 1, \ldots, P$, when Assumption 1 holds. However, we think that to have a better insight in our analysis, the following very preliminary results (see also [7]) can help the reader grasp the importance of the GSS class, in order to ensure convergence to stationary points.

Definition 1 Given the set of vectors $D = \{d_1, \ldots, d_m\}$ of $\mathbb{R}^n$, we say that D is a Positively Spanning Set (PSS) if for any vector $u \in \mathbb{R}^n$ we have

$$u = \sum_{i=1}^{m} \alpha_i d_i, \quad \alpha_i \geq 0,$$

i.e. any vector u of $\mathbb{R}^n$ can be expressed as the weighted sum of the vectors in D, using nonnegative weights.

Thus, a PSS substantially provides a set of vectors which positively span the space $\mathbb{R}^n$. It can be easily proved that if D is a PSS of $\mathbb{R}^n$, then its cardinality must be at least $n + 1$. It is very easy to define PSSs; simple examples of them in $\mathbb{R}^2$ are given in Fig. 1, where $m = 4$ (*top* and *bottom*) and $m = 3$ (*middle*). In addition, there is the following nice property of PSSs that we are going to exploit in our proposal. If the point $x \in \mathbb{R}^n$ is not stationary for f in (1) (i.e. $\nabla f(x) \neq 0$), given the PSS D in $\mathbb{R}^n$, there exists at least one vector, say $\hat{d} \in D$, such that $\nabla f(x)^T \hat{d} < 0$, meaning that

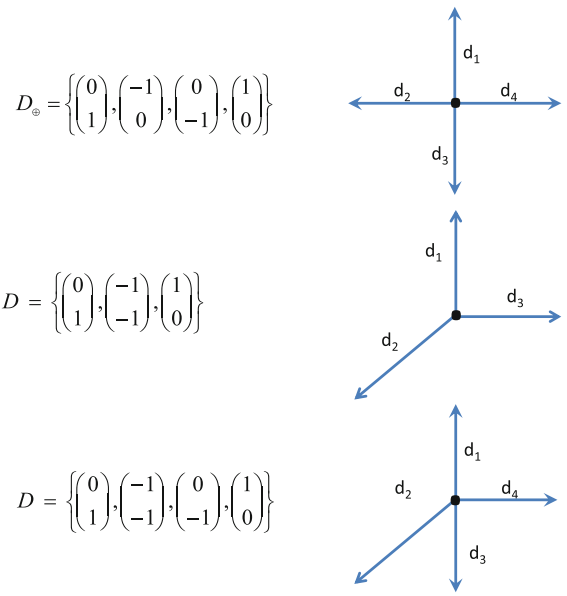


Fig. 1 Examples of PSSs in $\mathbb{R}^2$. The subscript '$\oplus$' in the uppermost PSS means that the vectors in the set are the coordinate unit vectors $\pm e_i$, $i = 1,\ldots,n$

the direction $\hat{d}$ is of descent for $f(x)$ at x. The latter fact ensures that if the current point is not stationary, and a PSS is available, roughly speaking there is at least one *direction of descent* for $f(x)$ in the PSS.

A so called *cosine measure* $cm(D)$ can be associated to the PSS D of $\mathbb{R}^n$, defined as follows.

Definition 2 Given the PSS $D = \{d_1,\ldots,d_m\}$ in $\mathbb{R}^n$, we define the cosine measure $cm(D)$ of D as

$$cm(D) = \min_{v \in \mathbb{R}^n \setminus \{0\}} \max_{d_i \in D} \left\{ \frac{v^T d_i}{\|v\| \|d_i\|} \right\},$$

being always $cm(D) > 0$.

By Definition 2 the quantity $cm(D)$ clearly represents a measure of the least orthogonal projection of any vector in $\mathbb{R}^n$ on vectors in D. As a consequence, if $cm(D) \to 0$ then D might be not a 'good' PSS and consequently it might be difficult to find a descent direction for f in D. The following result clarifies the importance of introducing PSSs in derivative-free optimization, in order to characterize stationary points, without using any information on the gradient of f in (1).

Theorem 1 *Let $D = \{d_1,\ldots,d_m\}$ be a PSS. Suppose the function $f(x)$ in (1) is continuously differentiable in $\mathbb{R}^n$ and the gradient $\nabla f(x)$ satisfies the Lipschitz condition*

$$\|\nabla f(y) - \nabla f(x)\| \leq v\|y - x\|, \qquad \forall y \in \overset{\circ}{B}(x, \alpha\delta), \quad \delta = \max_{1 \leq i \leq m} \|d_i\|,$$

for some $v > 0$ and $\alpha > 0$. If $f(x) \leq f(x + \alpha d_i)$, $i = 1,\ldots,m$, then the following bound holds for the gradient of $f(x)$

$$\|\nabla f(x)\| \leq \frac{v}{2}\frac{1}{cm(D)}\alpha\delta. \tag{16}$$

Proof Since D is a PSS, there exists at least one vector in the set $\{d_i\}$, say $\hat{d} \in D$, such that

$$cm(D) \leq \frac{-\nabla f(x)^T \hat{d}}{\|\nabla f(x)\|\|\hat{d}\|},$$

hence, recalling that α is positive, we have equivalently

$$cm(D)\|\nabla f(x)\|\|\hat{d}\|\alpha \leq -\nabla f(x)^T(\alpha\hat{d}). \tag{17}$$

□

On the other hand, the hypothesis of continuous differentiability of $f(x)$ allows to apply the Mean Value Theorem in the integral form, being for any $d \in D$

$$f(x + \alpha d) = f(x) + \int_0^1 \nabla f[x + t\alpha d]^T(\alpha d)dt,$$

or equivalently

$$0 \leq f(x + \alpha d) - f(x) = \int_0^1 \nabla f[x + t\alpha d]^T(\alpha d)dt. \tag{18}$$

Combining (17) and (18) we obtain for the direction $\hat{d}$

$$
\begin{aligned}
cm(D)\|\nabla f(x)\|\ \|\hat{d}\|\alpha &\leq -\nabla f(x)^T(\alpha\hat{d}) + \int_0^1 \nabla f[x + t\alpha\hat{d}]^T(\alpha\hat{d})dt \\
&\leq \int_0^1 \left|\nabla f[x + t\alpha\hat{d}]^T(\alpha\hat{d}) - \nabla f(x)^T(\alpha\hat{d})\right|dt \\
&\leq \int_0^1 \left\|\nabla f[x + t\alpha\hat{d}] - \nabla f(x)\right\|\|\alpha\hat{d}\|dt \\
&\leq v\int_0^1 t\|\alpha\hat{d}\|^2 dt \leq \frac{v}{2}\alpha^2\|\hat{d}\|^2,
\end{aligned}
$$

which immediately yields (16).

Loosely speaking, in order to suggest the reader the importance of Theorem 1, it can be rephrased in the following simple way. If the PSS D is available and the value $f(x)$ cannot be decreased on points along all the directions in D, then it means that the iterate x is a stationary point. Indeed, in the latter case, from (16) we have $\lim_{\alpha\to 0}\|\nabla f(x)\| = 0$. Also note that if the PSS D is poor (i.e. $cm(D)$ is small), then the bound on the gradient (16) is poor accordingly.

Since in our proposal we combine PSO with a line search-based derivative-free method, which relies on the use of PSSs, with out proposal we will be able to characterize stationarity conditions for a modified PSO scheme, without recurring to any information on the gradient $\nabla f(x)$. In the next section we describe some basic properties of the line search-based derivative-free method we couple with PSO.

5 Preliminaries on the Line Search-Based Method Adopted

We consider in this section the proposal by Lucidi and Sciandrone in 2002 [20], which includes some line search-based derivative-free methods. Since it is our intention to reduce the complexity of our proposal as much as possible, the next result represents a simplified version of the material of Lucidi and Sciandrone in 2002 [20].

Proposition 1 *Let $f : \mathbb{R}^n \to \mathbb{R}$, with f continuously differentiable in $\mathbb{R}^n$. Suppose that the points in the sequence $\{x^k\}$ are bounded. Suppose the directions $d_1^k, \ldots, d_m^k$ are bounded and form a positively spanning set of $\mathbb{R}^n$. Then, the following stationarity condition holds*

$$
\lim_{k\to\infty}\|\nabla f(x^k)\| = 0 \quad \text{if and only if} \quad \lim_{k\to\infty}\sum_{j=1}^{m}\min\left\{0, \nabla f(x^k)^T d_j^k\right\} = 0. \quad (19)
$$

Table 1 The line search-based derivative-free algorithm LS-DF (see also Lucidi and Sciandrone 2002 [20])

Step 0. Set $k = 0$; choose $x^0 \in \mathbb{R}^n$, set $\bar{\alpha}^0 > 0$, $\gamma > 0$, $\theta \in (0,1)$.

Step 1. If there exists $y^k \in \mathbb{R}^n$ such that $f(y^k) \leq f(x^k) - \gamma\bar{\alpha}^k$, then go to **Step 4**.

Step 2. If there exists $j \in \{1,\ldots,m\}$ and an $\alpha^k \geq \bar{\alpha}^k$ such that

$$f(x^k + \alpha^k d_j^k) \leq f(x^k) - \gamma(\alpha^k)^2,$$

then set $y^k = x^k + \alpha^k d_j^k$, set $\bar{\alpha}^{k+1} = \alpha^k$ and go to **Step 4**.

Step 3. Set $\bar{\alpha}^{k+1} = \theta\bar{\alpha}^k$ and $y^k = x^k$.

Step 4. Find x^{k+1} such that $f(x^{k+1}) \leq f(y^k)$, set $k = k+1$ and go to **Step 1**.

Let us consider the sequence $\{x^k\}$ in (19) and Theorem 1. By Proposition 1 necessary and sufficient conditions of stationarity for the sequence $\{x^k\}$ can be accomplished by simply exploiting at any iterate x^k the function $f(x)$ (through its directional derivative $\nabla f(x^k)^T d_j^k$), along the search directions $d_1^k, \ldots, d_m^k$. Table 1 details a line search-based derivative-free method for unconstrained minimization, which uses the results of Proposition 1. In Lucidi and Sciandrone [20] a complete convergence analysis was developed for the Algorithm LS-DF and the following conclusion was proved (see e.g. Proposition 5.1 in [20]).

Proposition 2 *Suppose the directions $d_1^k, \ldots, d_m^k$ satisfy Proposition 1. Consider the sequence $\{x^k\}$ generated by the Algorithm* LS-DF *and let the level set $\mathcal{L}_0 = \{x \in \mathbb{R}^n : f(x) \leq f(x^0)\}$ be compact. Then we have*

$$\liminf_{k\to\infty} \|\nabla f(x^k)\| = 0. \tag{20}$$

Note that the condition (20) is weaker than (19), and in principle is met only asymptotically by the Algorithm LS-DF. Of course, recalling also the results in Theorem 1, a practical stopping condition of Algorithm LS-DF could be obtained by monitoring the steplength $\bar{\alpha}^k$ at Steps 2 and 3. The algorithm can stop when $\bar{\alpha}^k$ becomes sufficiently small. Also observe that at Step 4 the point x^{k+1} might be computed for instance by any heuristic procedure; nevertheless, we can set in any case $x^{k+1} \equiv y^k$, since convergence analysis does not require $f(x^{k+1}) < f(y^k)$.

As a final consideration, note that relation (20) may be strongly strengthened by choosing a different (and computationally more expensive) strategy at Step 2 of the Algorithm LS-DF. Indeed, instead of requiring that at Step 2 just one direction is of sufficient decrease for the objective function (i.e. $f(x^k + \alpha^k d_j^k) \leq f(x^k) - \gamma(\alpha^k)^2$ for *at least one index* $j \in \{1,\ldots,m\}$), we can exploit $f(x)$ along *all* the directions

Table 2 The line search-based derivative-free algorithm LS-DF$_+$ in Lucidi and Sciandrone (2002) [20]

Step 0. Set $k = 0$. Choose $x^0 \in \mathbb{R}^n$ and $\bar{\alpha}_j^0 > 0$, $j = 1, \ldots, n+1$, $\gamma > 0$, $\delta \in (0,1)$, $\theta \in (0,1)$.

Step 1. Set $j = 1$ and $y_1^k = x^k$.

Step 2. If $f(y_j^k + \bar{\alpha}_j^k d_j^k) \leq f(y_j^k) - \gamma(\bar{\alpha}_j^k)^2$ then

compute α_j^k by Line search$(\bar{\alpha}_j^k, y_j^k, d_j^k, \gamma, \delta)$ and set $\bar{\alpha}_j^{k+1} = \alpha_j^k$;

else set $\alpha_j^k = 0$, and $\bar{\alpha}_j^{k+1} = \theta \bar{\alpha}_j^k$.

Set $y_{j+1}^k = y_j^k + \alpha_j^k d_j^k$.

Step 3. If $j < m$ then set $j = j + 1$ and go to **Step 2**.

Step 4. Find x^{k+1} such that $f(x^{k+1}) \leq f(y_{n+1}^k)$, set $k = k + 1$ and go to **Step 1**.

Line search$(\bar{\alpha}_j^k, y_j^k, d_j^k, \gamma, \delta)$:

Compute the steplength $\alpha_j^k = \min\left\{\bar{\alpha}_j^k/\delta^h,\ h = 0, 1, \ldots\right\}$ such that

$$f(y_j^k + \alpha_j^k d_j^k) \leq f(x^k) - \gamma(\alpha_j^k)^2,$$

$$f\left(y_j^k + \frac{\alpha_j^k}{\delta} d_j^k\right) \geq \max\left[f(y_j^k + \alpha_j^k d_j^k), f(y_j^k) - \gamma\left(\frac{\alpha_j^k}{\delta}\right)^2\right].$$

$\{d_1^k, \ldots, d_m^k\}$ in the PSS. The resulting algorithm is Algorithm LS-DF$_+$ in Table 2. We recall that, for the sake of simplicity and in order to keep the computational cost as low as possible, we will couple PSO only with Algorithm LS-DF. In the following proposition we summarize convergence properties also for Algorithm LS-DF$_+$: we remark that they are definitely stronger than the result in Proposition 2.

Proposition 3 *Suppose the directions $d_1^k, \ldots, d_m^k$ satisfy Proposition 1. Consider the sequence $\{x^k\}$ generated by the Algorithm* LS-DF$_+$ *and let the level set $\mathscr{L}_0 = \{x \in \mathbb{R}^n : f(x) \leq f(x^0)\}$ be compact. Then we have*

$$\lim_{k \to \infty} \|\nabla f(x^k)\| = 0. \tag{21}$$

It is evident that in Algorithm LS-DF$_+$ the stronger convergence result is obtained at the expense of a larger computational cost in Step 2. In addition, the procedure LINE SEARCH () is aimed to determine a possible expansion of the steplength α_j^k.

6 A Hybrid Algorithm

In this section we propose a hybrid algorithm, obtained by coupling the PSO scheme described in Sect. 2 with the algorithm in Table 1. We remind the reader that the resulting method should be endowed with both the (local) convergence properties of Algorithm LS-DF and the (global) strategies of exploration of PSO (see also Sect. 1).

Of course, it is far obvious that simply alternating a finite sequence of steps of PSO and a finite sequence of steps of Algorithm LS-DF would provide a method satisfying a property similar to Proposition 2. However, the latter strategy might be a blind sequential application of two different algorithms, which does not exploit their peculiarities. On the contrary, we prefer to consider a scheme which at once both *exploits* (local strategy) and *explores* (global strategy) the objective function. Thus, we consider here a PSO-based method which attempts to detect a global minimum of the objective function, while retaining the asymptotic convergence properties of Algorithm LS-DF.

Our proposal (namely Algorithm LS-DF_PSO) is summarized in Table 3 and its convergence properties to stationary points are summarized in the next proposition (see also [20]).

Proposition 4 *Suppose the directions $d_1^k,\ldots,d_m^k$ satisfy Proposition 1. Consider the sequence $\{x^k\}$ generated by the Algorithm* LS-DF_PSO *and let the level set $\mathscr{L}_0 = \{x \in \mathbb{R}^n : f(x) \leq f(x^0)\}$ be compact. Then we have*

$$\liminf_{k\to\infty} \|\nabla f(x^k)\| = 0. \tag{22}$$

Proof Observe that the Algorithm LS-DF and the Algorithm LS-DF_PSO differ only at Step 1 and Step 4. Indeed, Step 1 and Step 4 of Algorithm LS-DF_PSO are simply obtained from the corresponding steps of Algorithm LS-DF, observing that the iterates y^k (Step 1) and x^{k+1} (Step 4) are computed by using PSO. Therefore, convergence properties of the sequence generated by the Algorithm LS-DF_PSO are basically inherited from Proposition 2. □

We conclude this section by highlighting that similarly to Algorithm LS-DF_PSO, instead of using Algorithm LS-DF, we can couple PSO with the Algorithm LS-DF$_+$. The resulting hybrid scheme, would be much similar to Algorithm LS-DF_PSO but more expensive than the algorithm in Table 3. Moreover, it would be also endowed

Table 3 The line search-based derivative-free algorithm LS-DF_PSO

Data. Set $k = 0$; choose $x^0 \in \mathbb{R}^n$ and $z_j^0, v_j^0 \in \mathbb{R}^n$, $j = 1, \ldots, P$. Set $\bar{\alpha}^0 > 0$, $\gamma > 0$, $\theta \in (0, 1)$.

Step 1. Set $h_k \geq 1$ integer. Apply h_k PSO iterations considering the P particles with respective initial velocities and positions v_j^k and z_j^k, $j = 1, \ldots, P$. Set $y^k = argmin_{1 \leq j \leq P,\ \ell \leq h_k} \{f(z_j^\ell)\}$. If $f(y^k) \leq f(x^k) - \gamma \bar{\alpha}^k$, then set $v_j^k = v_j^{k+h_k}$ and $z_j^k = z_j^{k+h_k}$, and go to **Step 4**.

Step 2. If there exists $j \in \{1, \ldots, m\}$ and an $\alpha^k \geq \bar{\alpha}^k$ such that

$$f(x^k + \alpha^k d_j^k) \leq f(x^k) - \gamma(\alpha^k)^2,$$

then set $y^k = x^k + \alpha^k d_j^k$, $\bar{\alpha}^{k+1} = \alpha^k$ and go to **Step 4**.

Step 3. Set $\bar{\alpha}^{k+1} = \theta \bar{\alpha}^k$ and $y^k = x^k$.

Step 4. Set $q_k \geq 1$ integer. Apply q_k PSO iterations considering the P particles with respective initial velocities and positions v_j^k and z_j^k, $j = 1, \ldots, P$. Set $x^{k+1} = argmin_{1 \leq j \leq P,\ \ell \leq q_k} \{f(z_j^\ell)\}$; if x^{k+1} satisfies $f(x^{k+1}) \leq f(y^k)$, then set $k = k + 1$ and go to **Step 1**.

with convergence properties similar to those reported in Proposition 3, so that the condition (22), which is worth for Algorithm LS-DF_PSO, would be reinforced with a condition like (21).

7 Numerical Results

Numerical results are presented for the Rosenbrock function and for two real-world optimization problems in ship hydrodynamics. Specifically, a four-parameters shape optimization of a catamaran is investigated for (a) resistance reduction in calm water at fixed speed, and (b) expected resistance reduction in wave, taking into account stochastic sea state and speed. For all optimization problems, the deterministic implementation of PSO presented by Serani et al. in 2014 [28] is used for extension to LS-DF_PSO. A number of particles equal to $4n$ is used, with initialization over the variables domain by Hammersely sequence sampling, and PSO coefficients given by Clerc in 2006 [6], i.e., $\chi = 0.721$, $w = 1$, $cr = \overline{cr} = 1.655$ (see the work by Serani et al. in 2014 [28] for details). LS-DF_PSO parameters are set as $h_k = 1$, $\gamma = 10^{-3}$, $\theta = 0.5$, $\alpha^k = 0.25$ of the design variable range.

Specifically, the minimization in $\mathbb{R}^2$ of the Rosenbrock function,

$$f(x, y) = (a - x)^2 + b(y - x^2)^2 \tag{23}$$

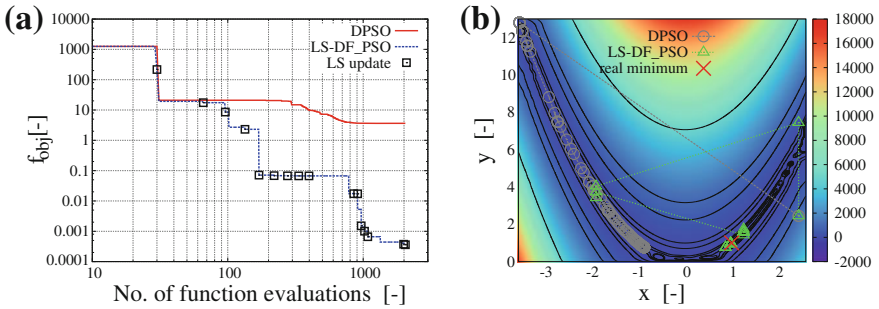


Fig. 2 Rosenbrock function minimization: convergence of the objective function (**a**) and global-optimum location history (**b**)

with $n = 2$, $a = 1$, $b = 100$, $-20 \leq x \leq 20$ and $-20 \leq y \leq 20$, is used as an explanatory test problem. Figure 2a shows the convergence of the LS-DF_PSO algorithm, compared to the standard PSO. Black squares indicate LS-DF_PSO iterations where the line search procedure LS is used to improve the optimum location. Figure 2b shows a comparison of the algorithms' convergence in a close up of the variables domain. The global-optimum location history is depicted, along with the real minimum, which is located at $x = 1$, $y = 1$. The beneficial effects of using PSO with LS are evident, providing a faster and more effective convergence to the optimum, along with the identification of the proper region of the global optimum.

The shape optimization of the Delft catamaran is shown as an example of industrial design problems. The Delft catamaran is a concept ship used for experimental and numerical benchmarks (see, e.g., the numerical studies presented by Diez et al. in 2013 [11]). Figure 3 shows the Delft catamaran model during towing tank experiments at CNR-INSEAN, along with a detail of the original hull shape. The design optimization problems are taken from the work by Chen et al. in 2014 [3] and Diez et al. in 2013 [9] respectively, and solved by means of stochastic radial-basis functions interpolation (details may be found in the work by Volpi et al. in 2014 [33]) of high-fidelity unsteady Reynolds-averaged Navier-Stokes (URANS)

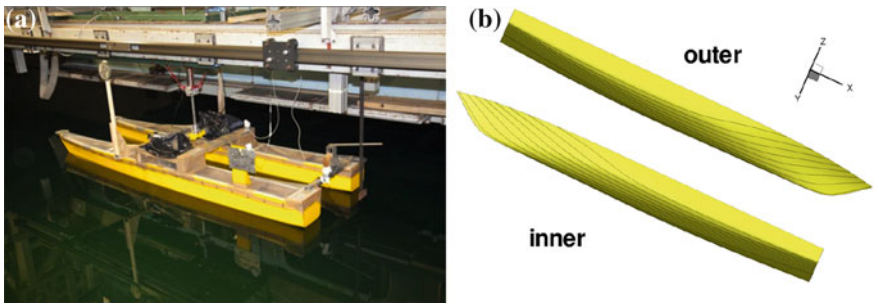


Fig. 3 Delft catamaran: towing tank experiments at CNR-INSEAN (**a**) and detail of the original geometry (**b**)

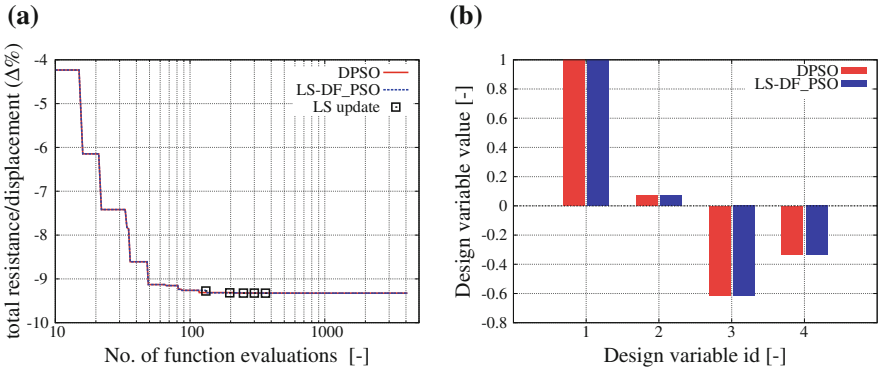


Fig. 4 Minimization of calm-water resistance for the Delft catamaran: convergence of the objective function (**a**) and global-optimum design variables (**b**)

simulations. For the first problem, the design objective is the minimization of the resistance in calm water at Froude number equal to 0.5 (see [3] for details). For the second problem, the design objective is the reduction of the expected value of the mean resistance in head waves, taking into account stochastic sea state in the North Pacific ocean and variable speed (see [9] for details). For both problems, four design variables control global shape modifications, based on the Karhunen-Loève expansion of the shape modification vector [10].

Figure 4a shows the convergence of the minimization procedure for the first problem, comparing PSO with LS-DF_PSO. LS is required only few times, as indicated by the black squares, and the minima provided by the two algorithms are extremely close, as shown in Figs. 4b and 5. Nevertheless, it may be noted that, at very reduced additional cost, LS-DF_PSO provides a solution certified with stationarity properties.

Figure 6a shows the convergence of the minimization procedure for the second problem, comparing PSO with LS-DF_PSO. LS is required and applied a larger number of times than in the previous problem, and is essential to identify the global

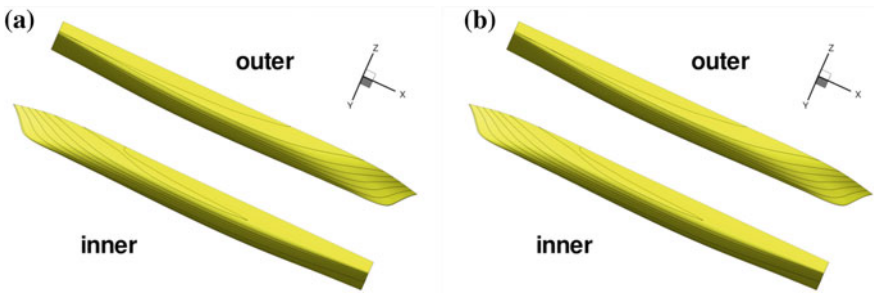


Fig. 5 Minimization of calm-water resistance for the Delft catamaran: optimal shape design by PSO (**a**) and LS-DF_PSO (**b**)

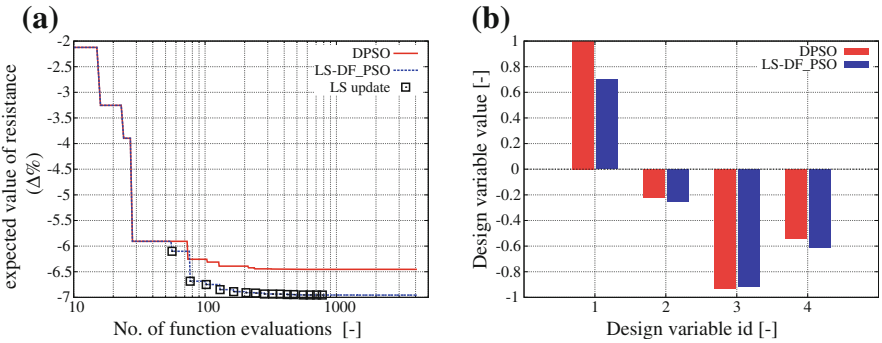


Fig. 6 Minimization of expected value of mean resistance in head waves for the Delft catamaran: convergence of the objective function (**a**) and global-optimum design variables (**b**)

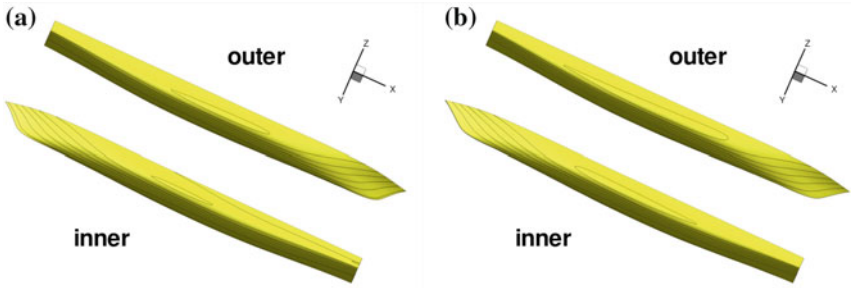


Fig. 7 Minimization of expected value of mean resistance in head waves for the Delft catamaran: optimal shape design by PSO (**a**) and LS-DF_PSO (**b**)

optimum, as shown in Fig. 6b. As a result, optimal shape designs provided by PSO and LS-DF_PSO are noticeably different (within the context of current application's variation), as shown in Fig. 7. Additionally, it may be noted that the solution given by LS-DF_PSO is also endowed with stationarity properties.

8 Conclusions

In this chapter we have detailed some globally convergent modifications of PSO iteration (2), for the solution of the unconstrained global optimization problem (1). Under mild assumptions, Proposition 4 proved that at least a subsequence of the iterates generated by our modified PSO, namely Algorithm LS-DF_PSO, converges to a stationary point, which is possibly a minimum point. We recall that using the standard PSO iteration, by no means we can guarantee convergence towards stationary points, unless we consider trivial cases of no practical interest. Thus, our

result reinforces the theoretical properties of modified PSO schemes. To the best of our knowledge, our result is also among the first attempts to couple PSO with line search-based derivative-free schemes (see also [30, 31] for extensions to trust-region derivative-free approaches), where a modified PSO scheme is proved to satisfy conditions like (20) or (21).

On the basis of our experience, which seems confirmed by the results reported here, we are persuaded that a fruitful coupling of PSO with an iterative globally convergent derivative-free method, should yield a compromise, between the fast progress of PSO (global search) in the early iterations, and the capability to exploit (local search) the objective function.

We also have reported numerical experiences on a significant test function and two ship design problems, which confirm that LS-DF_PSO is more effective (and to a great extent equally efficient) than PSO. Indeed, LS-DF_PSO is able to achieve better solutions/designs, and provides stationarity properties at the associated optimal points.

Acknowledgments The work of M. Diez is supported by the US Navy Office of Naval Research, NICOP Grant N62909-11-1-7011, under the administration of Dr. Ki-Han Kim and Dr. Woei-Min Lin. The work of A. Serani, E.F. Campana and G. Fasano is supported by the Italian Flagship Project RITMARE, coordinated by the Italian National Research Council and funded by the Italian Ministry of Education, within the National Research Program 2011–2013.

References

1. Campana, E.F., Fasano, G., Peri, D.: Globally convergent modifications of particle swarm optimization for unconstrained optimization. In: Bohua, S. (ed.) Particle Swarm Optimization: Theory, Techniques and Applications. Advances in Engineering Mechanics. Nova Publishers, Hauppauge (2011)
2. Campana, E.F., Fasano, G., Pinto, A.: Dynamic system analysis for the selection of parameters and initial population, in Particle Swarm Optimization. J. Global Optim. **48**(3), 347–397 (2010)
3. Chen, X., Diez, M., Kandasamy, M., Zhang, Z., Campana, E.F., Stern, F.: High-fidelity global optimization of shape design by dimensionality reduction, metamodels and deterministic particle swarm. Eng. Optim. (2014). doi:10.1080/0305215X.2014.895340
4. Christopher, W., Cleghorn, C., Engelbrecht, A.: A generalized theoretical deterministic particle swarm model. Swarm Intell. J. **8**(1), 35–59 (2014)
5. Clerc, M., Kennedy, J.: The particle swarm—explosion, stability, and convergence in a multidimensional complex space. IEEE Trans. Evol. Comput. **6**(1), 58–73 (2002)
6. Clerc, M.: Stagnation analysis in particle swarm optimization or what happens when nothing happens. Technical Report. http://hal.archives-ouvertes.fr/hal-00122031 (2006)
7. Conn, A.R., Scheinberg, K., Vicente, L.N.: Introduction to Derivative-Free Optimization. MPS-SIAM Series on Optimization, Philadelphia (2009)
8. Corazza, M., Fasano, G., Gusso, R.: Particle swarm optimization with non-smooth penalty reformulation for a complex portfolio selection problem. Appl. Math. Comput. **224**, 611–624 (2013)

9. Diez, M., Chen, X., Campana, E.F., Stern, F.: Reliability-based robust design optimization for ships in real ocean environment. In: Proceedings of 12th International Conference on Fast Sea Transportation, FAST2013, Amsterdam, The Netherlands (2013)
10. Diez, M., Campana, E.F., Stern, F.: Design-space dimensionality reduction in shape optimization by Karhunen-Loève expansion. Comput. Methods Appl. Mech. Eng. (2014). doi:10.1016/j.cma.2014.10.042
11. Diez, M., He, W., Campana, E.F., Stern, F.: Uncertainty quantification of Delft catamaran resistance, sinkage and trim for variable Froude number and geometry using metamodels, quadrature and Karhunen-Loève expansion. J. Mar. Sci. Technol. 19(2), 143–169 (2014). doi: 10.1007/s0077301302350
12. Fasano, G., Lucidi, S.: A nonmonotone truncated Newton-Krylov method exploiting negative curvature directions, for large scale unconstrained optimization. Optim. Lett. **3**(4), 521–535 (2009)
13. Gazi, V.: Stochastic stability analysis of the particle dynamics in the PSO algorithm. In: Proceedings of the IEEE International Symposium on Intelligent Control, Dubrovnik, Croatia, pp. 708–713. IEEE Press, New York (2012)
14. Griewank, A.: Evaluating Derivatives. SIAM Frontieres in Applied Mathematics, Philadelphia (2000)
15. Hart, W.E.: A stationary point convergence theory for evolutionary algorithms. Foundations of Genetic Algorithms 4, pp. 325–342. Morgan Kaufmann, San Francisco (1996)
16. Hart, W.E.: Evolutionary Pattern Search Algorithms for Unconstrained and Linearly Constrained Optimization. In Proceedings of the Evolutionary Programming VII, pp. 303–312. Springer, Berlin (2001)
17. Kadirkamanathan, V., Selvarajah, K., Fleming, P.: Stability analysis of the particle dynamics in particle swarm optimizer. IEEE Trans. Evol. Comput. **10**(3), 245–255 (2006)
18. Kennedy, J., Eberhart, R.C.: Particle swarm optimization. In: Proceedings of the 1995 IEEE International Conference on Neural Networks (Perth, Australia), vol. IV, pp. 1942–1948. IEEE Service Center, Piscataway, NJ (1995)
19. Kolda, T.G., Lewis, R.M., Torczon, V.: Optimization by direct search: new perspectives on some classical and modern methods. SIAM Rev **45**(3), 385–482 (2003)
20. Lucidi, S., Sciandrone, M.: On the global convergence of derivative-free methods for unconstrained optimization. SIAM J. Optim. **13**, 97–116 (2002)
21. Mendes, R.: Population topologies and their influence in particle swarm performance. Ph.D. Dissertation, University of Minho, Departamento de Informatica Escola de Engenharia Universidade do Minho (2004)
22. Nocedal, J., Wright, S.J.: Numerical Optimization, 2nd edn. Operations Research and Financial Engineering. Springer, New York (2006)
23. Peri, D., Diez, M.: Ship optimization by globally convergent modification of PSO using surrogate-based Newton method. Eng. Comput. **30**(4), 548–561 (2013). doi:10.1108/02644401311329361
24. Poli, R., Kennedy, J., Blackwell, T.: Particle swarm optimization. Swarm Intell. **1**(1), 33–57 (2007)
25. Poli, R.: Analysis of the publications on the applications of particle swarm optimisation. J. Artif. Evol. Appl. article ID 685175, 1–10 (2008)
26. Poli, R.: Mean and variance of the sampling distribution of particle swarm optimizers during stagnation. IEEE Trans. Evol. Comput. **13**(4), 712–721 (2009)
27. Sarachik, P.E.: Principles of Linear Systems. Cambridge University Press, New York (1997)
28. Serani, A., Diez, M., Leotardi, C., Peri, D., Fasano, G., Iemma, U., Campana, E.F.: On the use of synchronous and asynchronous single-objective deterministic Particle Swarm Optimization in ship design problems. In: Proceedings of OPT-i, International Conference on Engineering and Applied Sciences Optimization, Kos Island, Greece (2014)
29. Trelea, I.C.: The particle swarm optimization algorithm: convergence analysis and parameter selection. Inf. Process. Lett. **85**, 317–325 (2003)

30. Van den Bergh, F., Engelbrecht, A.P.: A study of particle swarm optimization particle trajectories. Inf. Sci. **176**, 937–971 (2006)
31. Vaz, A.I.F., Vicente, L.N.: A particle swarm pattern search method for bound constrained global optimization. J. Global Optim. **39**, 197–219 (2007)
32. Vaz, A.I.F., Vicente, L.N.: PSwarm: a hybrid solver for linearly constrained global derivative-free optimization. Optim. Methods Softw. **24**, 669–685 (2009)
33. Volpi, S., Diez, M., Gaul, N.J., Song, H., Iemma, U., Choi, K.K., Campana, E.F., Stern, F.: Development and validation of a dynamic metamodel based on stochastic radial basis functions and uncertainty quantification, Structural Multidisciplinary Optimization (2014). doi:10.1007/s00158-014-1128-5

Fireflies in the Fruits and Vegetables: Combining the Firefly Algorithm with Goal Programming for Setting Optimal Osmotic Dehydration Parameters of Produce

Raha Imanirad and Julian Scott Yeomans

Abstract This study employs the Firefly Algorithm (FA) to determine the optimal parameter settings needed in the osmotic dehydration process of fruits and vegetables. Two case studies are considered. For both cases, the functional form of the osmotic dehydration model is established using response surface techniques with the resulting optimization formulations being non-linear goal programming models. For optimization purposes, a computationally efficient, FA-driven method is employed and the resulting solutions are shown to be superior to those from previous approaches for the osmotic process parameters. The final component of this study provides a computational experimentation performed on the FA to illustrate the relative sensitivity of this nature-inspired metaheuristic approach over a range of the two key parameters that most influence its running time.

Keywords Firefly algorithm · Non-linear goal programming · Process parameter optimization · Osmotic dehydration · Papaya · Mushrooms

1 Introduction

The worldwide agricultural production of fruits and vegetables is a multi-trillion dollar enterprise. The annual global production of papayas and mushrooms currently exceeds 12 million tonnes and 6 million tonnes, respectively [1]. As with

R. Imanirad
Technology and Operations Management, Harvard Business School,
Boston, MA 02163, USA
e-mail: rimanirad@hbs.edu

J.S. Yeomans (✉)
OMIS Area, Schulich School of Business, York University, Toronto,
ON M3J 1P3, Canada
e-mail: syeomans@schulich.yorku.ca

X.-S. Yang (ed.), *Recent Advances in Swarm Intelligence and Evolutionary Computation*,
Studies in Computational Intelligence 585, DOI 10.1007/978-3-319-13826-8_3

many agricultural commodities, the high moisture content of both papayas and mushrooms renders them highly perishable and, due to various microbial, enzymatic and chemical reactions, they start to deteriorate immediately upon harvesting [2, 3]. Therefore, it becomes imperative to determine effective preservation methods that maintain the overall quality of the product. This is frequently accomplished through various combinations of drying using dehydration and heat processing techniques [4]. The drying of fruits and vegetables permits longer storage periods, reduces shipping weights, and minimizes their packaging requirements [2]. However, conventionally hot-air dried products using vacuum, cabinet or tray dryers have not achieved universal endorsement due to the poor resulting product quality [2, 5, 6].

Consequently, osmotic dehydration has been introduced as an alternative preservation technique that can yield a higher quality end product [7]. In the process of osmotic dehydration, fruits and vegetables are placed into a hypertonic solution where water is drawn out of the produce and into the solution due to the differences in their concentrations [7]. In this fashion, osmotic dehydration removes a proportion of the water content in the produce leading to a product of intermediate moisture content [8, 9]. A simultaneous transfer of solid materials—such as sugar and salt—also occurs between the product and the solution [7, 10, 11]. Beneficially, the actual osmotic process contributes only minimal thermal degradation to the nutrients due to the relatively low temperature water removal process in comparison to standard hot air drying techniques [7, 12–14].

Osmotic dehydration of fresh produce can also be used as a pre-treatment to supplemental dry-processing because it improves many sensory, functional and nutritional properties [15]. The quality of the subsequent product is superior to one without pre-treatment due to (i) increases in the solid gain transfer of sugar and salt from the solution, (ii) the improvements to texture of the fruits and vegetables, and (iii) the stability of the colour pigmentation during storage [2, 12]. Thus, in conjunction with other follow-on drying technologies, osmotic dehydration produces a superior quality, shelf-stable product for both local consumption and export markets.

Water removal during the dehydration process is influenced by many factors such as type and concentration of osmotic agents, temperature, circulation/agitation of solution, solution-to-sample ratio, thickness of food material, and pre-treatment [7]. While an expanding market currently exists for osmo-convective dehydrated fruits and vegetables in both domestic and world markets, only limited efforts have been undertaken to optimize the osmotic process parameters [2, 12, 16]. Specifically, an analysis of the mass transport occurring within the osmosis process measured in terms of water loss and solid (sugar, salt) gains is of considerable practical relevance [6, 7, 10].

In this study, the functional form of the osmotic dehydration model is established using a standard response surface technique [16–19]. The format of the resulting optimization model is shown to be a non-linear goal programming problem [16]. This study investigates the effects of using a procedure that employs a Firefly Algorithm (FA) [20–22] to determine the optimal osmotic parameters for the

dehydration cases of papaya introduced in [12] and of mushrooms in [2]. It can be shown that the resulting solutions for the osmotic process parameters determined by the FA are superior to those from the previous approaches. The final portion of the study revolves around an extensive computational experimentation performed on the FA using the osmotic dehydration models from these two cases to determine the relative sensitivity of this nature-inspired metaheuristic over a range of the two key parameters that most influence its running time.

2 Functional Form and Mathematical Model of the Osmotic Dehydration Process

The first portion of the study examines the dehydration case of papaya from [12]. The initial stage requires the construction of an appropriate model of the responses to the three main osmotic process parameters—(i) solution temperature, (ii) syrup solution concentration and (iii) duration of osmosis—on the water loss and sugar gain of the papaya. This functional representation can then be used to predict the water loss and sugar gain impacts in the papaya over the requisite experimental ranges of the three parameters. Once an appropriate model has been determined, the next step is to optimize this model in order to find the maximum water loss and the optimum sugar gain achieved during the osmotic dehydration process. In the subsequent formulations, let T represent the syrup solution temperature in °C, C be the syrup solution concentration in °Brix, and D be the duration of the osmosis measured in hours. For the response variables, let WL be the percentage of water loss and SG represent the solid gain of the product during the dehydration process. In this instance, SG corresponds to the percentage of sugar gain in the papaya.

Response surface procedures are statistical techniques frequently used for optimization in empirical studies [17–19]. Response surfaces employ quantitative data in appropriately designed experiments to simultaneously ascertain the various variable relationships within multivariate problems [18]. The equations constructed describe the effect of various test variables on responses, determine interrelationships among the test variables and represent the combined effect of all test variables in any response. Response surfaces enable an experimenter to undertake an efficient exploration of a process or system [18, 19]. These approaches have frequently been used in the optimization of food processes [2, 12, 23–26] and will, consequently, be employed in this study to determine the appropriate mathematical representation. The proposed model can then be used to predict the water loss and sugar gain in the dehydration of papaya over the different experimental ranges for the process durations, syrup concentrations and syrup solution temperatures.

For the osmotic dehydration process, it should be noted that the exact mathematical representation for the relationship between the parameters remains unknown. Thus a response surface methodology enables an empirical approximation to it using efficient experimental design techniques [18, 19]. The specific

Table 1 Response surface experimental design layout for 3 variables and 3 levels

Treatment No.	Level for T	Temperature (°C)	Level for C	Concentration (°Brix)	Level for D	Duration (h)
1	1	50	1	70	0	5
2	1	50	−1	50	0	5
3	−1	30	1	70	0	5
4	−1	30	−1	50	0	5
5	1	50	0	60	1	6
6	1	50	0	60	−1	4
7	−1	30	0	60	1	6
8	−1	30	0	60	−1	4
9	0	40	1	70	1	6
10	0	40	1	70	−1	4
11	0	40	−1	50	1	6
12	0	40	−1	50	−1	4
13	0	40	0	60	0	5
14	0	40	0	60	0	5
15	0	40	0	60	0	5

testing design actually contains the three variables (T, C, D) each set at three levels using the data taken from [12] in order to determine the corresponding water loss (WL) and sugar gain (SG) responses. The design for the various combinations of input variables and levels requires the fifteen experimental combinations shown in Table 1 (see [12]).

The corresponding experimental values for the response variables WL and SG appear in last two columns of Table 2.

Based upon the response surface experimental design appropriately applied to the water loss and the sugar gain outputs of Table 2 [17–19], the functional equations empirically determined for responses are:

$$WL = 63.745 - 1.56275T - 0.6615C - 6.075D + 0.0286T^2 + 0.00925C^2 + 0.79D^2 \quad (1)$$

$$SG = 13.90875 - 0.830275T - 0.044875C + 0.51249D + 0.01058T^2 + 0.002825TC. \quad (2)$$

Reference [12] established organoleptic ranges for the osmotic dehydration parameters and restricted their search for best parameter settings to values within these ranges. Organoleptic properties refer to sensory aspects of food including taste, sight, smell, touch, dryness, moisture content, and stale-fresh factors. In order to find values for the osmotic dehydration parameters, [12] constructed a number of contour plots by varying the values of the three variables and observed the effect that these had on their response functions. By superimposing the various contours

Table 2 Experimental data for water loss and sugar gain under different treatments

Temperature (°C)	Concentration (°Brix)	Duration (h)	Water loss (%)	Sugar gain (%)
50	70	5	44.5	8.1
50	50	5	35.2	5.5
30	70	5	31.7	4.5
30	50	5	23.6	3.0
50	60	6	44.5	8.2
50	60	4	39.6	7.0
30	60	6	27.2	3.9
30	60	4	23.2	2.5
40	70	6	37.8	4.8
40	70	4	34.8	4.3
40	50	6	28.4	4.4
40	50	4	25.7	3.4
40	60	5	29.7	4.3
40	60	5	30.0	4.3
40	60	5	30.2	4.4

Table 3 Best osmotic dehydration parameters determined by reference [12]

Temperature (°C)	Concentration (°Brix)	Duration (h)	Water loss (%)	Sugar gain (%)
37	60	4.25	28	4.0

onto a single figure, they visually determined best values for the temperature, concentration, and duration as 37 °C, 60 °Brix and 4.25 h, respectively. These settings invoke responses of 28 % for the water loss and 4.0 % for the sugar gain (see Table 3).

3 A Goal Programming Formulation for Setting Osmotic Dehydration Parameters

The determination of the parameters settings can be viewed as a multi-response optimization process and could, therefore, be transformed into a corresponding mathematical programming model [16]. In this section, this formulation will be accomplished by converting the parameter setting process into an equivalent goal programming format.

Table 4 Ranges for process variables and response goals in the osmotic dehydration

Parameter	Goal	Requirement	Lower limit	Upper limit	Relative importance
Temperature (°C)	1	Minimize	30	50	Important
Concentration (°Brix)	2	Minimize	50	70	Important
Duration (h)	3	Minimize	4	6	Important
Water loss (%)	4	Maximize	23.02	44.05	Very important
Sugar gain (%)	5	Target = 4.0	2.56	8.1	Extremely important

Based upon the organoleptic requirements established for the parameters and response functions by [12], the technical constraints for the problem can be specified as:

$$23.02 \leq WL \leq 44.5$$
$$2.56 \leq SG \leq 8.1.$$
$$30 \leq T \leq 50$$
$$50 \leq C \leq 70$$
$$4 \leq D \leq 6$$

Additional organoleptic preferences can be applied to the responses and variables for the solution. The targets for these desired criteria are summarized in Table 4. From a hierarchical preference attainment perspective, several of these criteria can be recognized as more important attributes to achieve than the others. Namely, from a dehydration perspective, the water loss should be as high as possible within the indicated range, while from a taste perspective, the sugar gain needs to be as close to 4 % as possible. The relative importance for the achievement of these hierarchy targets is indicated in the last column of Table 4.

Hence, from a mathematical perspective, each of these desired targets can be specified as a definitive goal and the entire formulation can then be transformed into a conventional goal programming problem. An objective function that appropriately penalizes deviations from the desired targets must be created and, in the subsequent mathematical programming formulation, a percentage deviation objective weighted by the relative importance of each goal is employed. Consequently, the problem of determining osmotic dehydration parameter values can be transformed into the following non-linear goal programming formulation.

$$\text{Minimize} \quad W_1 * P_1 + W_2 * P_2 + W_3 * P_3 + W_4 * N_4 + W_5 * (P_5 + N_5)$$

subject to

$$\begin{aligned}
P_1 &= T-30 \\
N_1 &= 50-T \\
P_2 &= C-50 \\
N_2 &= 70-C \\
P_3 &= D-4 \\
N_3 &= 6-D \\
N_4 &= 44.05-WL \\
P_4 &= WL-23.02 \\
N_5 &= 4.00-SG \\
P_5 &= SG-4.00 \\
N_6 &= 8.1-SG \\
P_6 &= SG-2.56 \\
P_i \geq 0,\ N_i &\geq 0 \quad i = 1, 2, 3, 4, 5, 6
\end{aligned}$$

In order to complete the transformation of the problem into the series of defined goals, several additional deviation variables have been introduced. Namely, for the goal model, define P_i and N_i, i = 1–6, to be the positive and negative deviations, respectively, from the disparate goal targets and constraint limits shown for the variables in Table 4. Let W_i correspond to weighting factors applied to goal i, i = 1–5, to reflect the relative importance in achieving that goal's target. Each W_i also contains the appropriate denominator constant required to transform the deviation variables into the requisite percentage deviation value format. Thus, solving the goal programming model would be equivalent to determining optimal parameter values for the osmotic dehydration process.

4 A Goal Programming, Firefly Algorithm-Driven Optimization Approach

Although several alternative solution approaches could have been applied to the resulting optimization problem, the approach actually employed uses the FA. For optimization purposes, [21] has demonstrated that the FA is more computationally efficient than other such commonly-used metaheuristics as genetic algorithms, simulated annealing, and enhanced particle swarm optimization. Hence, an FA approach is a very computationally efficient procedure. While this section briefly outlines the FA procedure, more detailed descriptions appear in [21] and [20].

Objective Function $F(\boldsymbol{X})$, $\boldsymbol{X} = (x_1, x_2, \ldots x_d)$
Generate the initial population of n fireflies, $\boldsymbol{X}_i$, $i = 1, 2, \ldots, n$
Light intensity I_i at $\boldsymbol{X}_i$ is determined by $F(\boldsymbol{X}_i)$
Define the light absorption coefficient γ
while (t < MaxGeneration)
for $i = 1: n$, all n fireflies
for $j = 1: n$, all n fireflies (inner loop)
if ($I_i < I_j$), Move firefly i towards j; **end if**
Vary attractiveness with distance r via $e^{-\gamma r}$
end for j
end for i
Rank the fireflies and find the current global best solution $\boldsymbol{G}^*$
end while
Postprocess the results

Fig. 1 Pseudo code of the firefly algorithm

The FA is a biologically-inspired, population-based metaheuristic with each firefly in the population representing a potential solution to the problem. An FA procedure employs three idealized rules: (i) All fireflies within a population are unisex, so that one firefly will be attracted to other fireflies irrespective of their sex; (ii) Attractiveness between fireflies is proportional to their brightness, implying that for any two flashing fireflies, the less bright one will move towards the brighter one; and (iii) The brightness of a firefly is determined by the value of its objective function. For a maximization problem, the brightness can be considered proportional to the value of the objective function. Reference [21] demonstrates that the FA approaches the global optima whenever the number of fireflies $n \rightarrow \infty$ and the number of iterations t, is set so that $t \gg 1$. In reality, the FA has been shown to converge extremely quickly into both local and global optima [20, 21]. The basic operational steps of the FA are summarized in Fig. 1 [21].

In the FA, there are two important issues to resolve: the variation of light intensity and the formulation of attractiveness. For simplicity, it can always be assumed that the attractiveness of a firefly is determined by its brightness which in turn is associated with the encoded objective function. In the simplest case, the brightness of a firefly at a particular location $\boldsymbol{X}$ would be its calculated objective value $F(\boldsymbol{X})$. However, the attractiveness, β, between fireflies is relative and will vary with the distance r_{ij} between firefly i and firefly j. In addition, light intensity decreases with the distance from its source, and light is also absorbed in the media, so the attractiveness should be allowed to vary with the degree of absorption. Consequently, the overall attractiveness of a firefly can be defined as

$$\beta = \beta_0 \exp(-\gamma r^2) \tag{3}$$

Table 5 Optimal process parameters determined for the osmotic dehydration of papaya

	Temperature (°C)	Concentration (°Brix)	Duration (h)	Water loss (%)	Sugar gain (%)
Reference [12]	37	60	4.25	28	4.0
FA solution	37.776	70	4	32.8	4.02

where β_0 is the attractiveness at distance $r = 0$ and γ is the fixed light absorption coefficient for a specific medium. If the distance r_{ij} between any two fireflies i and j located at $\boldsymbol{X}_i$ and $\boldsymbol{X}_j$, respectively, is calculated using the Euclidean norm, then the movement of a firefly i that is attracted to another more attractive (i.e. brighter) firefly j is determined by

$$\boldsymbol{X}_i = X_i + \beta_0 \exp(-\gamma(r_{ij})^2)\left(X_i - X_j\right) + \alpha\varepsilon_i. \tag{4}$$

In this expression of movement, the second term is due to the relative attraction and the third term is a randomization component. Reference [21] indicates that α is a randomization parameter normally selected within the range [0, 1] and $\boldsymbol{\varepsilon}_i$ is a vector of random numbers drawn from either a Gaussian or uniform (generally [–0.5, 0.5]) distribution. It should be pointed out that this expression is a random walk biased toward brighter fireflies and if $\beta_0 = 0$, it becomes a simple random walk. The parameter γ characterizes the variation of the attractiveness and its value determines the speed of the algorithm's convergence. For most applications, γ is typically set between 0.1 and 10 [21]. For all computational approaches for the FA considered in this study, the variation of attractiveness parameter γ is fixed at 5 while the randomization parameter α is initially set at 0.6, but is then gradually decreased to a value of 0.1 as the procedure approaches its maximum number of iterations (see [21]).

By optimizing the goal programming problem formulation using the FA-driven procedure, best process parameters for the osmotic dehydration of the papaya were determined and these resulting values are displayed in Table 5. In comparison to the values found by [12], it can be observed that the syrup concentration has increased by 10 °Brix, the duration of dehydration process has been reduced slightly by 0.25 h, while the temperature parameter remains essentially the same. More importantly, in terms of the key responses, the water loss has increased by 5 %, while the sugar gain has remained at the highly desirable target of 4 %. Consequently, since the water loss response—which is obviously the fundamental feature of the osmotic dehydration process—has been increased significantly from that determined by [12], this goal programming solution represents a significant improvement.

In any given optimization problem, for a very large number of fireflies $n \gg k$ where k is the number of local optima, the initial locations of the n fireflies should be distributed as uniformly as possible to ensure that a comprehensive search throughout the search domain occurs. As the FA proceeds, the fireflies should converge into all of the local optima, including the global ones. By comparing the best solutions among all these optima, the global optima can easily be

determined. As noted above, the FA approaches the global optima whenever the number of fireflies $n \rightarrow \infty$ and the number of iterations t, is set so that $t \gg 1$ [21]. In reality, the FA has a tendency to converge very quickly into both local and global optima [20, 21, 27].

In general, the two parameters that most directly impact the solution search time of the FA are the values selected for n and t. Using terminology from computational complexity, the running time for the FA is linear in t, but is second order polynomial in n. Obviously, for practical applications, the desire is to be able to determine the best solution in the shortest amount of time. This would correspond to setting n and t at the minimum possible values that produce the best solution(s). However, since the FA's search process incorporates random components within its solution search, the parameter setting is clearly not a strictly deterministic issue—determining appropriate values for n and t reflects a component of choice on the part of the decision-maker.

Consequently, for the dehydration of papaya case, an ensuing, post-optimization sensitivity analysis was performed to investigate the impact for different combinations of the number of fireflies, n, and the number of iterations, t, on the solution quality. Specifically, the value of the firefly parameter was set at n = 20, 50, 100, 150, 200, 250, 250, 500 and the value for the number of iterations was set at t = 100, 250, 500, 1,000, 1,500, 2,500. For 30 runs of each parametric combination of fireflies and iterations, the corresponding responses for the water loss and sugar gain were recorded. The average values of these responses over the 30 runs per combination are provided in Table 6 and visual representation of these values appears in Figs. 2 and 3, respectively.

As might have been reasonable to anticipate a *priori*, it is interesting to observe that more consistent solutions (i.e. where the average values are closer to optimal) are obtained as both the number of fireflies and the number of iterations increases. Namely larger values of n or t produce on average the actual optimal solution, while combinations involving smaller parameter values show more solution variability in

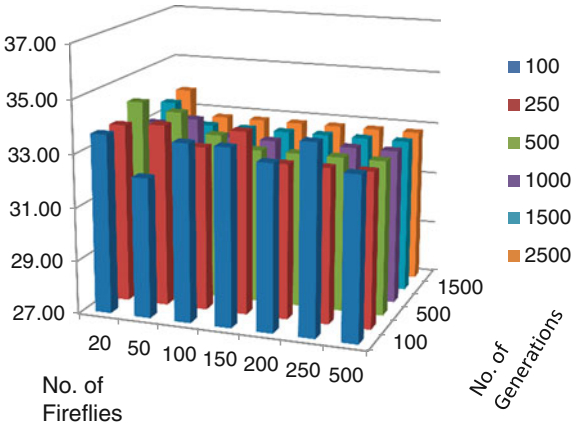


Fig. 2 Average water loss (%) in the papaya for different parameter settings of the firefly algorithm

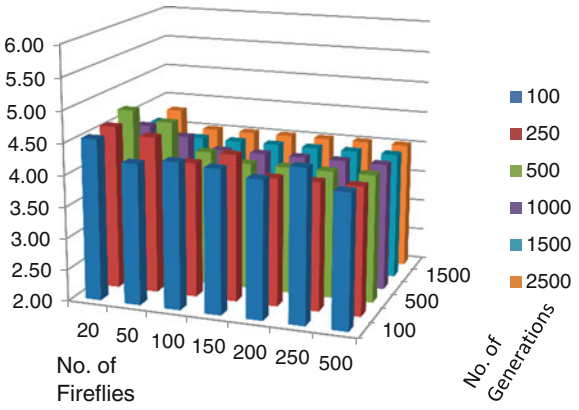


Fig. 3 Average sugar gain (%) in the papaya for different parameter settings of the firefly algorithm

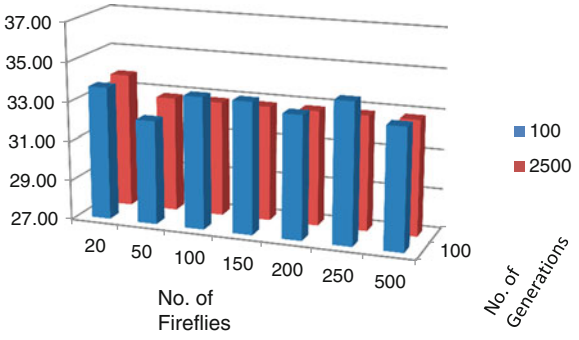


Fig. 4 Comparison of average water loss (%) in the papaya from runs of 100 generations and 2,500 generations in the firefly algorithm

both the water loss and sugar gain. While there are multiple approaches that can be undertaken to parse these results, Figs. 4 and 5 provide comparisons of the average water loss and sugar gain responses obtained for the minimum and maximum number of iterations considered in the experimentation. From Figs. 4 and 5, it can be observed that at t = 2,500, the FA always produces the optimal water loss and sugar gain solution, on average, for any number of fireflies other than n = 20 (i.e. the FA always generated the optimal solution in each of the 30 runs). Conversely, at t = 100, the average water loss and sugar gain values indicate that there can be variability in the quality of the solution obtained irrespective of the number of fireflies employed in the FA process. This indicates that the more iterations used, the better the solution quality obtained by the FA.

Similar to the preceding analysis, Figs. 6 and 7 provide a comparison of the average water loss and sugar gain responses obtained for the minimum and

Table 6 Average sugar gain (%) and water loss (%) for different Parameter settings of the firefly algorithm

No. of fireflies	Number of iterations											
	100		250		500		1,000		1,500		2,500	
	SG	WL	SG	WL	SG	WL	SG	WL	SG	WL	SG	WL
20	4.55	33.71	4.60	33.71	4.72	34.25	4.32	33.12	4.25	33.60	4.30	33.79
50	4.23	32.24	4.49	33.83	4.57	33.98	4.19	33.38	4.03	32.81	4.02	32.80
100	4.31	33.65	4.14	33.14	4.15	33.23	4.02	32.80	4.02	32.80	4.02	32.80
150	4.27	33.63	4.32	33.85	4.02	32.80	4.02	32.79	4.02	32.80	4.02	32.80
200	4.17	33.21	4.02	32.79	4.02	32.80	4.02	32.80	4.02	32.80	4.02	32.80
250	4.41	34.08	4.02	32.80	4.02	32.80	4.02	32.80	4.02	32.79	4.02	32.80
500	4.11	33.10	4.02	32.80	4.02	32.80	4.02	32.80	4.02	32.80	4.02	32.80

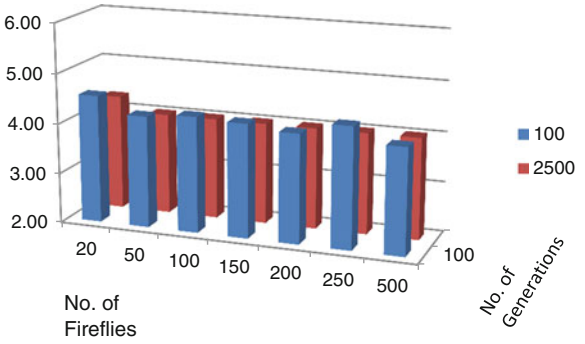
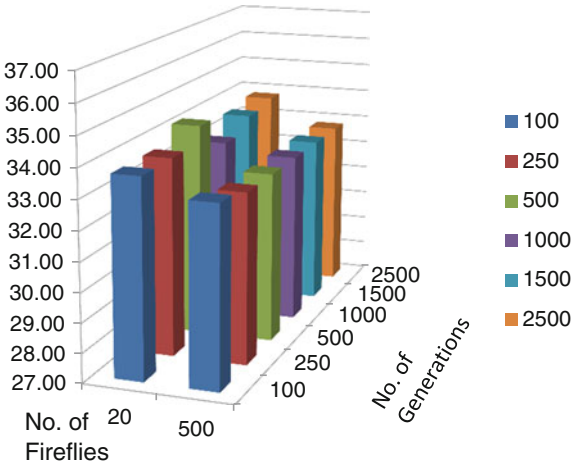


Fig. 5 Comparison of average sugar gain (%) in the papaya from runs of 100 generations and 2,500 generations in the firefly algorithm

Fig. 6 Comparison of average water loss (%) in the papaya from runs with 20 fireflies and runs with 500 fireflies



maximum number of fireflies considered. From these Figures, it can be seen that at $n = 500$ fireflies, the FA always produces the optimal water loss and sugar gain solutions, on average, for any number of iterations other than $t = 100$. Furthermore, even at $t = 100$, the average solution is extremely close to the optimal solution. Conversely, at $n = 20$ fireflies, the average water loss and sugar gain values show that there can be considerable variability in the quality of the solution obtained irrespective of the number of iterations employed in the FA. These findings clearly illustrate that the more fireflies used in the FA, the better the solution quality.

It should be further reinforced that Table 6 shows the average response values for each combination of n and t. While the FA in the larger parameter value combinations always converged to the overall optimal solution, the smaller combinations would also frequently produce this optimal value within the set of the 30 runs. However, there would also be occasions where divergent solution values were

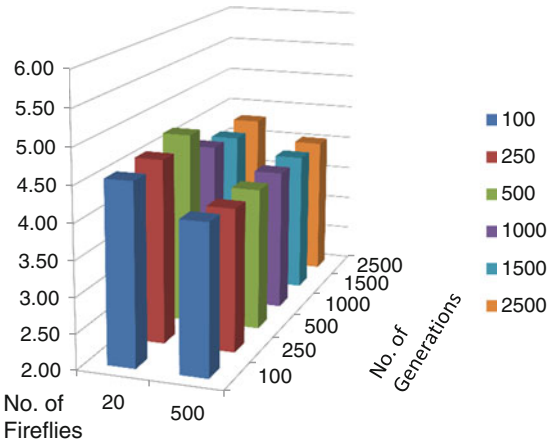


Fig. 7 Comparison of average sugar gain (%) in the papaya between runs with 20 fireflies and runs with 500 fireflies

found within the runs, thereby distorting the overall averages. Given the running time complexities for the FA, a combination of a relatively smaller value of n combined with a relatively larger value for t would be preferable from both a solution time and solution accuracy perspective. Table 6 shows that even the intermediate values in the experimental ranges considered for n and t tend to consistently produce very high quality solutions. The experimentation for this specific problem would indicate that the value for t needs to be set somewhere in the range of 500–1,000 iterations, while the value for n should be between 100 and 150 fireflies if calculating the true optimal solution is always required.

5 Mathematical Model and Optimization of Mushroom Dehydration

The preceding FA-driven goal programming optimization approach was subsequently reapplied to an analogous case for the osmotic dehydration of mushrooms taken from [2]. In this case, a brine solution is employed for the mushroom dehydration and the resulting solid gain involves the transport of salt into the mushrooms. The three main osmotic process parameters are the brine solution temperature, the salt concentration of the brine and the duration period for the osmosis. The two measured responses are now the water loss and salt gain of the mushrooms. Since the two cases overlap considerably and in order to retain as much consistency for direct comparisons between the cases as possible, the models derived and the variables employed will remain essentially identical in the analysis. Thus, the variables used are T for the brine solution temperature in °C, C representing the salt concentration measured in percent, and D as the duration of osmosis

Table 7 Response surface design, data, and responses under different treatment levels

Level for T	Temperature (°C)	Level for C	Concentration (%)	Level for D	Duration (Min)	Water loss (%)	Salt gain (%)
1	55	1	20	0	45	44.93	3.24
1	55	−1	10	0	45	36.38	1.03
−1	35	1	20	0	45	39.70	2.56
−1	35	−1	10	0	45	29.92	0.59
1	55	0	15	1	60	43.92	2.90
1	55	0	15	−1	30	34.23	2.24
−1	35	0	15	1	60	37.09	2.34
−1	35	0	15	−1	30	29.54	1.73
0	45	1	20	1	60	45.04	3.03
0	45	1	20	−1	30	35.51	2.22
0	45	−1	10	1	60	33.69	1.06
0	45	−1	10	−1	30	26.18	0.33
0	45	0	15	0	45	38.05	2.57
0	45	0	15	0	45	38.44	2.64
0	45	0	15	0	45	38.27	2.64
0	45	0	15	0	45	38.55	2.79
0	45	0	15	0	45	38.60	2.82

in minutes. The corresponding response variables are *WL* for the percentage of water loss and *SG* as the percentage of salt gain of the mushrooms.

As with the papaya analysis, the specific response surface design for mushroom dehydration contains each of the variables (T, C, D) set at three levels using the data taken from [2] in order to construct the corresponding water loss (WL) and salt gain (SG) response functions. The design for the various combinations of input variables and levels requires the experimental combinations shown in Table 7 (see [2]), while the values determined for the response variables *WL* and *SG* appear in last two columns of Table 7.

Based upon the response surface design [17–19] applied to the water loss and salt gain outputs from Table 7, the corresponding equations determined for the responses are:

$$WL = 19.58 - 0.13T + 1.7C + 0.98D + 0.00357TD + 0.00673CD - 343C^2 - 0.0106D^2 \tag{5}$$

$$SG = -13.87 + 0.11T + 1.09C + 0.14D - 0.000973T^2 - 0.0296C^2 - 0.00129D^2. \tag{6}$$

Table 8 Ranges for process variables and response goals in the osmotic dehydration

Parameter	Goal	Requirement	Lower limit	Upper limit	Relative importance
Temperature (°C)	1	Minimize	35	55	Important
Concentration (%)	2	Minimize	10	20	Important
Duration (mins)	3	Minimize	30	60	Important
Water loss (%)	4	Maximize	23.02	44.05	Very important
Salt gain (%)	5	Target = 2.98	0.33	3.24	Very important

Reference [2] established appropriate organoleptic ranges for the osmotic dehydration parameters (see Table 8). Following the approach of [2, 12] designed a number of contour plots of the osmotic dehydration parameters by varying the values of the T, C and D, and then observing the effect that these variations had on the response functions. By superimposing these contours onto a single chart, the best settings for the temperature, concentration, and duration variables were determined to be 44.89 °C, 16.53 % and 47.59 min, respectively. These settings generate responses of 40.55 % for water loss and 2.98 % for salt gain (see Table 9). Using the organoleptic ranges established for the parameters and response functions, the technical constraints for the problem become:

$$26.18 \leq WL \leq 45.04$$
$$0.33 \leq SG \leq 3.24.$$
$$35 \leq T \leq 55$$
$$10 \leq C \leq 20$$
$$30 \leq D \leq 60$$

The stated goals for the desired organoleptic criteria are categorized in Table 8. In terms of hierarchical goal satisfaction, the water loss needs to be as high as possible, while the salt gain should be as close to 2.98 % as possible. The relative importance for the achievement of these hierarchical goals is indicated in the last column of Table 8.

Once again, the determination of osmotic dehydration parameters can be transformed into a non-linear goal programming problem. To complete the transformation into a goal model, define P_i and N_i, i = 1 to 6, to be the positive and negative deviations, respectively, from the disparate goal targets and constraint limits shown for the variables in Table 8. Let W_i correspond to weighting factors

Table 9 Optimal process parameters determined for the osmotic dehydration of mushrooms

	Temperature (°C)	Concentration (%)	Duration (mins)	Water loss (%)	Salt gain (%)
Reference [2]	44.89	16.53	47.59	40.55	2.98
FA solution	54.043	19.031	46.777	45.04	2.98

applied to goal i, $i = 1$ to 5, to reflect the relative importance in achieving that goal's target. As before, each W_i contains an appropriate denominator that transforms the deviation measures into a percentage deviation format. The resulting goal programming formulation for the osmotic dehydration of mushrooms can, therefore, be transformed into the following model.

$$\text{Minimize} \quad W_1 * P_1 + W_2 * P_2 + W_3 * P_3 + W_4 * \ N_4 + W_5 * (P_5 + N_5)$$

subject to

$$\begin{aligned}
P_1 &= T - 35 \\
N_1 &= 55 - T \\
P_2 &= C - 10 \\
N_2 &= 20 - C \\
P_3 &= D - 30 \\
N_3 &= 60 - D \\
P_4 &= WL - 26.18 \\
N_4 &= 45.04 - WL \\
P_5 &= SG - 2.98 \\
N_5 &= 2.98 - SG \\
P_6 &= SG - 0.33 \\
N_6 &= 3.24 - SG \\
P_i &\geq 0,\ N_i \geq 0 \quad i = 1, 2, 3, 4, 5, 6
\end{aligned}$$

Optimizing the goal programming model using the FA-driven procedure, provided the best process parameters for the osmotic dehydration of the mushrooms shown in Table 9. In comparison to the values found by [2], it can be observed that the salt concentration has increased by 2.5 %, the required temperature must be increased by 9 °C, while the duration of dehydration should remain essentially unchanged. In terms of the resulting responses to these parameter settings determined by the FA, the water loss increases by 4.5 %, while the salt gain remains at the desired organoleptic target of 2.98 %. Since the water loss has increased significantly, this goal programming solution provides a significant improvement to that found in [2].

As with the papaya case, an ensuing, post-optimization sensitivity analysis was performed to examine the impact of different combinations of the number of fireflies, n, and the number of iterations, t, on the solution quality. For 30 runs of each parametric combination, the corresponding average responses for the water loss and salt gain were recorded. The average values of these response outputs over the 30 runs per combination are provided in Table 10 and visual representations of these responses appear in Figs. 8 and 9, respectively. Again, as might have been reasonably foreseen, results closer to the optimum are more consistently observed

Table 10 Average salt gain (%) and water loss (%) for different parameter settings of the firefly algorithm

No. of fireflies	Number of Iterations											
	100		250		500		1,000		1,500		2,500	
	SG	WL	SG	WL	SG	WL	SG	WL	SG	WL	SG	WL
20	3.00	45.01	2.69	42.60	2.89	43.24	2.93	43.67	2.88	45.03	2.93	45.03
50	3.04	45.04	2.89	45.04	2.96	45.04	2.96	45.04	2.94	45.04	2.90	45.03
100	2.94	45.01	2.95	45.03	3.02	45.04	3.03	45.04	2.95	45.04	2.98	45.04
150	3.00	45.04	2.93	45.04	2.92	45.03	2.92	45.04	3.01	45.04	2.93	45.04
200	3.00	45.04	2.92	45.04	2.92	45.04	2.94	45.04	3.01	45.03	2.98	45.04
250	2.98	45.04	2.99	45.03	3.03	45.04	2.98	45.04	2.98	45.04	3.01	45.04
500	3.04	45.03	2.94	45.04	2.99	45.03	2.93	45.04	2.99	45.04	3.03	45.04

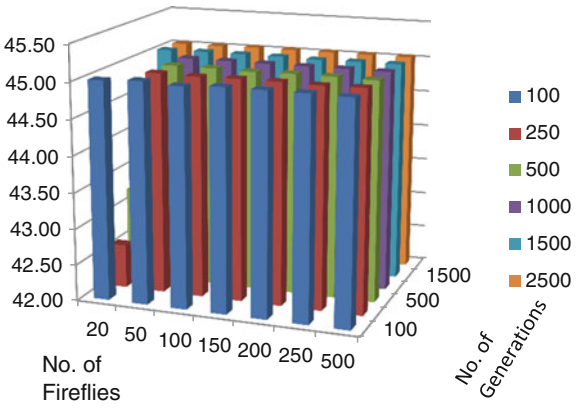


Fig. 8 Average water loss (%) in the mushrooms for different parameter settings of the firefly algorithm

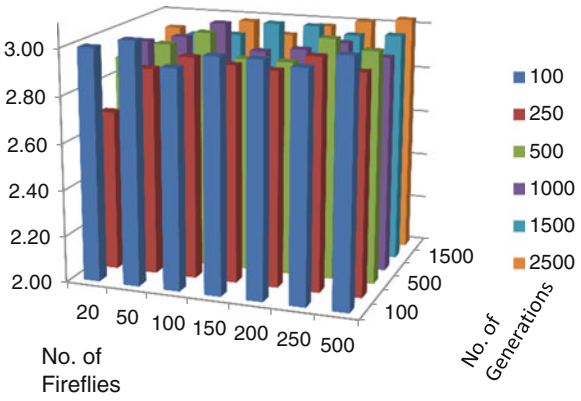


Fig. 9 Average salt gain (%) in the mushrooms for different parameter settings of the firefly algorithm

when either the number of fireflies or the number of iterations are set at the higher end of the ranges considered, although most parameter combinations produce extremely good solutions on average.

6 Conclusions

In this study, the functional form of the osmotic dehydration responses for papaya and mushrooms were established using an empirical response surface approach and the formats of the resulting optimization models for both cases were formulated as a non-linear goal programming problems. Subsequently, the optimal osmotic drying solutions to the goal programming problems were determined using an FA-directed algorithm. The osmotic process parameters found using the FA were superior to the solutions found in all previous instances. Computational experimentation on the osmotic dehydration models highlighted the relative sensitivity of the nature-inspired

FA over the key running-time parameters of the number of fireflies and the number of iterations. This experimentation demonstrated that for intermediate-to-high values of either of the two key parameters, the FA would always determine overall optimal solutions, while lower values of either parameter produced greater variability in the solution quality. Since the running time complexity of the FA is linear in the number of iterations but polynomial in the number of fireflies, this experimentation would seem to confirm that it would be more computationally practicable to implement an FA using a relatively larger number of iterations together with a "reasonable" number of fireflies than vice versa. Since an FA can clearly be modified to solve a wide variety of "real world" problem types beyond the realm of fruits and vegetables, the practicality of this approach can clearly be extended into numerous other "real world" applications. These extensions will become the focus of future research.

References

1. Geohive (2014) Geohive World Crop Production www.geohive.com/charts/ag_crops.aspx
2. Mehta, B.K., Jain, S.K., Sharma, G.P., Mugdal, V.D., Verma, R.C., Doshi, A., Jain, H.K.: Optimization of osmotic drying parameters for button mushroom (Agaricus bisporus). Food Sci. Technol. **3**(10A), 1298–1305 (2012)
3. Venturini, M.E., Reyes, J.E., Rivera, C.S., Oria, R., Blanco, D.: Microbiological quality and safety of fresh cultivated and wild mushrooms commercialized in Spain. Food Microbiol. **28** (8), 1492–1498 (2011)
4. Ranganna, S.: Handbook of Analysis and Quality Control for Fruits and Vegetable Products. Tata McGraw Hill Publishing, New Delhi (1986)
5. Jain, S.K., Verma, R.C.: Osmotic dehydration: A new, promising and emerging industry. Beverage Food World **30**(1), 30–34 (2003)
6. Rosa, M.D., Giroux, F.: Osmotic treatments and problems related to the solution management. J. Food Eng. **49**(3), 223–236 (2001)
7. Rastogi, N.K., Raghavarao, K.S.M.S., Niranjan, K., Knorr, D.: Recent developments in osmotic dehydration: Method to enhance mass transfer. Food Sci. Technol. **13**(1), 48–59 (2002)
8. Hawkes, J., Fink, J.M.: Osmotic concentration of fruit slices prior to dehydration. Food Process. Preserv. **2**(4), 265–267 (1978)
9. Shukla, B.D., Singh, S.P.: Osmo-convective drying of cauliflower, mushroom and green pea. Food Eng. **80**(2), 741–747 (2007)
10. Nieto, A., Castro, M.A., Alzamora, A.: Kinetics of moisture transfer during air drying of blanched and/or osmotically dehydrated mango. J. Food Eng. **50**(2), 175–185 (2001)
11. Tonon, R.V., Baroni, A.F., Hubinges, M.D.: Osmotic dehydration of tomato in ternary solutions: Influence of process variables on mass transfer kinetics and an evaluation of the retention of arytenoids. Food Eng. **82**(4), 509–517 (2007)
12. Jain, S.K., Verma, R.C., Murdia, L.K., Jain, H.K., Sharma, G.P.: Optimization of process parameters for osmotic dehydration of papaya cubes. Food Sci. Technol. **48**(2), 211–217 (2011)
13. Kar, A., Gupta, D.K.: Osmotic dehydration characteristics of button mushrooms. J. Food Sci. Technol. **38**(4), 352–357 (2001)
14. Sodhi, N.S., Singh, N., Komal, K.: Osmotic dehydration kinetics of carrots. J. Food Sci. Technol. **43**(4), 374–376 (2006)

15. Torreggiani, D., Bertolo, G.: Osmotic pretreatments in fruit processing: chemical, physical and structural effects. J. Food Eng. **49**(30), 247–253 (2001)
16. Yeomans, J.S., Yang, X.S.: Determining optimal osmotic drying parameters using the firefly algorithm. International conference on applied operational research (ICAOR), Vancouver, Canada, 29–31 July (2014a)
17. Box, G.E., Behnken, D.W.: Some new three level designs for the study of quantitative three variables. Technometrics **2**(4), 455–475 (1960)
18. Montgomery, D.C.: Design and Analysis of Experiments, 4th edn. Wiley, New York (1997)
19. Myers, R.H., Montgomery, D.C.: Response Surface Methodology : Process and Product Optimization Using Designed Experiments. Wiley, New York (1995)
20. Imanirad, R., Yang, X.S., Yeomans, J.S.: Modelling-to-generate-alternatives via the firefly algorithm. J. Appl. Oper. Res. **5**(1), 14–21 (2013)
21. Yang, X.S.: Nature-Inspired Metaheuristic Algorithms, 2nd edn. Luniver Press, Frome (2010)
22. Yeomans, J.S., Yang, X.S.: Municipal waste management optimization using a firefly algorithm-driven simulation-optimization approach. Int. J. Process Manage. Benchmarking (4 (4), 363–375, 2014b)
23. Alam, M.S., Singh, A., Sawhney, B.K.: Response surface optimization of osmotic dehydration process for anola slices. Food Sci. Technol. **47**(1), 47–54 (2010)
24. Mudhar, G.S., Toledo, R.T., Floros, J.D., Jen, J.J.: Optimization of carrot dehydration process using response surface methodology. J. Food Sci. **54**(11), 714–719 (1989)
25. Shi, L., Xue, C.H., Zhao, Y., Li, Z.J., Wang, X.Y., Luan, D.L.: Optimization of processing parameters of horse mackerel (Trachurus japonicus) dried in a heat pump dehumidifier using response surface methodology. Food Eng. **87**(1), 74–81 (2008)
26. Uddin, M.B., Amsworth, P., Ibanoglu, S.: Evaluation of mass exchange during osmotic dehydration of carrots using response surface methodology. Food Eng. **65**(4), 473–477 (2004)
27. Yeomans, J.S.: Simulation-driven optimization in waste management facility expansion planning. J Comput. Methods Sci. Eng. **12**(1/2), 111–127 (2012)

Hybrid Metaheuristic Algorithms: Past, Present, and Future

T.O. Ting, Xin-She Yang, Shi Cheng and Kaizhu Huang

Abstract Hybrid algorithms play a prominent role in improving the search capability of algorithms. Hybridization aims to combine the advantages of each algorithm to form a hybrid algorithm, while simultaneously trying to minimize any substantial disadvantage. In general, the outcome of hybridization can usually make some improvements in terms of either computational speed or accuracy. This chapter surveys recent advances in the area of hybridizing different algorithms. Based on this survey, some crucial recommendations are suggested for further development of hybrid algorithms.

Keywords Bio-inspired · Diversity · Evolutionary algorithms · Hybrid algorithms · Metaheuristics · Nature-inspired algorithms

1 Introduction

Hybrid algorithms are two or more algorithms that run together and complement each other to produce a profitable synergy from their integration [1]. These algorithms are commonly known as hybrid metaheuristics (HMs) [2, 3].

T.O. Ting (✉) · K. Huang
Department of Electrical and Electronic Engineering, Xi'an Jiaotong-Liverpool University, Suzhou, Jiangsu Province, China
e-mail: toting@xjtlu.edu.cn

K. Huang
e-mail: kaizhu.huang@xjtlu.edu.cn

X.-S. Yang
School of Science and Technology, Middlesex University, The Burroughs, London NW4 4BT, UK
e-mail: x.yang@mdx.ac.uk

S. Cheng
Division of Computer Science, The University of Nottingham, Ningbo, Zhejiang Province, China
e-mail: shi.cheng@nottingham.edu.cn

X.-S. Yang (ed.), *Recent Advances in Swarm Intelligence and Evolutionary Computation*, Studies in Computational Intelligence 585, DOI 10.1007/978-3-319-13826-8_4

The hybridization of EAs is popular, partly due to its better performance in handling noise, uncertainty, vagueness, and imprecision [4, 5]. For simplicity here, instead of using HM, we prefer to use the general term *hybrid algorithms* to refer to the similar notion.

There are in fact two prominent issues of EAs in solving global and highly nonconvex optimization problem. These are:

(i) *Premature convergence*: The problem of premature convergence results in the lack of accuracy of the final solution. The final solution is a feasible solution close to the global optimal, often regarded as satisfactory or close-to-optimal solution.
(ii) *Slow convergence*: Slow convergence means the solution quality does not improve sufficiently quickly. It shows stagnation or almost flat on a convergence graph (either a single iteration or the average of multiple iterations).

These two issues above are often partly related to the solution diversity that an algorithm can produce in the searching process. In nature, the diversity is maintained by the variety (quality) and abundance (quantity) of organisms at a given place and time [6], and this principle is applicable to EAs. At the beginning of a search process, usually diversity is high, and it decreases as the population may move towards the global optimum. High diversity may provide better guarantee to find the optimal solution with better accuracy, but this will usually lead to slow convergence, and thus there are some tradeoffs between convergence and accuracy. On the other hand, low diversity may lead to fast convergence while sacrificing the guarantee to find global optimality and with poor solution accuracy. This scenario is illustrated in Fig. 1. Hereby, we call the intersection between the convergence rate and accuracy as the *tradeoff point*. Obviously, this is an ideally approach, and the real-world is much grayer and things are not clearly cut.

It is worth pointing out that the diversity is one factor, which is related to the more general concept of exploration and exploitation. High diversity encourages exploration, and low diversity does not necessarily mean exploitation because exploitation requires the use of landscape information and the information extracted from the population during the search process. Even with sufficient diversity, there is no guarantee to solve the convergence problem because convergence is a much complicated issue. Simply balancing the exploration and exploitation may make an algorithm work to its best capability, but this does not necessarily mean its convergence rate is high. Good convergence requires clever exploitation at the right time and at the right place, which is still an open problem.

In addition, it is widely known that one prominent factor for premature convergence is the lack of diversity. Therefore, in order to escape and jump out of local optima, proper diversity should be maintained during the search process, even with the expense of slower convergence. To enable this, hybridization has been a widely acceptable strategy to promote diversity along the search for the global optimum.

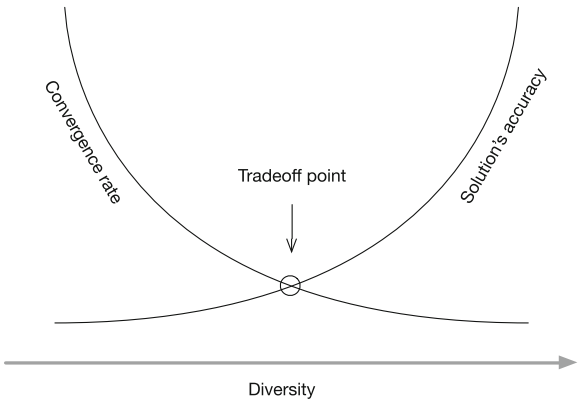


Fig. 1 Compromising accuracy and convergence rate

In the rest of this chapter, we will briefly review the past, present and future of the hybridization strategies used to promote solution diversity in combining algorithms. Obviously, it is not possible to cover everything in a single chapter, and thus we will focus on some key points concerning the latest developments.

2 Hybrid Algorithms

Hybrid algorithms are very diverse, and they form a long list of algorithms and variants. Here, we only briefly touch the tip of the algorithm iceberg.

2.1 The Past

Evolutionary algorithms (EAs) [7, 8] are stochastic global optimizers that mimic the metaphor of biological evolution. They are almost always population-based algorithms that learn from the past searches by using a group of individuals or agents. These algorithms often usually possess behaviour inspired by social or biological behaviour in the natural world. Loosely speaking, there are three categories of EAs, which are:

(i) Evolutionary Programming (EP) [9]
(ii) Evolutionary Strategies (ES) [10], and
(iii) Genetic Algorithms (GAs) [11]

These algorithms were among the first to offer great advantages in terms of locating the global optimality in vast and complex search spaces, especially when gradient information is unavailable. The implementation of these algorithms are

relatively straightforward and easy, based upon simple and easy to understand concepts. They are also reasonably flexible as parameters can be changed easily for better performance.

There were many hybrid algorithms or variants about various evolutionary algorithms. However, key issues such as slow convergence or premature convergence still exist. In addition, these algorithms can be computationally extensive and requires a lot of iterations for highly complex problems.

2.2 The Present

The present developments tend to provide some improvements based on the extensive developments in last few decades, and researchers are still actively trying to design new hybrid algorithms. For example, in [1], Rodriguez et al. developed hybrid metaheuristic by integrating an EA with Simulated Annealing (SA). In their review, they found that there were at about 312 publications indexed by ISI Web of Science that utilized both EA and SA algorithms. In comparison, there were only 123 publications that hybridized EAs with other metaheuristics such as the greedy search, iterated local search, descent gradient, and tabu search. However, Rodriguez et al.'s survey was limited to EAs and SA methods.

In the current literature, hybrid algorithms seem widely developed. Using Particle Swarm Optimization (PSO) as an example [12], the combination of PSO with other auxiliary search techniques seems very effective in improving its performance [13]. Genetic algorithm hybrids (or use of genetic operators with other methods) are by far the most widely studied. Genetic operators such as selection, crossover, and mutation have been integrated into PSO to produce better candidates [14]. Differential evolution [15], ant colony optimization [16] and also conventional local search techniques have been used to combine with PSO [17].

In addition, to avoid locating previously detected solutions, techniques such as deflection, sketching, repulsion [18], self-organizing, and dissipative methods [19] have also been used in the hybrid PSO algorithms. Some biology-inspired operators such as niche [20] and specification technique [21] are introduced into PSO to prevent the swarm from crowding too closely and to locate as many good solutions as possible. A cellular particle swarm optimization, in which a cellular automata mechanism is integrated in the velocity update to modify the trajectories of particles, is proposed in [22].

PSO is just one example, and other hybrid algorithms concerning differential evolution and simulated annealing are also widely studied. The current trends seem the hybridization of the conventional/well-established new algorithms. For example, the new eagle strategy has been hybridized with differential evolution and improved performance has been achieved [23].

2.3 The Future

Many new algorithms have been developed in recent years. For example, the bio-inspired algorithms such as Artificial Bee Colony Algorithm (ABC), Bat Algorithm (BA), Cuckoo Search (CS), Firefly Algorithm (FA), Flower Pollination Algorithm (FPA), Glowworm Swarm Algorithm (GlowSA), Hunting Search Algorithm (HSA), Eagle Strategy (ES), Roach Infestation Optimization (RIO), Gravitational Search Algorithm (GravSA), Artificial Fish School Algorithm (AFS), Bacterial Evolutionary Algorithm (BEA), Artificial Plant Optimization Algorithm (APO), Krill Herd Algorithm (KHA) and others [24].

The list is expanding rapidly. These algorithms may possess entities and some novel characteristics for hybridization that remain to be discovered in the near future. However, it is worth pointing out that simple, random hybridization should not be encouraged. In a keynote talk by Xin-She Yang at the 15th EU workshop in metaheuristics and engineering (15th EU/ME) in Istanbul, Turkey in 2014, Yang warned about the danger of random hybridization. Suppose there are n algorithms, if one chooses $2 \leq k \leq n$ to produce a hybrid, there will be

$$C_n^k = \binom{n}{k} = \frac{n!}{k!(n-k)!},$$

possible combinations. For $n = 20$ and $k = 2$, there will be 190 hybrids, while for $n = 20$ and $k = 10$, there will be 184,756 hybrids, which might produce 184,756 random (and almost useless) hybrids. Considering other combinations of $k = 2, 3, \ldots, 19$, there will be about $2^n = 1{,}048{,}576$ hybrids. This estimate comes from the sum of binomial coefficients

$$\sum_{k=0}^{n} \binom{n}{k} = 2^n.$$

As a further joke here, for any four algorithms such as PSO, Grass Colony Optimization, Donkey Algorithm, Blue Sky Algorithm (purely random names), any sensible researchers should not produce PSO-Donkey-BlueSky Algorithm, or Grass-PSO-Donkey algorithm, or BlueSky-Grass-Donkey Algorithm! No one should ever do it (except a future robot hitchhiker from another Galaxy).

Research efforts should focus on truly novel, insightful and efficient approaches. Novel techniques and approaches should be based on mathematical theory and insightful analysis of the main mechanisms in algorithms so that new hybrid algorithms should truly provide more effective ways of problem-solving to large-scale, real-world applications.

As it is really challenging to produce a good hybrid, we will try to summarize some observations and developments concerning hybrid algorithms in a very informal manner in the rest of this chapter.

3 Motivations for Hybridization

In a hybrid algorithm, two or more algorithms are collectively and cooperatively solving a predefined problem. In some hybrids, one algorithm may be incorporated as a sub-algorithm to locating the optimal parameters for another algorithm, while in other cases, different components of algorithms such mutation and crossover are used to improve another algorithm in the hybrid structure. With regards to this nature, hybrid algorithms can loosely be divided into two categories:

(i) *Unified purpose hybrids*. Under this category, all sub-algorithms are utilized to solve the same problem directly; and different sub-algorithms are used in different search stages. Hybrid metaheuristic algorithms with local search is a typical example. The global search explores the search space, while the local search is utilized to refine the areas that may contain the global optimum.
(ii) *Multiple purpose hybrids*. One primary algorithm is utilized to solve the problem, while the sub-algorithm is applied to tune the parameters for the primary algorithm. For example, PSO can be applied to find the optimal value of mutation rate in GAs. Hereby, PSO is not solving the problem, but assisting in finding better solutions by searching for the optimal parameter for better performance. The hyper-heuristic algorithms can be regarded as a kind of hybrid methods. In hyper-heuristic methods, parameters are selected (by a sub-algorithm or via a learning mechanism) [25].

4 Taxonomy of Hybrid Algorithms

Generally speaking, hybrid algorithms can be grouped into two categories, which are described in the following subsections.

4.1 Collaborative Hybrids

This involves the combination of two or more algorithms running either in sequential or parallel. The contributing weight of each participating algorithm can be regarded as half and half in the simplest case. The possible frameworks of the hybrid algorithms under this category are illustrated in Fig. 2. Three structures are depicted in this figure, which are:

(i) *Multi-stage*. There are two stages involved in this case. The first algorithm acts as the global optimizer whereas the second algorithm performs local search. This category can fit well into the framework of the eagle strategy described by Yang in [26]. The first algorithm is capable of exploring the search space

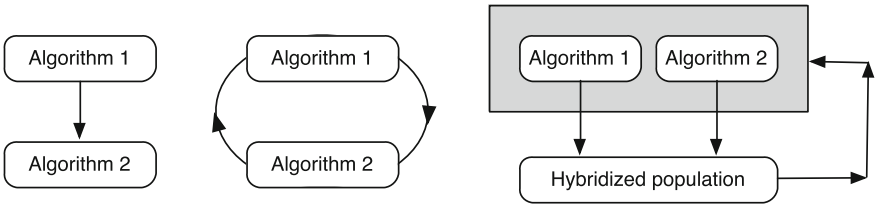


Fig. 2 Collaborative framework of hybrid algorithm, depicting multi-stage, sequential, and parallel structures

globally to locate promising area of convergence. Then the second algorithm will perform intensive local search such as hill-climbing and downhill simplex method [27]. A challenging issue in such an implementation is to know when to switch to the second algorithm. Measures such as diversity should be incorporated to assist in the switch criteria. Previous works in [28, 29] utilized Genetic Algorithm as the global optimizer (first algorithm), with Particle Swarm Optimization (PSO) as the local searcher (second algorithm).

(ii) *Sequential*. In this structure, both algorithms are run alternatively until one of the convergence criteria is met. For simplicity, both algorithms will be run for similar number of iterations before proceeding to the next algorithm.

(iii) *Parallel.* Two algorithms are run simultaneously, manipulating on the same population. One of the algorithms may be executed on a pre-specified percentage of an algorithm.

4.2 Integrative Hybrids

In this aspect, one algorithm is regarded as a subordinate, embedded in a master metaheuristic. For this category, the contributing weight of the secondary algorithm is approximately 10–20 %. This involves incorporation of a manipulating operator from a secondary algorithm into a primary algorithm. For example, many algorithms utilized the mutation operator from GA into PSO, resulted in so called Genetic PSO or Mutated PSO. Others may incorporate the gradient techniques such as hill-climbing, steepest descent, and Newton-Raphson into the primary algorithm. Examples include works published in [15, 19, 30].

Under this category, Fig. 3 illustrates the two possible approaches:

(i) *Full manipulation*. The entire population is manipulated at every iteration. Such operation can be integrated inline with the existing source code, usually as a subroutine/subfunction.

(ii) *Partial manipulation*. In this manipulation, only a portion of the entire population is accelerated using local search methods such as gradient methods. Choosing the right portion and the right candidate to be accelerated pose a great challenge in assuring the success of this hybrid structure.

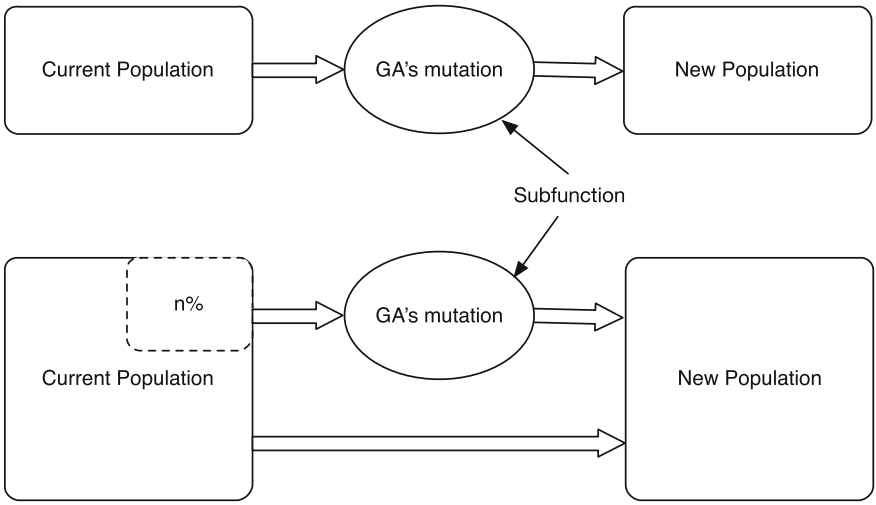


Fig. 3 Integrative structure of a hybrid algorithm, with full and partial manipulations

5 Disadvantages and Challenges of Hybrid Algorithms

Although hybrid algorithms offers great advantage of increasing the diversity in a population and hence enhancing the search capability of the developed hybrid algorithm, some drawbacks do exist, which will be discussed in the following subsections.

5.1 Naming Convention

The inclusion of another algorithm usually leads to a naming issue. Some researchers adopt very different names to their hybrid algorithms. For instance, the GAAPI algorithm [6] is an acronym for Hybrid Ant Colony-Genetic Algorithm, which is a bit confusing to other researchers. A hybrid name such as HPSO-BFGS [31] seems to be a tedious abbreviation, which is harder to read. In comparison to both architectures mentioned in Sect. 4, the collaborative type of hybrid algorithm seems to create more sophisticated names. For example, it may be interesting to compare the names of Hybrid GA-PSO (collaborative) to Mutated PSO (integrative), though those two hybrids combined GA with PSO.

5.2 Complexity of Hybrid Algorithm

In terms of algorithm architecture, the hybridization process usually creates extra components in the overall architecture of the hybrid algorithm. This increases the

complexity of the hybrid algorithm. Due to a more complicated structure, hybrid algorithms have some resistance to be accepted by researchers. In the literature, two popular hybrid algorithms are Hybrid Taguchi-Genetic Algorithm [32] and Hybrid Genetic Algorithm [33], with 305 and 294 citations (from Scopus [34]), both published in 2004. From the citations, they seem to be received well. It is interesting to note that both algorithms fall in the *integrative* type of hybrid algorithm, which have simpler taxonomy/architecture.

5.3 Computational Speed

In many works, hybrid algorithms seem to improve results in terms of the overall convergence speed and accuracy. However, these convergence graphs are often plotted with respect to the number of iterations. This simply means that the faster convergence does not mean the true convergence rate because the hybrid usually uses a higher number of (internal or implicit) iterations. For example, for collaborative (sequential type) hybrid algorithm such as GA-PSO, a cycle, or one iteration comprises GA and PSO. For a fair comparison, this should be considered as two cycles instead of one in the convergence graph. To avoid this issue, the final run time should be utilized as a metric when comparing a hybrid algorithm with non-hybrid algorithms. Besides, due to a more complicated architecture in hybrid algorithms, the overhead arises alongside its complexity, often unavoidable. This affects the overall performance and thereby truncates its robustness. The time consumed by overheads should be taken into account for a fair comparison. Again, this is possible by recording the true number of iterations taken to reach a pre-specified target, though time complexity should be compared as well.

There are other issues concerning hybrid algorithms. For example, most hybrid algorithms will increase the number of parameters in the algorithms, thus making it harder to tune their parameters. I addition, the complicated structure of a hybrid usually makes it harder to be analyzed, and thus gaining little insight into the reasons why such hybrids work. Furthermore, hybrid algorithms are slightly harder to implement, and thus more prone to errors. Thus, care should be taken when interpreting results from hybrid algorithms.

6 Examples of Hybrid Algorithms

In this section, we focus on some recent hybrid algorithms that have been performed on a wide range of numerical benchmark problems for global optimization. These hybrid algorithms are grouped either under collaborative (Table 1) or integrative (Table 2) categories. The prescription for both categories have been elaborated in Sect. 4.

Table 1 Recent collaborative hybrid algorithms, published after 2010

Abbreviation	Full name
GAAPI	Hybrid ant colony-genetic algorithm (GAAPI) [6]
HPSO-BFGS	PSO-broyden-fletcher-goldfarb-shanno [31]
PSO-ACO	Particle swarm optimization-ant colony optimization [16]
DE-BBO	Hybrid differential evolution-biogeography based optimization [35]
HABCDE	Hybrid artificial bee colony-differential evolution [36]

Table 2 Recent integrative hybrid algorithms, published after 2010

Abbreviation	Full name
CPSO	Cellular particle swarm optimization [22]
TGA	Taguchi genetic algorithm [32]
OGA	Orthogonal genetic algorithm [37]
HABC	Hybrid artificial bee colony [38]
HMA	Hybrid memetic algorithm [39]
HCS	Hybrid cuckoo search [40]
TC-ABC	Taguchi chaos-artificial bee colony [41]
CLA-DE	Cellular learning automata [42]
HPSO	Hybrid particle swarm optimization [43]
HDE	Hybrid differential evolution [44]

As discussed in Sect. 5.1, collaborative algorithms (in Table 1) tend to have longer names compared to their integrative counterparts, as listed in Table 2. Also, from our analysis above, the number of journals published for the latter category is almost twice, compared to the first category. This means that the integrative hybrids are more widely accepted by researchers due to the simplicity in their development process.

In reality, hybrid algorithms have been proven successful in solving a wide range of applications. These include a myriad of different areas, such as neural network [45], cluster analysis [33, 46], telecommunications [47, 48], scheduling [49], protein structure analysis [50], image processing [51, 52], power system [28, 29, 53, 54] and many others.

7 Recommendations for Future Developments

From the above analysis and observations, we can highlight some insights that should be useful to any future developments in this area as follows:

(i) Simpler algorithm are preferred than more complex algorithms. Einstein once said: “Everything should be made as simple as possible, but not simpler.” In the same sense, algorithms should be made as simple as possible. In reality,

people tend to use a robust algorithm that has simpler architecture for the ease of implementation and is yet efficient to be useful to real-world applications.

(ii) Shorter names are much preferred in the scientific community. Names should be as short as possible and 3–4 letters for the abbreviation.

(iii) New hybrids (either collaborative or integrative hybrids) should have a clear structure that is easier for implementations. Ideally, any combination should be based on clear thinking, novel feature and insightful mechanisms, which is more likely to produce better hybrids in the long run.

8 Conclusions

In this chapter, we reviewed a wide range of hybrid algorithms and investigated the motivations of their developments. We have also categorized these algorithms, based on hybridization techniques. In addition, some drawbacks were discussed concerning hybridization. Recent examples of hybrid algorithm from the literature have been presented, with a brief summary of some prominent applications. Finally, some suggestions were recommended that can be useful to the future developments of hybrid algorithms.

Acknowledgments The work is supported by National Natural Science Foundation of China (NSFC) under grant No. 61473236, 61273367; and Ningbo Science & Technology Bureau (Project No. 2012B10055).

References

1. Rodriguez, F., Garcia-Martinez, C., Lozano, M.: Hybrid metaheuristics based on evolutionary algorithms and simulated annealing: taxonomy, comparison, and synergy test. Evol. Comput. IEEE Trans. **16**, 787–800 (2012)
2. Talbi, E.-G.: A taxonomy of hybrid metaheuristics. J. Heuristics **8**(5), 541–564 (2002)
3. Raidl, G.: A unified view on hybrid metaheuristics. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 4030 LNCS, pp. 1–12 (2006)
4. Grosan, C., Abraham, A.: Hybrid evolutionary algorithms: methodologies, architectures, and reviews. Hybrid evolutionary algorithms, pp. 1–17. Springer, Berlin (2007)
5. Preux, P., Talbi, E.-G.: Towards hybrid evolutionary algorithms. Int. Trans. Oper. Res. **6**(6), 557–570 (1999)
6. Ciornei, I., Kyriakides, E.: Hybrid ant colony-genetic algorithm (gaapi) for global continuous optimization. Syst. Man Cybern. Part B Cybern. IEEE Trans. **42**, 234–245 (2012)
7. Back, T., Fogel, D.B., Michalewicz, Z.: Handbook of evolutionary computation. IOP Publishing Ltd., London (1997)
8. Eiben, A.E., Smith, J.E.: Introduction to evolutionary computing. Springer, Berlin (2003)
9. Fogel, D.B. Evolutionary computation: toward a new philosophy of machine intelligence, Vol. 1, John Wiley & Sons (2006)

10. Beyer, H.-G., Schwefel, H.-P.: Evolution strategies—a comprehensive introduction. Nat. Comput. **1**(1), 3–52 (2002)
11. Goldberg, D.E.: Genetic algorithms in search, optimization, and machine learning. Addison-Wesley Longman, Boston (1989)
12. Eberhart, R., Kennedy, J.: A new optimizer using particle swarm theory. In: Proceedings of the Sixth International Symposium on Micro Machine and Human Science, pp. 39–43 (1995)
13. Cheng, S.: Population diversity in particle swarm optimization: definition, observation, control, and application. Ph.D. thesis, Department of Electrical Engineering and Electronics, University of Liverpool, Liverpool (2013)
14. Angeline, P.J.: Using selection to improve particle swarm optimization. In: Proceedings of the 1998 Congress on Evolutionary Computation (CEC 1998), pp. 84–89 (1998)
15. Zhang, W.-J., Xie, X.-F.: DEPSO: hybrid particle swarm with differential evolution operator. In: Proceedings of the IEEE International Conference on Systems, Man and Cybernetics (SMC 2003), vol. 4, pp. 3816–3821 (2003)
16. Shelokar, P.S., Siarry, P., Jayaraman, V.K., Kulkarni, B.D.: Particle swarm and ant colony algorithms hybridized for improved continuous optimization. Appl. Math. Comput. **188**, 129–142 (2007)
17. Liang, J.J., Suganthan, P.N.: Dynamic multi-swarm particle swarm optimizer with local search. In: Proceedings of 2005 IEEE Congress on Evolutionary Computation (CEC 2005), vol. 1, pp. 552–528 (2005)
18. Parsopoulos, K.E., Vrahatis, M.N.: On the computation of all global minimizers through particle swarm optimization. IEEE Trans. Evol. Comput. **8**, 211–224 (2004)
19. Xie, X.F., Zhang, W.J., Yang, Z.L.: A dissipative particle swarm optimization. In: Proceedings of the Fourth Congress on Evolutionary Computation (CEC 2002), vol. 2, pp. 1456–1461 (2002)
20. Brits, R., Engelbrecht, A.P., van den Bergh, F.: Locating multiple optima using particle swarm optimization. Appl. Math. Comput. **189**, 1859–1883 (2007)
21. Parrott, D., Li, X.: Locating and tracking multiple dynamic optima by a particle swarm model using speciation. IEEE Trans. Evol. Comput. **10**, 440–458 (2006)
22. Shi, Y., Liu, H., Gao, L., Zhang, G.: Cellular particle swarm optimization. Inf. Sci. **181**, 4460–4493 (2011)
23. Yang, X.-S., Deb, S.: Two-stage eagle strategy with differential evolution. Int. J. Bio-Inspired Comput. **4**, 1–5 (2012)
24. Yang, X.S., Cui, Z., Xiao, R., Gandomi, A.H., Karamanoglu, M. (eds.): Swarm intelligence and bioinspired computation: theory and applications. Newnes (2013)
25. Burke, E.K., Gendreau, M., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., Qu, R.: Hyper-heuristics: a survey of the state of the art. J. Oper. Res. Soc. **64**, 1695–1724 (2013)
26. Yang, X.-S., Karamanoglu, M., Ting, T.O., Zhao, Y.-X.: Applications and analysis of bio-inspired eagle strategy for engineering optimization. Neural Comput. Appl. pp. 1–10 (2013)
27. Yang, X.-S., Ting, T.O., Karamanoglu, M.: Random walks, lévy flights, markov chains and metaheuristic optimization. In Future information communication technology and applications, pp. 1055–1064, Springer, Netherlands (2013)
28. Ting, T.O., Wong, K.P., Chung, C.Y.: A hybrid genetic algorithm/particle swarm approach for evaluation of power flow in electric network. Lect. Notes Comput. Sci. **3930**, 908–917 (2006)
29. Ting, T.O., Wong, K.P., Chung, C.Y.: Investigation of hybrid genetic algorithm/particle swarm optimization approach for the power flow problem. In: Proceedings of 2005 International Conference on Machine Learning and Cybernetics, 2005, vol. 1, pp. 436–440, IEEE (2005)
30. Varnamkhasti, M., Hassan, N.: A hybrid of adaptive neuro-fuzzy inference system and genetic algorithm. J. Intell. Fuzzy Syst. **25**(3), 793–796 (2013)
31. Li, S., Tan, M., Tsang, I., Kwok, J.-Y.: A hybrid pso-bfgs strategy for global optimization of multimodal functions. Syst. Man Cybern. Part B: Cybern. IEEE Trans. **41**, 1003–1014 (2011)
32. Tsai, J.-T., Liu, T.-K., Chou, J.-H.: Hybrid taguchi-genetic algorithm for global numerical optimization. IEEE Trans. Evol. Comput. **8**(4), 365–377 (2004)

33. Oh, I.-S., Lee, J.-S., Moon, B.-R.: Hybrid genetic algorithms for feature selection. IEEE Trans. Pattern Anal. Mach. Intell. **26**(11), 1424–1437 (2004)
34. Scopus: www.scopus.com. Last checked Aug (2014)
35. Gong, W., Cai, Z., Ling, C.: De/bbo: a hybrid differential evolution with biogeography-based optimization for global numerical optimization. Soft. Comput. **15**(4), 645–665 (2011)
36. Xiang, W., Ma, S., An, M.: Habcde: a hybrid evolutionary algorithm based on artificial bee colony algorithm and differential evolution. Appl. Math. Comput. **238**, 370–386 (2014)
37. Leung, Y.-W., Wang, Y.: An orthogonal genetic algorithm with quantization for global numerical optimization. Evol. Comput. IEEE Trans. **5**, 41–53 (2001)
38. Kong, X., Liu, S., Wang, Z., Yong, L.: Hybrid artificial bee colony algorithm for global numerical optimization. J. Comput. Inf. Syst. **8**(6), 2367–2374 (2012)
39. Li, Y., Jiao, L., Li, P., Wu, B.: A hybrid memetic algorithm for global optimization. Neurocomputing **134**, 132–139 (2014)
40. Long, W., Liang, X., Huang, Y., Chen, Y.: An effective hybrid cuckoo search algorithm for constrained global optimization. Neural Comput Appl, 1–16 (2014)
41. Tien, J.-P., Li, T.-H.: Hybrid taguchi-chaos of artificial bee colony algorithm for global numerical optimization. Int. J. Innovative Comput. Inf. Control **9**(6), 2665–2688 (2013)
42. Vafashoar, R., Meybodi, M., Momeni Azandaryani, A.: Cla-de: a hybrid model based on cellular learning automata for numerical optimization. Appl. Intell. **36**(3), 735–748 (2012)
43. Wang, J.: A hybrid particle swarm optimization for numerical optimization. Int. J. Adv. Comput. Technol. **4**(20), 190–196 (2012)
44. Yan, J., Guo, C., Gong, W.: Hybrid differential evolution with convex mutation. J. Soft. vol. 6 (11 SPEC. ISSUE), pp. 2321–2328 (2011)
45. Juang, C.-F.: A hybrid of genetic algorithm and particle swarm optimization for recurrent network design. Syst. Man Cybern. Part B Cybern. IEEE Trans. **34**, 997–1006 (2004)
46. Firouzi, B., Sadeghi, M., Niknam, T.: A new hybrid algorithm based on pso, sa, and k-means for cluster analysis. Int. J. Innovative Comput. Inf. Control **6**(7), 3177–3192 (2010)
47. Xu, Y., Qu, R.: Solving multi-objective multicast routing problems by evolutionary multi-objective simulated annealing algorithms with variable neighbourhoods. J. Oper. Res. Soc. **62** (2), 313–325 (2011)
48. Guo, L., Li, Q., Chen, F.: A novel cluster-head selection algorithm based on hybrid genetic optimization for wireless sensor networks. J. Networks **6**(5), 815–822 (2011)
49. M'Hallah, R.: Minimizing total earliness and tardiness on a single machine using a hybrid heuristic. Comput. Oper. Res. **34**(10), 3126–3142 (2007)
50. Tantar, A.-A., Melab, N., Talbi, E.-G.: A grid-based genetic algorithm combined with an adaptive simulated annealing for protein structure prediction. Soft. Comput. **12**(12), 1185–1198 (2008)
51. Bhandarkar, S., Zhang, H.: Image segmentation using evolutionary computation. IEEE Trans. Evol. Comput. **3**(1), 1–21 (1999)
52. Qureshi, S., Mirza, S., Rajpoot, N., Arif, M.: Hybrid diversification operator-based evolutionary approach towards tomographic image reconstruction. IEEE Trans. Image Process. **20**(7), 1977–1990 (2011)
53. Ting, T.O., Wong, K.P., Chung, C.Y.: Locating type-1 load flow solutions using hybrid evolutionary algorithm. In: 2006 International Conference on Machine Learning and Cybernetics, pp. 4093–4098, IEEE (2006)
54. Ting, T.O., Wong, K.P., Chung, C.: Two-phase particle swarm optimization for load flow analysis. In: IEEE International Conference on Systems, Man and Cybernetics, 2006. SMC'06. vol. 3, pp. 2345–2350, IEEE (2006)

Binary Flower Pollination Algorithm and Its Application to Feature Selection

Douglas Rodrigues, Xin-She Yang, André Nunes de Souza and João Paulo Papa

Abstract The problem of feature selection has been paramount in the last years, since it can be as important as the classification step itself. The main goal of feature selection is to find out the subset of features that optimize some fitness function, often in terms of a classifier's accuracy or even the computational burden for extracting each feature. Therefore, the approaches to feature selection can be modeled as optimization tasks. In this chapter, we evaluate a binary-constrained version of the Flower Pollination Algorithm (FPA) for feature selection, in which the search space is a boolean lattice where each possible solution, or a string of bits, denotes whether a feature will be used to compose the final set. Numerical experiments over some public and private datasets have been carried out and comparison with Particle Swarm Optimization, Harmony Search and Firefly Algorithm has demonstrated the suitability of the FPA for feature selection.

Keywords Feature selection · Flower pollination algorithm · Optimum-path forest

D. Rodrigues · J.P. Papa (✉)
Department of Computing, UNESP, Bauru, SP, Brazil
e-mail: papa@fc.unesp.br

D. Rodrigues
e-mail: douglasrodrigues.dr@gmail.com

X.-S. Yang
School of Science and Technology, Middlesex University, London NW4 4BT, UK
e-mail: x.yang@mdx.ac.uk

A.N. de Souza
Department of Electrical Engineering, UNESP, Bauru, SP, Brazil
e-mail: andrejau@feb.unesp.br

X.-S. Yang (ed.), *Recent Advances in Swarm Intelligence and Evolutionary Computation*,
Studies in Computational Intelligence 585, DOI 10.1007/978-3-319-13826-8_5

1 Introduction

Machine learning techniques have been actively studied in recent years with an increasing number of applications that make use of the so-called intelligence-based decision processes. Roughly speaking, a standard workflow for tackling such problems can be divided in four phases: (i) to preprocess the data (signal or image filtering, for instance); (ii) to extract features; (ii) to train a machine learning technique, and finally (iv) to evaluate its effectiveness over an unseen (test) data [3].

One of the most important steps concerns feature extraction is to find the most important subset of features that leads to the best recognition rates. There are situations we may obtain the same accuracy as before (with the original set of features) even after feature selection, but we can save computational efforts by avoiding extracting some features that are too costly.

Several studies have modeled the problem of feature selection as an optimization task, since the idea is to find out the subset of features that maximizes the accuracy of a given classifier, or minimizes its error over some validating set, for instance. Such approaches can be useful to the application of evolutionary optimization techniques to solve complex tasks. The readers can refer to some recent literature such as the Binary Particle Swarm Optimization (BPSO) [5], Binary Firefly Algorithm (BFA) [4], Binary Harmony Search (BHS) [14], Binary Gravitational Search Algorithm (BGSA) [16], Binary Cuckoo Search (BCS) [17], Binary Charged System Search (BCSS) [19], and Binary Bat Algorithm (BBA) [18].

Yang and Honavar [23] presented a multicriteria Genetic Algorithm (GA) to deal with feature selection, in which the main idea was to optimize both the accuracy and the feature extraction computational costs. Later on, Oh et al. [10] proposed a hybrid GA to tackle the same problem with seemingly better final performance. In addition, there are many papers that address feature selection using other methods such as the ant colonization [2, 7, 21]. The main idea consists of reducing the number of possible paths visited by ants in some works, as well as modified pheromone update rules. Other approaches such as Artificial Bee Colony [9, 20] and Gravitational Search Algorithm [1, 15] have been also employed to the same context.

Basically, the main idea of these methods is to convert the position of the agents (bats, particles, harmonies, etc.) into binary-valued coordinates, which are represented by a string of bits, each denoting the presence or absence of a feature. The problem of feature selection can also be considered as a search task in a boolean lattice, in which the number of dimensions stands for the number of features. As the original versions of most evolutionary optimization techniques were proposed to handle continuous-valued problems, the idea is to apply a discretization function (usually a constrained sigmoid function) to map the agent locations to the boolean lattice.

Very recently, Yang [25, 26] proposed the Flower Pollination Algorithm (FPA), which is inspired by the flower pollination process of flowering plants. This approach has demonstrated interesting results for traditional (continuous-valued) optimization problems, which motivated us to extend it to solve binary optimization tasks. In this case, we now to tackle the problem of feature selection and propose

approach the Binary Flower Pollination Algorithm (BFPA). In regard to the fitness function, we have used a classifier's effectiveness over a validating set: as we need to train a classifier every time an agent (pollen) changes its position, we need a fast classifier. For this purpose, we use the Optimum-Path Forest (OPF) [12, 13], which has demonstrated very promising results in several applications, and this approach is also parameter-independent. The proposed approach has been compared with other methods such as BPSO, BFA and BHS to evaluate several datasets. The results are also analyzed by using statistical tools.

The remainder of this chapter is organized as follows. Section 2 introduces the theory background about FPA and OPF techniques. Sections 3 and 4 present the methodology and the experimental results, respectively. Finally, Sect. 5 draws some conclusions and future works.

2 Theoretical Background

In this section, we first briefly review some of the main concepts and techniques to be used in this chapter, as well as the proposed Binary Flower Pollination Algorithm.

2.1 Flower Pollination Algorithm

The Flower Pollination Algorithm was proposed by Yang [25], inspired by the pollination process of flowering plants. The FPA is governed by four basic rules:

1. Biotic cross-pollination can be considered as a process of global pollination, and pollen-carrying pollinators move in a way that obeys Lévy flights.
2. For local pollination, abiotic pollination and self-pollination are used.
3. Pollinators such as insects can develop flower constancy, which is equivalent to a reproduction probability that is proportional to the similarity of two flowers involved.
4. The interaction or switching of local pollination and global pollination can be controlled by a switch probability $p \in [0, 1]$, slightly biased towards local pollination.

However, it is necessary that the aforementioned basic rules be converted into appropriate updating equations. For example, in the global pollination step, flower pollen gametes are carried by pollinators such as insects, and pollen can travel over a long distance because insects can often fly and move over a much longer range [25]. Therefore, Rules 1 and 3 can be represented mathematically as:

$$x_i^{(t+1)} = x_i^t + \alpha L(\lambda)(g_* - x_i^t), \tag{1}$$

where

$$L(\lambda) = \frac{\lambda \cdot \Gamma(\lambda) \cdot \sin(\lambda)}{\pi} \cdot \frac{1}{s^{1+\lambda}}, \quad s > 0 \tag{2}$$

where x_i^t is the pollen i (solution vector) at iteration t, g_* is the current best solution found among all solutions at the current generation, and α is a scaling factor to control the step size, $L(\lambda)$ is the Lévy flights step size corresponding to the strength of the pollination. In addition, $\Gamma(\lambda)$ stands for the gamma function, and s is the step size. Since insects may move over a long distance with various distance steps, Lévy flights can be used to mimic this characteristic efficiently.

For local pollination, both Rules 2 and 3 can be represented as:

$$x_i^{(t+1)} = x_i^t + \varepsilon(x_j^t - x_k^t), \tag{3}$$

where x_j^t and x_k^t stand for the pollen from different flowers j and k of the same plant species, respectively. This mimics flower constancy in a limited neighbourhood. Mathematically, if x_j^t and x_k^t come from the same species or are selected from the same population, it equivalently becomes a local random walk if ε is drawn from a uniform distribution in [0,1]. In order to mimic the local and global flower pollination, a switch probability (Rule 4) or proximity probability p is used.

2.1.1 Binary Flower Pollination Algorithm

In the standard FPA, the solutions are updated in the search space towards continuous-valued positions. However, in the proposed Binary Flower Pollination Algorithm the search space is modelled as a d-dimensional boolean lattice, in which the solutions are updated across the corners of a hypercube. In addition, as the problem is to select or not a given feature, a binary solution vector is used, where 1 corresponds to that a feature will be selected to compose the new dataset with 0 being otherwise. In order to build this binary vector, we have to use Eq. (5) just after Eq. (3), which can restrict the new solutions to only binary values:

$$S(x_i^j(t)) = \frac{1}{1 + e^{-x_i^j(t)}}, \tag{4}$$

$$x_i^j(t) = \begin{cases} 1 & \text{if } S(x_i^j(t)) > \sigma, \\ 0 & \text{otherwise,} \end{cases} \tag{5}$$

where $\sigma \sim U(0, 1)$. Algorithm 1 presents the proposed BFPA for feature selection using the recognition rate of the OPF classifier as the objective function. Note the proposed approach can be used with any other supervised classification technique.

Lines 1–4 initialize each pollen's position as being a binary string with random values, as well as the fitness value f_i of each individual i. The main loop in Lines

6–27 is the core of the proposed algorithm, in which the inner loop in Lines 7–13 is responsible for creating the new training Z'_1 and evaluating sets Z'_2, and then OPF is trained over Z'_1 and it is used to classify Z'_2. The recognition accuracy over Z'_2 is stored in *acc* and then compared with the fitness value f_i (accuracy) of individual i: if the latter is worse than *acc*, the old fitness value is kept; otherwise, the fitness value is then updated. Lines 12–13 update the best local position of the current pollen. Lines 14–18 update the global optimum, and the last loop (Lines 19–27) moves each pollen to a new binary position restricted by Eq. (5) (Lines 25–27).

Algorithm 1: BFPA - Binary Flower Pollination Algorithm.

input : Training set Z_1 and evaluating set Z_2, α, number of flowers m, dimension d and iterations T.
output : Global best position $\widehat{g}$.
auxiliaries: Fitness vector f with size m and variables *acc*, $maxfit$, $globalfit$ and *maxindex*.

1 **for** *each flower* i $(\forall i = 1,\ldots,m)$ **do**
2 **for** *each dimension* j $(\forall j = 1,\ldots,d)$ **do**
3 $x_i^j(0) \leftarrow \text{Random}\{0,1\}$;
4 $f_i \leftarrow -\infty$;
5 $globalfit \leftarrow -\infty$;
6 **for** *each iteration* t $(t = 1,\ldots,T)$ **do**
7 **for** *each flower* i $(\forall i = 1,\ldots,m)$ **do**
8 Create Z'_1 and Z'_2 from Z_1 and Z_2, respectively, such that both contains only features such that $x_i^j(t) \neq 0, \forall j = 1,\ldots,d$;
9 Train OPF over Z'_1, evaluate its over Z'_2 and stores the accuracy in *acc*;
10 **if** $(acc > f_i)$ **then**
11 $f_i \leftarrow acc$;
12 **for** *each dimension* j $(\forall j = 1,\ldots,d)$ **do**
13 $\widehat{x}_i^j \leftarrow x_i^j(t)$;
14 $[maxfit, maxindex] \leftarrow max(f)$;
15 **if** $(maxfit > globalfit)$ **then**
16 $globalfit \leftarrow maxfit$;
17 **for** *each dimension* j $(\forall j = 1,\ldots,d)$ **do**
18 $\widehat{g}^j \leftarrow x^j_{maxindex}(t)$;
19 **for** *each flower* i $(\forall i = 1,\ldots,m)$ **do**
20 **for** *each dimension* j $(\forall j = 1,\ldots,d)$ **do**
21 $rand \leftarrow \text{Random}\{0,1\}$;
22 **if** $rand < p$ **then**
23 $x_i^j(t) \leftarrow x_i^j(t-1) + \alpha \oplus \text{Lévy}(\lambda)$; **else**
24 $x_i^j(t) \leftarrow x_i^j(t-1) + \varepsilon(x_i^j(t-1) - x_i^k(t-1))$;
25 **if** $(\sigma < \frac{1}{1+e^{x_i^j(t)}})$ **then**
26 $x_i^j(t) \leftarrow 1$; **else**
27 $x_i^j(t) \leftarrow 0$;

2.2 Optimum-Path Forest Classifier

The Optimum-Path Classifier [12, 13] models the samples as graph nodes, whose arcs are defined by an adjacency relation and weighted by some distance function. Further, a role competition process between some key nodes (prototypes) is carried out in order to partition the graph into optimum-path trees (OPTs) according to some path-cost function. Therefore, to design an Optimum-Path Forest-based classifier, one needs to define: (i) an adjacency relation, (ii) a path-cost function and (iii) a methodology to estimate prototypes.

Suppose we have a fully labeled dataset $Z = Z_1 \cup Z_2$, in which Z_1 and Z_2 stand for training and test sets, respectively. Let $S \subset Z_1$ be a set of prototypes of all classes (i.e., key samples that best represent the classes). Let (Z_1, A) be a complete graph whose nodes are the samples in Z_1 and any pair of samples defines an arc in $A = Z_1 \times Z_1$. Let π_s be a path in the graph that ends in sample $s \in Z_1$, and $\langle \pi_s \cdot (s,t) \rangle$ the concatenation between π_s and the arc (s, t), $t \in Z_1$. In this chapter, we employ a path-cost function that returns the maximum arc-weight along a path in order to avoid chains, and also to show the idea of connectivity between samples. This path-cost function is denoted here as Ψ, and it can be computed as follows:

$$\begin{aligned} \Psi(\langle s \rangle) &= \begin{cases} 0, & \text{if } s \in S, \\ +\infty, & \text{otherwise,} \end{cases} \\ \Psi(\pi_s \cdot \langle s,t \rangle) &= \max\{\Psi(\pi_s), d(s,t)\}, \end{aligned} \tag{6}$$

in which $d(s, t)$ means the distance between nodes s and t. Thus, the objective of the Optimum-Path Forest algorithm (supervised version) is to minimize $\Psi(\pi_t)$, $\forall t \in Z_1$.

An optimal set of prototypes S^* can be found by exploiting the theoretical relation between the minimum-spanning tree and optimum-path tree for Ψ. By computing a minimum-spanning tree in the complete graph (Z_1, A), we obtain a connected acyclic graph whose nodes are all samples of Z_1 and the arcs are undirected and weighted by the distances d between adjacent samples. The spanning tree is optimum in the sense that the sum of its arc weights is the minimum as compared to any other spanning tree in the complete graph. In the minimum-spanning tree, every pair of samples is connected by a single path, which is optimum according to Ψ. Thus, the minimum-spanning tree contains one optimum-path tree for any selected root node. The optimum prototypes are the closest elements of the minimum-spanning tree with different labels in Z_1.

The Optimum-Path Forest training phase consists, essentially, of starting the competition process between prototypes in order to minimize the cost of each training sample. At the final of such procedure, we obtain an optimum-path forest, which is a collection of optimum-path trees rooted at each prototype. A sample connected to an OPT means that it is more strongly connected to the root of that tree than to any other root in the forest.

Furthermore, in the classification phase, for any sample $t \in Z_2$, we consider all arcs connecting t with samples $s \in Z_1$, as though t were part of the training graph. Considering all possible paths from S^* to t, we find the optimum path $P^*(t)$ from S^* and label t with the class $\lambda(R(t))$ of its most strongly connected prototype $R(t) \in S^*$. This path can be identified incrementally, by evaluating the optimum cost $C(t)$ as:

$$C(t) = \min\{\max\{C(s), d(s,t)\}\}, \ \forall s \in Z_1. \tag{7}$$

Let the node $s^* \in Z_1$ be the one that satisfies (Eq. 7) (i.e., the predecessor $P(t)$ in the optimum path $P^*(t)$). Given that $L(s^*) = \lambda(R(t))$, the classification simply assigns $L(s^*)$ as the class of t.

3 Methodology

In this section, we present the methodology used to evaluate the performance of BFPA. Details about the dataset used, experimental setup and the compared techniques are also provided.

3.1 Datasets

Table 1 presents the datasets used in this work.[1] Such datasets differ on the number of samples, features and also classes. Therefore, the idea is to evaluate the proposed approach in different contexts.

The last two datasets, i.e., NTL_c and NTL_i, are related to non-technical losses detection in comercial and industrial profiles, respectively. These are private datasets obtained by a Brazilian electrical power company. Such sort of problem is of great interest to electrical power companies, mainly in Brazil, in which the amount of losses in energy thefts can reach up to 20 % in some regions. Therefore, the characterization of illegal consumers, i.e., to find out the most important features that allow us to identify them, is so important as to effective recognize them.

3.2 Nature-Inspired Metaheuristic Algorithms

In this work, we have also employed three others evolutionary optimization techniques for comparison purposes. A brief detail about each of them is given below.

[1] The first four datasets can be found on http://featureselection.asu.edu/datasets.php.

Table 1 Description of the benchmarking datasets

Dataset	# samples	# features	# classes
GLI-85	85	22,283	2
SMK-CAN-187	187	19,993	2
TOX-171	171	5,748	4
AR10P	130	2,400	10
NTL_c	4,952	8	2
NTL_i	3,182	8	2

Particle Swarm Optimization (PSO): PSO was inspired by the social behavior of bird flocking or fish schooling [8]. The fundamental idea is that each particle represents a potential solution which is updated according to its own experience and from its neighbors' knowledge. The motion of an individual particle for the optimal solution is governed through its position and velocity interactions, and also by its own previous best performance and the best performance of their neighbors.

Firefly Algorithm (FA): FA was also proposed by Yang [24], based on the flashing behaviour and attractiveness of fireflies. The brightness of a firefly at a given position is determined by the value of the objective function in that position. Each firefly is attracted by a brighter firefly through the attraction factor that vary with their distance.

Harmony Search (HS): HS was a meta-heuristic algorithm inspired by the improvisation process of music players [6]. Musicians often improvise in searching for a perfect state of harmony. The main idea is to use the same process adopted by musicians to create new songs to obtain a near-optimal solution according to some fitness function. Each possible solution is modelled as a harmony, and each musical note corresponds to one decision variable.

In this present work, we have used all the binary optimization versions of each aforementioned technique, i.e., Binary PSO (BPSO) [5], Binary Firefly (BFA) [4, 11], as well as Binary HS (BHS) [14].

3.3 Experimental Setup

Firstly, the dataset Z is randomly partitioned in N folds, i.e., $Z = F_1 \cup F_2 \cup \cdots \cup F_N$. For each fold F_i, we train a given instance of the OPF classifier over it, for further evaluation of another fold F_j, $i \neq j$. Therefore, the classification accuracy over F_j is then used as the fitness function to guide the optimization algorithms for selecting the most representative set of features. Each agent of the population (pollen, particle, firefly, harmony) in these meta-heuristic algorithms is associated with a string of bits denoting the presence or absence of a feature. Thus, for each agent, we construct a classifier from the training set F_i only with the selected features, say F_i^*, and assigns the accuracy over F_j as the fitness function. As long as the procedure converges, i.e., all generations of a population were computed, the agent with the highest fitness value encodes a solution with the best compacted set of features.

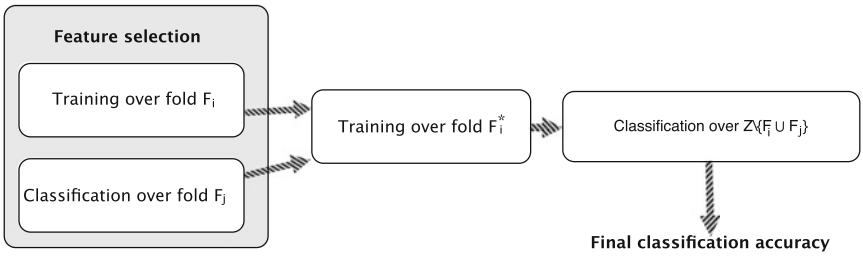


Fig. 1 Proposed methodology to evaluate the compared techniques

After that, we build a classification model using the training set with the selected features (F_i^*), and we also evaluate the quality of the solution through a classification process over the test set, which is built over the remaining folds in $Z \backslash \{F_i \cup F_j\}$. This procedure is conducted for each fold F_i in the dataset to be part of the training set, and thus we have $N(N-1)$ combinations in the final of the process, which will be averaged for comparison purposes. Figure 1 illustrates the methodology described above.

In regard to the recognition rate, we used an accuracy measure proposed by Papa et al. [12]. If there are two classes, for example, with very different sizes and a classifier always assigns the label of the largest class, its accuracy will fall drastically due to the high error rate on the smallest class. The accuracy is measured by taking into account that the classes may have different sizes in a testing set F_j. Let us define:

$$e_{i,1} = \frac{FP_i}{|F_j| - |F_j^i|}, \tag{8}$$

and

$$e_{i,2} = \frac{FN_i}{|F_j^i|}, \quad i = 1, 2, \ldots, C, \tag{9}$$

where C stands for the number of classes, $|F_j^i|$ concerns with the number of samples in F_j that come from class i, and FP_i and FN_i stand for the false positives and false negatives for class i, respectively. That is, FP_i is the number of samples from other classes that were classified as being from the class i in F_j, and FN_i is the number of samples from the class i that were incorrectly classified as being from other classes in F_j. The error terms $e_{i,1}$ and $e_{i,2}$ are then used to define the total error from class i:

$$E_i = e_{i,1} + e_{i,2}. \tag{10}$$

Finally, the accuracy Acc is then defined as follows:

$$Acc = 1 - \frac{\sum_{i=1}^{C} E_i}{2C}. \quad (11)$$

4 Experimental Results

In this section, we summarize and discuss the experimental results regarding the proposed approach for feature selection. The results presented in this section stand for the mean accuracy and standard deviation over 25 independent runs using the methodology presented in Sect. 3.3. Since the evolutionary optimization algorithms are non-deterministic, such approaches seem to be robust to avoid biased results. The optimization algorithms (BFPA, BPSO, BHS and BFA) were implemented in C language following the guidelines provided by their references. The experiments were executed on a computer with a Pentium Intel Core *i5*® 3.20 Ghz processor, 4 GB of RAM and Linux Ubuntu Desktop LTS 10.04 as the operational system.

Table 2 presents the parameters used for each optimization technique employed in this work. The c_1 and c_2 parameters of PSO control the pace during the particles' movement, and the "Harmony Memory Considering Rate" (HMCR) of BHS stands for the amount of information that will be used from the artist's memory (songs that have been already composed) in order to compose a new harmony. In regard to BFA, α and β_0 are related to the step size of a firefly, and γ stands for the light absorption coefficient. In addition, we have used a population of 30 agents and 100 iterations for all techniques, with such values being an empirical set.

Figure 2 displays the mean accuracy results using the proposed methodology (Sect. 3.3). It can be observed that the feature selection techniques can slightly improve the results obtained using the original datasets, i.e., without feature selection. The second point is that all techniques achieved quite similar results. Therefore, the results showed BFPA is suitable for feature selection tasks. We have also performed the statistical Wilcoxon Signed-Rank Test [22] to verify whether there is a significant difference between BFPA and the other techniques used in this work (considering the OPF recognition rate). Table 3 displays the p-values, being the bold ones the situations in which BFPA and the respective technique have obtained different performances, i.e., when the p-values are lower than a significance level of $\alpha = 0.05$.

Table 2 Parameters used for each meta-heuristic optimization technique. Notice the inertia weight w for PSO was linearly decreased from 0.9 to 0.4 during the convergence process

Technique	Parameters
BPSO	$c_1 = c_2 = 2$
BFA	$\gamma = 0.8$, $\beta_0 = 1.0$, $\alpha = 0.01$
BHS	HMCR = 0.9
BFPA	$a = 1.0$, $p = 0.8$

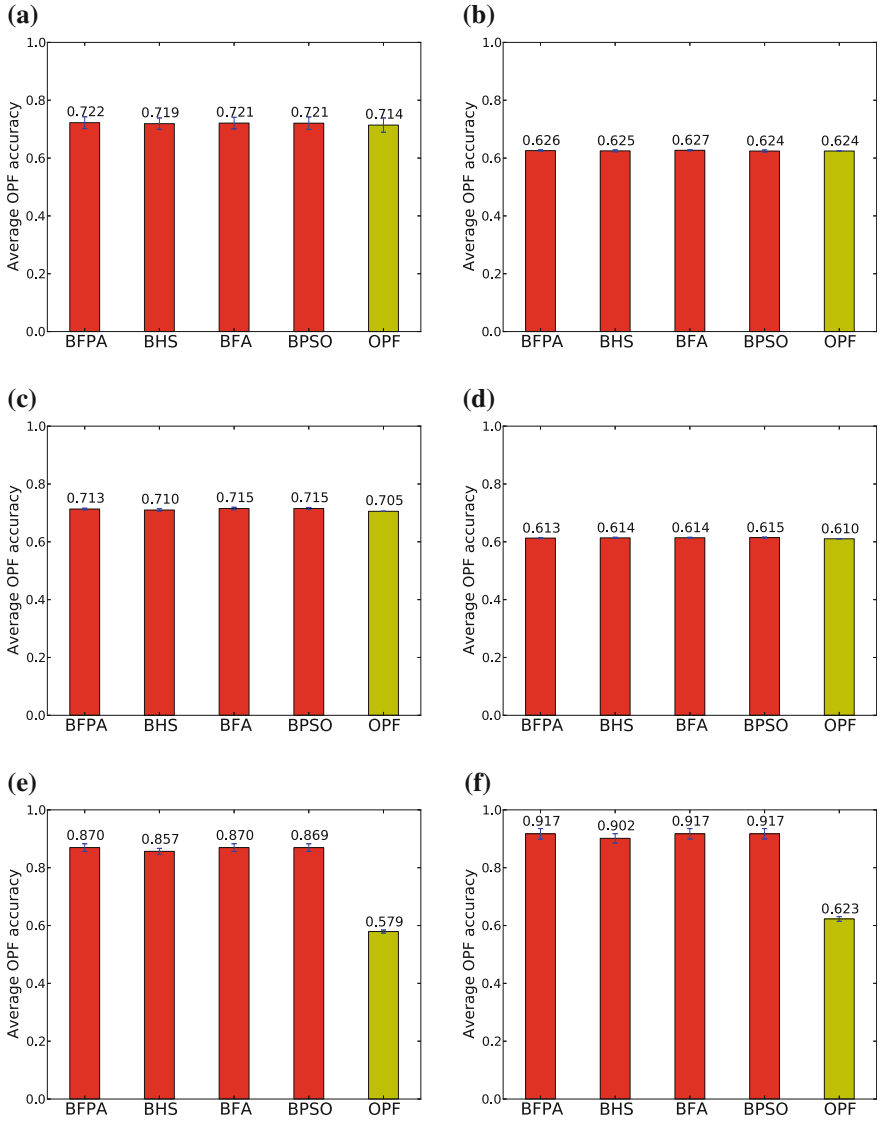


Fig. 2 Average OPF accuracy over **a** GLI-85, **b** SMK-CAN-187, **c** TOX-171, **d** AR10P, **e** NTL_c and **f** NTL_i datasets

It can also be seen that there is a statistical difference between BFPA and BPSO, and BFPA and BFA for AR10P dataset, and also a difference between BFPA and BHS considering TOX-171 dataset. Figure 3 displays the convergence rates of all techniques considering the datasets employed in this work. Such recognition rates are the ones obtained over the validating set F_j, as depicted in Fig. 1.

Table 3 Wilcoxon Signed-Rank Test evaluation: *p*-values computed between BFPA, BPSO, BFA and BHS

Dataset	BPSO	BFA	BHS
GLI-85	0.3130	0.5629	0.1425
SMK-CAN-187	0.1829	0.2012	0.4432
TOX-171	0.1742	0.1500	**0.0112**
AR10P	**0.0023**	**0.0197**	0.0573
NTL_c	0.3281	0.4769	1.2290
NTL_i	0.1957	0.3318	1.2290

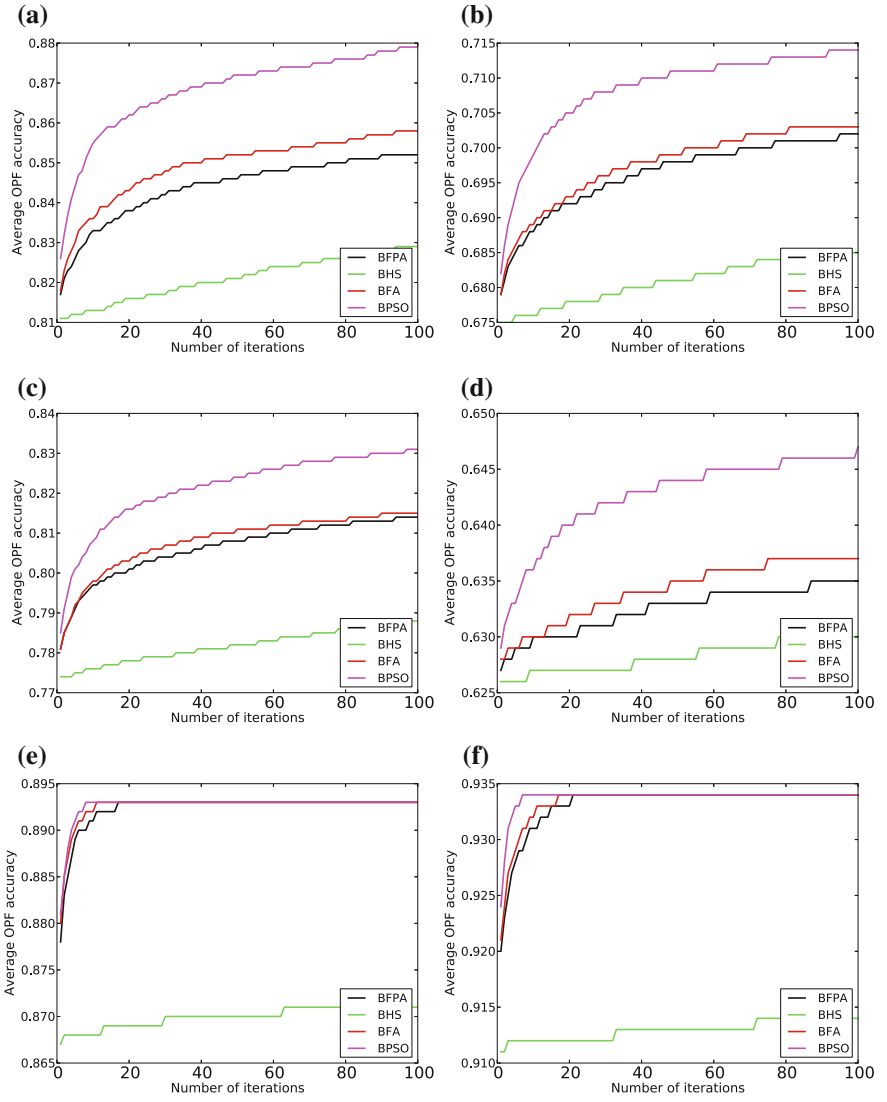


Fig. 3 Average convergence rate considering the OPF accuracy over the feature selection step for **a** GLI-85, **b** SMK-CAN-187, **c** TOX-171, **d** AR10P, **e** NTL_c and **f** NTL_i datasets

From Fig. 3, it is possible to observe that BPSO has been the technique with the fastest convergence rate, followed by BFA and BFPA. However, the good BPSO performance over the feature selection process does not seem to enhance a lot its final accuracy over the test set, as displayed in Fig. 2. In addition, BHS has the slowest convergence process, since it updates only one agent (harmony) per iteration, which turns it fast considering the execution time, but it tends to be slower for

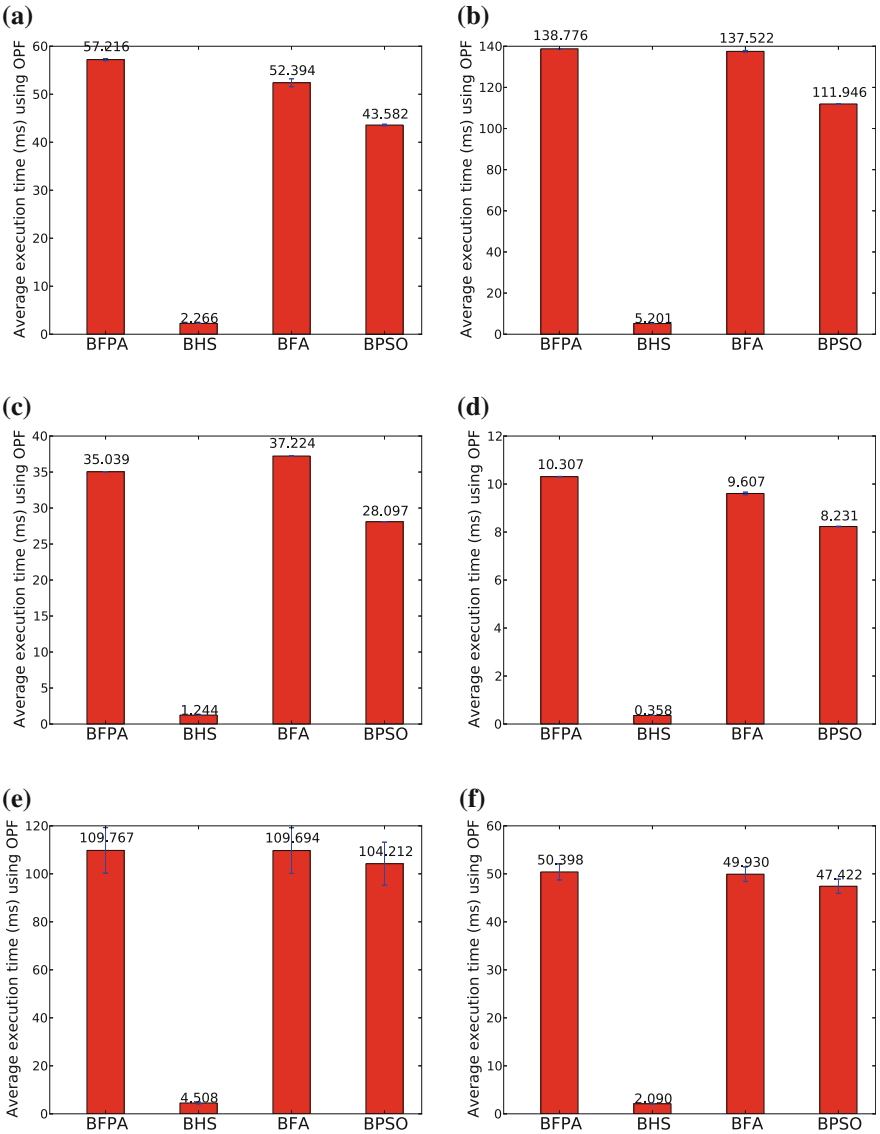


Fig. 4 Average execution time (ms) for feature selection considering **a** GLI-85, **b** SMK-CAN-187, **c** TOX-171, **d** AR10P, **e** NTL_c and **f** NTL_i datasets

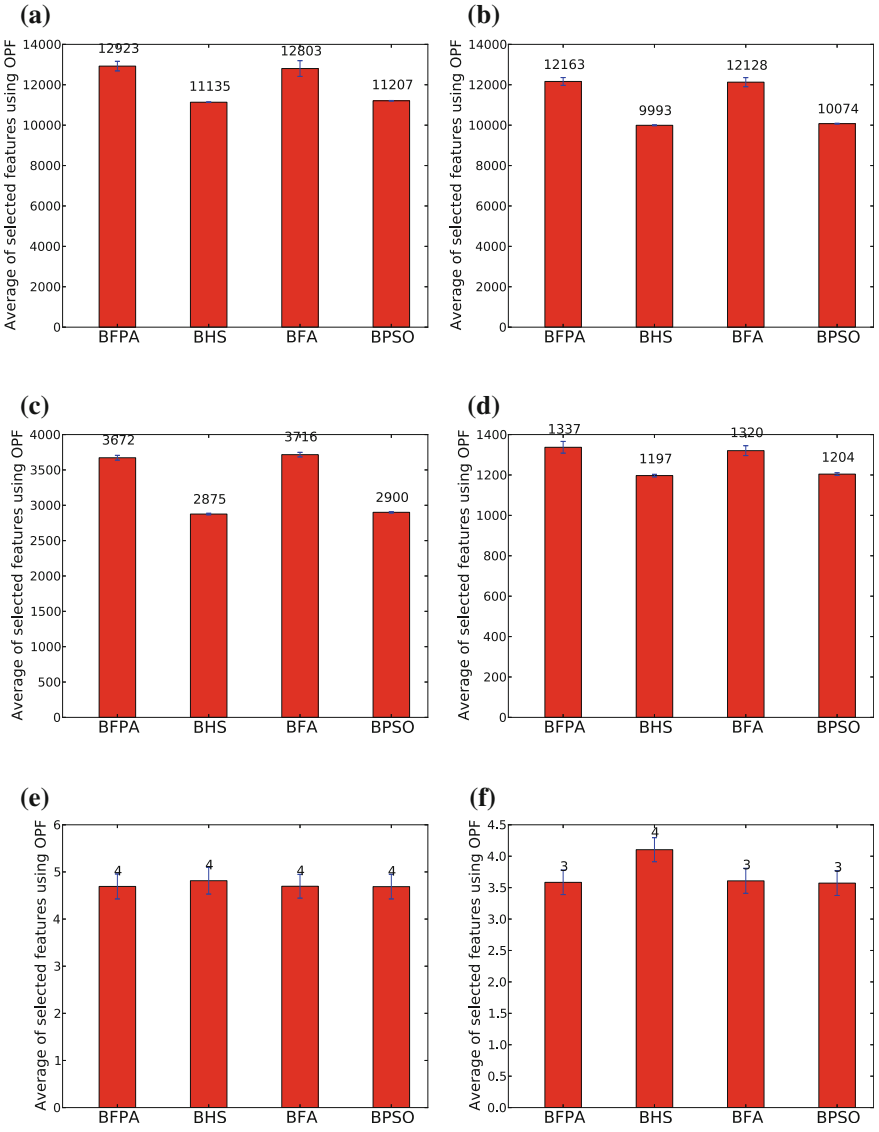


Fig. 5 Mean number of selected features considering **a** GLI-85, **b** SMK-CAN-187, **c** TOX-171, **d** AR10P, **e** NTL_c and **f** NTL_i datasets. The numbers have been truncated for sake of presentation

convergence. Figure 4 displays the execution time for all considered optimization techniques. Though it is possible to observe that BFPA has been one of the slowest techniques, since it is the only one that employs Lévy flights to move pollens across the search space, which can increase the computational burden slightly. It is also observed that BFPA in general produces better results in terms of accuracy, as seen in Fig. 2.

In addition, Fig. 5 displays the mean number of selected features for each dataset. It is possible to observe BHS has selected the fewest number of features, followed by BPSO. However, as we have high dimensional datasets (in case of GLI-85, SMK-CAN-187, TOX-171 and AR10P), the absolute number do not differ a lot from all techniques. It seems that all techniques have obtained similar results considering the recognition rate, except for the computational load and convergence speed.

5 Conclusions

In this work, we have solve the problem of feature selection by considering feature selection as an evolutionary-based optimization task, constrained on a boolean lattice. The idea is to represent each possible solution as a string of bits, in which each of them denotes whether or not a feature will be used to compose the final set.

We have evaluated a recent nature-inspired approach, namely the Flower Pollination Algorithm, to tackle this task on six datasets. The proposed approach has been compared with other methods such as Particle Swarm Optimization, Harmony Search and Firefly Algorithm. The experimental results have been analyzed in terms of the recognition rates, convergence speed, number of selected features and computational loads. All techniques have obtained similar recognition rates, and it seems that PSO has the fastest convergence process, while HS has lowest computational cost. Therefore, we have showed that FPA is also suitable for feature selection tasks, since its results are comparable to the ones obtained by some state-of-the-art evolutionary techniques. Future research can focus on the parametric studies of the FPA as well as its extension and hybridization with other techniques.

References

1. Bababdani, B.M., Mousavi, M.: Gravitational search algorithm: a new feature selection method for {QSAR} study of anticancer potency of imidazo[4,5-b]pyridine derivatives. Chemometr. Intell. Lab. Syst. **122**(15), 1–11 (2013)
2. Chen, B., Chen, L., Chen, Y.: Efficient ant colony optimization for image feature selection. Signal Process. **93**(6), 1566–1576 (2013). (special issue on Machine Learning in Intelligent Image Processing)
3. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern classification, 2nd edn. Wiley, New York (2001)
4. Falcon, R., Almeida, M., Nayak, A.: Fault identification with binary adaptive fireflies in parallel and distributed systems. In: IEEE Congress on Evolutionary Computation (CEC), pp. 1359–1366 (2011)
5. Firpi, HA., Goodman, E.: Swarmed feature selection. In: Proceedings of the 33rd Applied Imagery Pattern Recognition Workshop. IEEE Computer Society, Washington, DC, pp. 112–118 (2004)
6. Geem, Z.W.: Music-inspired harmony search algorithm: theory and applications, 1st edn. Springer, Berlin (2009)

7. Kabir, M., Shahjahan, M., Murase, K.: An efficient feature selection using ant colony optimization algorithm. In: Leung, C., Lee, M., Chan, J. (eds.) Neural Information Processing, Lecture Notes in Computer Science, vol. 5864, pp. 242–252, Springer, Berlin (2009)
8. Kennedy, J., Eberhart, R.C.: Swarm intelligence. Morgan Kaufman, Burlington (2001)
9. Marinakis, Y., Marinaki, M., Matsatsinis, N.: A hybrid discrete artificial bee colony—GRASP algorithm for clustering. In: Proceedings of the International Conference on Computers Industrial Engineering, pp. 548–553 (2009)
10. Oh, I.S., Lee, J.S., Moon, B.R.: Hybrid genetic algorithms for feature selection. IEEE Trans. Pattern Anal. Mach. Intell. **26**(11), 1424–1437 (2004)
11. Palit, S., Sinha, S.N., Molla, M.A., Khanra, A., Kule, M.: A cryptanalytic attack on the knapsack cryptosystem using binary firefly algorithm. In: 2nd International Conference on Computer and Communication Technology (ICCCT), pp. 428–432 (2011)
12. Papa, J.P., Falcão, A.X., Suzuki, C.T.N.: Supervised pattern classification based on optimum-path forest. Int. J. Imaging Syst. Technol. **19**(2), 120–131 (2009)
13. Papa, J.P., Falcão, A.X., Albuquerque, V.H.C., Tavares, J.M.R.S.: Efficient supervised optimum-path forest classification for large datasets. Pattern Recogn. **45**(1), 512–520 (2012)
14. Ramos, C.C.O., Souza, A.N., Chiachia, G., Falcão, A.X., Papa, J.P.: A novel algorithm for feature selection using harmony search and its application for non-technical losses detection. Comput. Electr. Eng. **37**(6), 886–894 (2011)
15. Ramos, C.C.O., De Souza, A., Falcão, A., Papa, J.: New insights on nontechnical losses characterization through evolutionary-based feature selection. Power Deliv. IEEE Trans. **27** (1), 140–146 (2012)
16. Ramos, C.C.O., de Souza, A.N., Falcão, A.X., Papa, J.P.: New insights on non-technical losses characterization through evolutionary-based feature selection. IEEE Trans. Power Deliv. **27**(1), 140–146 (2012)
17. Rodrigues, D., Pereira, L.A.M., Almeida, T.N.S., Papa, J.P., Souza, A.N., Ramos, C.C.O., Yang, X.S.: A binary cuckoo search algorithm for feature selection. In: Proceedings of IEEE International Symposium on Circuits and Systems, pp. 465–468 (2013a)
18. Rodrigues, D., Pereira, L.A.M., Nakamura, R.Y.M., Costa, K.A.P., Yang, X.S., Souza, A.N., Papa, J.P.: A wrapper approach for feature selection based on bat algorithm and optimum-path forest. Expert Syst. Appl. **41**(5), 2250–2258 (2013)
19. Rodrigues, D., Pereira, L.A.M., Papa, J.P., Ramos, C.C.O., Souza, A.N., Papa, L.P.: Optimizing feature selection through binary charged system search. In: Proceedings of 15th International Conference on Computer Analysis of Images and Patterns, pp. 377–384 (2013c)
20. Schiezaro, M., Pedrini, H.: Data feature selection based on artificial bee colony algorithm. EURASIP J. Image Video Process. **1**, 1–8 (2013)
21. Sivagaminathan, R.K., Ramakrishnan, S.: A hybrid approach for feature subset selection using neural networks and ant colony optimization. Expert Syst. Appl. **33**(1), 49–60 (2007)
22. Wilcoxon, F.: Individual comparisons by ranking methods. Biometrics Bull. **1**(6), 80–83 (1945)
23. Yang, J., Honavar, V.: Feature subset selection using a genetic algorithm. IEEE Intell. Syst. Appl. **13**(2), 44–49 (1998)
24. Yang, X.S.: Firefly algorithm, stochastic test functions and design optimisation. Int. J. Bio-Inspired Comput. **2**(2), 78–84 (2010)
25. Yang, X.S.: Flower pollination algorithm for global optimization. In: Proceedings of the 11th International Conference on Unconventional Computation and Natural Computation, pp. 240–249. Springer, Berlin (2012)
26. Yang, X.S., Karamanoglu, M., He, X.: Flower pollination algorithm: a novel approach for multiobjective optimization. Eng. Optim. **46**(9), 1222–1237 (2014)

Bat Algorithm Application for the Single Row Facility Layout Problem

Sinem Büyüksaatçı

Abstract Facility layout, which involves planning, designing and optimization of physical arrangement of resources, can be defined as one of the most fundamental operations in manufacturing systems. A good placement of facilities contributes to the overall efficiency of operations and reduces total operating expenses. Because of its importance, the facility layout problem has attracted attention in the research community. However, layout problems are known as complex and solvable in exponential time. Due to the combinatorial nature of such problems, no efficient exact algorithms exist. Thus, during the last decades, several metaheuristics have been applied to obtain efficient solutions. One of the special classes of the facility layout problem is the Single Row Facility Layout Problem (SRFLP), which is expressed by finding an optimal linear placement of facilities with varying dimensions on a straight line. In this study, bat algorithm is used to solve single row layout problem. Before application, the optimal settings of the bat algorithm parameters are determined through experimental analysis. Then the performance of the bat algorithm is tested on six-problem set selected from the literature.

Keywords Bat algorithm · Single row facility layout problem · Design of experiments · Experimental analysis

1 Introduction

The layout arrangement is an inevitable problem in all-industrial plants and the decisions regarding the layout of machines receive intensive attention in production and operations management. The physical layout design is extremely important because it affects the required initial investments, the amount of in-progress inventory, the production lead times, the production rate and the material handling cost.

S. Büyüksaatçı (✉)
Faculty of Engineering, Department of Industrial Engineering, Istanbul University, Avcılar, İstanbul, Turkey
e-mail: sinemb@istanbul.edu.tr

X.-S. Yang (ed.), *Recent Advances in Swarm Intelligence and Evolutionary Computation*, Studies in Computational Intelligence 585, DOI 10.1007/978-3-319-13826-8_6

As stated by Tompkins et al. [47], estimates show that 20–50 % of the total operating expenses of manufacturing systems are spent on material handling cost and a suitable layout design can decrease the operating costs by 10–30 %.

Facility layout can be defined as a process in which the planning for the placement of different types of facilities such as machines, employee workstations, utilities, customer service areas, restrooms, material storage areas, lunchrooms, drinking fountains, offices, internal walls etc. according to a determined objective function.

There are several alternative facility layout types that are referred in literature. According to the production variety and volume, the facility shapes, the material handling system chosen, the number of floors on which the machines can be assigned, the different possible flows allowed for parts and the pick-up/drop-off locations, layout problems are differentiated [15].

When dealing with a material handling system, the problem consists in arranging facilities along the material-handling path. Heragu and Kusiak in 1988 [19] indicated that the type of material-handling device determines the pattern to be used for the layout of machine. The most commonly used material handling devices (MHDs) are conveyors (belt, roller, wheel), automated guided vehicles (AGV), handling robot, gantry robot, etc. The most frequently encountered layout types connected to the MHDs are:

- linear single row machine layout
- linear double row machine layout
- linear multi row machine layout
- semi-circular (u-shaped) machine layout
- loop layout
- serpentine (or open field) machine layout

Although modern material handling systems often allow for complex flow path configurations, the single row facility layout is the most widely implemented layout pattern in the configuration of manufacturing systems. In this configuration, machines are arranged along a straight line where a material-handling device moves the items from one machine to another.

The single row facility layout problem (SRFLP), which is also called “the one-dimensional machine location problem” or “the linear machine cell location problem”, connected with some combinatorial optimization problems. One of those that has been extensively studied in literature is quadratic assignment problem (QAP) formulated by Koopmans and Beckman in 1957 [24]. In this problem type, three $n \times n$ input matrices, which are flow matrix, distance matrix and cost of placing matrix, are given. The goal of the problem is one-to-one assignment of n facilities to n locations while minimizing the sum of the distances multiplied by the flow between locations and the cost for placing a facility at a certain site. Linear ordering (arrangement) problem also is a special case of SRFLP and QAP. If there exists no interaction between facilities such that one is concerned only with locating new facilities relative to existing facilities, the problem is linear arrangement problem [47]. This problem type also differs from QAP by its efficient, polynomial-time solution methods as mentioned by Burkard [12].

SRFLP has a number of practical applications in literature, including the arrangement of departments on one side of a corridor in supermarkets, hospitals and office buildings [42], the arrangement of books on a shelf in a library and the assignment of files to disk cylinders in computer storage [37], the layout of machines in flexible manufacturing systems, where machines within manufacturing cells are often placed along a straight path travelled by an automated guided vehicle [19] and the assignment of airplanes to gates in an airport terminal [9].

2 Formulations of Single Row Facility Layout Problem

A variety of formulations exist for the SRFLP in literature. In order to model the SRFLP, the following main notations are required:

- c_{ij}: Cost of transporting a unit of material for unit distance between facilities i and j.
- f_{ij}: Material flow between facilities i and j.
- d_{ij}: Distance between facilities i and j.
- l_i: Length of facility i.

The term c_{ij} has been used in different meanings in literature. It has been defined as cost by Simmons in 1969 [42], frequency of travel by Heragu and Kusiak in 1988 [19], affinity by Romero and Sánchez-Flores in 1990 [39] and transition probabilities by Picard and Queyranne in 1981 [37].

The first formulation, which is given in Eq. (1), was proposed by Simmons [42].

$$\text{Minimize} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} c_{ij} s_{ij} \tag{1}$$

where s_{ij} is the sum of the half-lengths of facility i and j added to the lengths of all facilities between them considering the permutations.

Love and Wong in 1976 [32] presented a binary mixed integer programming formulation for the SRFLP. The formulation is as follows.

$$\begin{aligned} \text{Minimize} \quad & \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} w_{ij}\left(R_{ij} + L_{ij}\right) \\ \text{subject to} \quad & \\ & R_{ij} - L_{ij} = x_i - x_j + \tfrac{1}{2}(l_j - l_i) \\ & x_i - x_j + M\left(\alpha_{ij}\right) \geq l_i \\ & x_j - x_i + M\left(1 - \alpha_{ij}\right) \geq l_j \\ & l_j \leq x_i \leq \sum_{1}^{n} l_i \\ & \alpha_{ij} = 0, 1 \\ & R_{ij}, L_{ij}, x_1, \ldots, x_n \geq 0 \end{aligned} \tag{2}$$

where

x_i Endpoint of facility i farthest from the origin,
α_{ij} 1 if facility i is to the left of facility j, 0 otherwise,
R_{ij} Distance between centroid of facility i and centroid of facility j if facility i is to the right of facility j, 0 otherwise,
L_{ij} Distance between centroid of facility i and centroid of facility j if facility i is to the left of facility j, 0 otherwise;
M An arbitrarily large number

Kumar et al. [28] provided a quadratic assignment problem formulation for the case where a number of distinct facilities are to be assigned to an equal number of equidistant locations which lie on a straight line in 1995. The mathematical formulation is given in Eq. (3).

$$\text{Minimize} \quad \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{k'=1}^{n} x_{ik}x_{jk'}f_{ij}|k-k'|$$

subject to

$$\begin{aligned} &\sum_{i=1}^{n} x_{ik} = 1, \quad k = 1,2,\ldots,n \\ &\sum_{k=1}^{n} x_{ik} = 1, \quad i = 1,2,\ldots,n \\ &x_{ij} \in \{0,1\} \quad i,k = 1,2,\ldots,n, \end{aligned} \tag{3}$$

where $x_{ik} = 1$ if facility i is assigned to location k; 0 otherwise.

3 Solution Methods for SRFLP

In the context of combinatorial optimization (CO), algorithms can be classified as either exact (complete) or approximate algorithms as mentioned by Talbi [45]. Exact algorithms are guaranteed to find an optimal solution for every finite size instance of a CO problem in bounded time. Yet, for CO problems that are NP-hard, complete methods need exponential computation time. Besides this, in approximate methods such as heuristics, it is important to get good solutions in a significantly reduced amount of time. But they don't guarantee to find optimal solutions.

In the class of exact methods the following classical algorithms may be found: dynamic programming, branch and X family of algorithms (branch and bound, branch and cut, branch and price), constraint programming etc. These methods can be applied to small instances of difficult problems. The size of the instance is not only a unique indicator that describes the difficulty of a problem. For a given problem, an exact algorithm cannot solve some small instances while some large instances may be solved exactly by the same algorithm.

In the class of approximate methods, two subclasses of algorithms may be distinguished: approximation algorithms and heuristic algorithms. Unlike heuristics,

which usually find reasonably "good" solutions in a reasonable time, approximation algorithms provide provable solution quality and provable run-time bounds.

Heuristics find "good" solutions on large-size problem instances. They allow obtain acceptable performance at acceptable costs in a wide range of problems. In general, heuristics do not have an approximation guarantee on the obtained solutions.

In the 1970s, a new kind of approximate algorithm had emerged which basically tries to combine basic heuristic methods in higher level frameworks aimed at efficiently and effectively exploring a search space. These methods are nowadays commonly called metaheuristics.

The word *heuristic* has its origin in the old Greek word *heuriskein*, which means "to find" while the suffix *meta*, also a Greek word, means "beyond, in an upper level". Dr. Fred Glover first introduced the term *metaheuristic* in 1986 [17], which is a master strategy that guides and modifies other heuristics to produce solutions beyond those that are normally generated in a quest for local optimality [2, 10, 45].

This section presents the exact and approximate algorithms that have been used to formulate and solve single row facility layout problem in literature. All these relevant algorithms are listed in Table 1 and a detailed explanation is given.

3.1 Exact Solution Methods

Several exact methods have also been suggested to solve the SRFLP in literature. Simmons first studied and suggested a branch-and-bound algorithm for SRFLP in 1969 [42]. The feasible layout of the facilities formed step by step in this algorithm i.e., the permutation of facilities was not built up until the late stages of the algorithm. For this reason the technique is also known as the branch and build up approach. SRFLP instances of size up to 11 were solved using branch-and-bound algorithm.

Afterwards Simmons [43] pointed out the dynamic programming would give better results in solving one dimensional assignment problem. Picard and Queyranne [37] first implemented the dynamic programming successfully to the SRFLP instances with up to 15 facilities. However the high memory requirement of this technique has rendered it unattractive for the research community. Another dynamic programming algorithm was presented by Kouvelis and Chiang in 1996 [27]. They studied the row layout problem under the design objective of minimizing the total backtracking distance of the material handling device and their algorithm was able to solve SRFLP instances with 20 facilities.

Adolphson and Hu [1] suggested an algorithm for the optimal linear ordering problem to extract the linear facility sequence for problems whose flow matrix can be represented as a single rooted tree.

A nonlinear programming model, which is called ABSMODEL was presented by Heragu and Kusiak in 1991 [20]. Love and Wong [32] formulated the SRFLP as a linear mixed-integer program using distance variables and solved it using the IBM

Table 1 Solution methods for the single row facility layout problem

			EXACT METHODS								HEURISTICS and METAHEURISTICS							
			Branch and bound	Dynamic programming	Graph theory	Nonlinear programming	Linear mixed-integer programming	Cutting plane approach	Semi-definite programming	Branch and cut	Eigenvector approach	Constructive heuristic algorithm	Simulated annealing	Tabu search	Genetic algorithm	Ant colony optimization	Particle swarm optimization	Others
Years	**Authors**	**References**																
1969	Simmons	[42]	✓															
1970	Hall	[18]									✓							
1971	Simmons	[43]		✓														
1973	Adolphson and Hu	[1]			✓													
1974	Neghabat	[34]										✓						
1976	Love and Wong	[32]					✓											
1981	Picard and Queyranne	[37]		✓														
1988	Heragu and Kusiak	[19]										✓						
1990	Romero and Sanchez-Flores	[39]											✓					
1991	Heragu and Kusiak	[20]				✓												
1992	Heragu	[21]											✓	✓				✓
1992	Heragu and Alfa	[22]											✓					
1992	Kouvelis and Chiang	[26]											✓					
1995	Kumar et al.	[28]										✓						
1996	Kouvelis and Chiang	[27]		✓														
1997	Braglia	[11]																✓
2000	Alvarenga et al.	[3]											✓	✓				
2001	Djellab and Gourgand	[14]																✓
2001	Ponnambalam and Ramkumar	[38]													✓			✓
2005	Anjos et al.	[7]							✓									
2005	Solimanpur et al.	[44]														✓		

(continued)

Table 1 (continued)

			EXACT METHODS								HEURISTICS and METAHEURISTICS							
			Branch and bound	Dynamic programming	Graph theory	Nonlinear programming	Linear mixed-integer programming	Cutting plane approach	Semi-definite programming	Branch and cut	Eigenvector approach	Constructive heuristic algorithm	Simulated annealing	Tabu search	Genetic algorithm	Ant colony optimization	Particle swarm optimization	Others
Years	**Authors**	**References**																
2006	Amaral	[4]					✓											
2008	Teo and Ponnambalam	[46]														✓	✓	
2008	Anjos and Vanelli	[8]						✓	✓									
2008	Amaral	[5]					✓											
2009	Amaral	[6]						✓										
2009	Anjos and Yen	[9]							✓									
2009	Lin	[31]													✓			
2010	Samarghandi et al.	[41]															✓	
2010	Samarghandi and Eshghi	[40]												✓				
2011	Kumar et al.	[29]																✓
2011	Hungerländer and Rendl	[23]							✓									
2011	Datta et al.	[13]													✓			
2011	Letchford and Amaral	[30]								✓								
2012	Ozcelik	[36]													✓			
2014	Kothari and Ghosh	[25]													✓			

Mixed Integer Programming (MIP) code. Amaral in 2006 [4] proposed a new mixed-integer linear programming model for the SRFLP. The new model presented the same number of zero-one variables and a smaller number of continuous variables than the model presented by Love and Wong in 1976 [32]. In addition, the formulation of the new model contained facet-defining inequalities, thus it was stronger than Love and Wong's model from a theoretical viewpoint. Amaral [5] achieved a more efficient linear mixed integer program by linearizing a quadratic model based on ordering variables in 2008. This code could solve SRFLP instances with sizes up to 18 that were proposed in Simmons [42], Love and Wong [32], Heragu and Kusiak [20] within reasonable time.

Amaral [6] suggested a linear programming based cutting plane approach to solve SRFLP. The algorithm started by optimizing a linear program over the partial description given and used some valid inequalities introduced as cutting planes. Several instances from literature as well as new large instances with size $n = 33$ and $n = 35$ are tested in the work. The computational results demonstrated that for all the instances tested ($n \leq 35$) the proposed lower bound was equal to the cost of an optimal layout and could be computed relatively fast.

Anjos et al. [7] have constructed a semi-definite programming (SDP) relaxation in 2005, which provides a lower bound on the optimal value of SRFLP. Semi-definite programming refers to the class of optimization problems where a linear function of a matrix variable X is optimized subject to linear constraints on the elements of X and an additional constraint that X be positive semi definite. This includes linear programming problems as a special case, namely when all the matrices involved are diagonal. Anjos and Vanelli in 2008 [8] demonstrated that the combination of a semi-definite programming relaxation with cutting planes is able to compute globally optimal layouts for large SRFLPs with up to 30 departments. Their computational results suggested that their approach could routinely obtain optimal layouts in a few hours for SRFLPs with up to 25 facilities and in several dozen hours for SRFLPs with up to 30 facilities. Anjos and Yen [9] extended the work on the application of semi-definite programming presented by Anjos et al. [7]. The original SDP relaxation had $O(n^3)$ linear constraints. They proposed a new matrix-based formulation that yields a new semi-definite programming relaxation with $O(n^2)$ linear constraints and used it to obtain nearly-optimal solutions for instances with up to 100 facilities with optimality gaps of consistently 5 % or less. Hungerländer and Rendl [23] presented a systematic investigation of semi-definite optimization based relaxations for the quadratic ordering problem in 2011, extending and improving Anjos et al. [7] and Anjos and Yen [9] approaches. The method by Hungerländer and Rendl built on the subgradient optimization technique and dealt with triangle inequality cuts and the LS cuts through Langrangian duality. They successfully applied SDP to the even larger SRFLP instances up to 100 departments.

Codes based on the work by Amaral [4, 5] suffered from weak lower bounds and had high computation times and memory requirements. Letchford and Amaral [30] achieved significant progress in that direction through the first polyhedral study of the distance polytope for SRFLP and proposed a branch and cut algorithm in 2011.

A branch-and-cut algorithm uses a combination of a branching rules and cutting planes to reduce the feasible region of a given mathematical programming problem. They avoided the use of additional variables and used a specialised branching rule to achieve feasibility. The branch-and-cut algorithm was tested on several instances from Simmons [42], Heragu and Kusiak [19], Anjos and Vannelli [8], Anjos and Yen [9]. The results showed that the branch-and-cut algorithm is capable of solving all of the instances and their cutting planes yield excellent lower and upper bounds very quickly for instances with $n \leq 30$, but computing times can be quite long for larger instances.

3.2 Heuristics and Metaheuristics in SRFLP

Due to the combinatorial nature of the facility layout problem, several heuristic and metaheuristic procedures have been developed to obtain good solutions rather than optimal ones. Researchers used heuristics and metaheuristics to solve larger sized SRFLP instances.

Hall in 1970 [18] described efficient eigenvector approaches for solving quadratic placement problems. Neghabat [34] proposed a constructive heuristic algorithm that is capable of handling nonuniform modules and accommodates efficiently restrictions such as the lower bounds on the relative positions between facilities. The algorithm compared with three heuristic procedures that are Hillier, modified Hillier and Computerized Relative Allocation of Facilities Technique (CRAFT). Heragu and Kusiak [19] illustrated the basic types of flexible manufacturing system layouts and presented two new construction algorithms for solving the problem in 1988. The algorithms generated solutions with acceptable quality in low computational time.

Romero and Sanchez-Flores [39] first applied simulated annealing to the SRFLP. Heragu [21] surveyed the different models of the layout problem in 1992 and compared seven heuristic algorithms for solving the models, which are 2-way exchange algorithm, 3-way exchange algorithm, CRAFT, Modified Penalty (MP) algorithm, Simulated Annealing (SA) algorithm, Tabu Search (TS) algorithm and Hybrid Simulated Annealing (HSA) algorithm. Heragu and Alfa [22] tested a hybrid simulated annealing algorithm on the single row layout problems with facilities of unequal area and the multi row layout problems with facilities of equal area. Kouvelis and Chiang [26] utilised a simulated annealing procedure to determine a flow line (or single-row layout) under the assumptions that the number of machines is fixed and backtrack movements were allowed.

Kumar et al. [28] described a constructive heuristic that provided solutions to the SRFLP to minimize the material handling cost in 1995. The heuristic differed from earlier algorithms since at any stage of the process more than two facilities could be added to the solution sequence. Braglia [11] solved the SRFLP with a modified heuristic derived from a heuristic that was developed for the flow shop-scheduling problem. Alvarenga et al. [3] compared performance of TS and SA in case of

single-row and multi-row facility layout problem. They reported that the processing time of TS is better than SA though both of them have same quality solution. Djellab and Gourgand in 2001 [14] presented a heuristic procedure for the single row machine layout problem. They first constructed a best insertion (BI) heuristic to determine an initial permutation of machines, which exploited the special structure of the problem. Combining this heuristic by exploiting the current permutation, they developed an iterative best insertion (IBI) procedure to reduce the space of feasible solutions until only one solution was feasible.

Ponnambalam and Ramkumar [38] developed two heuristic search algorithms in 2001, one combining flow line analysis-5 and Genetic algorithm (GA) and the other combining flow line analysis-6 with GA for the design of single row layout problem. They used the flow-line analysis methods to obtain the initial solution and a genetic algorithm was used to improve the solution. Solimanpur et al. [44] have formulated a 0–1 non-linear mathematical model for the single row layout problem and employed an ant algorithm to solve the model. Teo and Ponnambalam [46] proposed a hybrid ant colony optimization (ACO)/particle swarm optimization (PSO) heuristic to solve single-row layout problems in 2008. In the proposed algorithm, ACO was used as constructive heuristic with a new pheromone update developed to achieve better performance and PSO was used as an improvement heuristic to guide the ants to reach the best solution.

Lin [31] addressed the topic of minimizing the moving distance among cutting pieces during apparel manufacturing and described a hierarchical order-based genetic algorithm to quickly identify an optimal layout that effectively shortens the distance among. Samarghandi et al. [41] used a PSO algorithm in 2010 to solve SRFLP in which the sizes of facilities were assumed to be different. Performance of the proposed algorithm was tested over a large variety of the problems available from literature (32 problems) and also compared to many other algorithms existing in literature. The computational results verified the efficiency of the algorithm in finding good quality, and near-optimum solutions in a similar time compared to the best-published heuristics in literature. Samarghandi and Eshghi [40] used tabu search (TS) algorithm to solve a special case of the SRFLP in which the size of the machines was assumed to be different.

Kumar et al. in 2011 [29] presented a simple heuristic to determine a common linear machine sequence for multiple products with different operation sequences and a limited number of duplicate machine types available for the job. The heuristic was based on minimization of the total flow distance travelled by a product on the linear machine sequence. Datta et al. [13] preferred a permutation-based genetic algorithm to solve the SRFLP. The GA individuals are obtained by using rule-based permutations, which are improved towards the optimum value by means of specialized mutation and crossover operators. Ozcelik [36], proposed a hybrid GA to solve the single row layout design problem with unequal sized machines and unequal clearances. The algorithm was developed by hybridization of a genetic algorithm with a local search operator. Kothari and Ghosh [25] present a genetic algorithm called GENALGO in 2014 to solve large single row facility layout problem instances.

4 Bat Algorithm

Bat Algorithm (BA), which is a search algorithm inspired by social behavior of microbats and the phenomenon of echolocation to sense distance, proposed by Yang in 2010 [48]. For simplicity, the algorithm is based on idealizing some of the echolocation characteristics of bats, which are the following approximate or idealized rules:

- All bats use echolocation to sense distance, and they also "know" the difference between food/prey and background barriers in some magical way;
- Bats randomly fly with velocity v_i at position x_i with a fixed frequency $f_{\min}$, varying wavelength λ and loudness A_0 to search for prey. They can automatically adjust the wavelength (or frequency) of their emitted pulses and adjust the rate of pulse emission $r \in [0, 1]$, depending on the proximity of their target;
- Although the loudness can vary in many ways, it is assumed that the loudness varies from a large (positive) A_0 to a minimum constant value $A_{\min}$.

The basic steps of BA can be summarized as the pseudo code shown in Fig. 1.

At first step of BA, n number of bats, which are real-valued vectors with problem dimension d, randomly spread to the search space by taking into account lower and upper boundaries. Each bat is also defined by its position x_i, velocity v_i, pulse

Objective function $f(x)$, $x = (x_1,..., x_d)^T$
Initialize the bat population x_i $(i = 1,2,...,n)$ and v_i
Define pulse frequency f_i at x_i
Initialize the pulse rate r_i and the loudness A_i
while ($t <$ *max number of iterations*)
    Generate new solutions by adjusting frequency,
    And update velocities and locations/solutions
        if ($rand > r_i$)
        Select a solution among the best solutions
        Generate a local solution around the selected best solution
        end if
        Generate a new solution by flying randomly
        if ($rand < A_i$) & ($f(x_i) < f(x_*)$)
        Accept the new solutions
        Increase r_i and reduce A_i
        end if
    Rank the bats and find the current best (x_*)
end while
Postprocess results and visualization

Fig. 1 Pseudo code of the bat algorithm [16, 48, 50]

frequency f_i, loudness A_i and the pulse emission rate r_i. The pulse frequency is randomly assigned to bats that uniformly dispersed in $[f_{\min}, f_{\max}]$ while the pulse emission rate can simply be in the range of [0,1]. 0 means no pulses exist and 1 means that the pulse emission rate is maximum. Depending on the domain size of the problem of interest, frequency range can be changed. During the iterations, the new solutions x_i^t and velocities v_i^t at time step t can be calculated by

$$f_i = f_{\min} + (f_{\max} - f_{\min})\beta \tag{4}$$

$$v_i^t = v_i^{t-1} + \left(x_i^{t-1} - x_*\right)f_i \tag{5}$$

$$x_i^t = x_i^{t-1} + v_i^t \tag{6}$$

where $\beta \in [0, 1]$ is a random vector drawn from a uniform distribution. Here, x_* is the current global best location (solution), which is located after comparing all the solutions among all n number of bats.

For the local search part, once a solution is selected among the current best solutions, a new solution for each bat is generated locally using random walk

$$x_{new} = x_{old} + \varepsilon A^t \tag{7}$$

where $\varepsilon \in [-1, 1]$ is a scaling factor which is a random number while $A^t = \left\langle A_i^t \right\rangle$ is the average loudness of all the bats at time step t. The diversity of solutions is hereby increased.

Furthermore, as bats come to close their prey, the pulse emission rate r_i increases while the loudness A_i usually decreases. The update process as the iterations proceed as shown in Eq. (8),

$$A_i^{t+1} = \alpha A_i^t \quad r_i^{t+1} = r_i^0[1 - \exp(-\gamma t)] \tag{8}$$

where α and γ are constant. The amount of decrease, which is determined by α, plays a similar role as cooling factor in the simulated annealing algorithm [48, 49, 51, 52]. For any $0 < \alpha < 1$ and $\gamma > 0$,

$$A_i^t \to 0 \quad r_i^t \to r_i^0, \quad \text{as } t \to \infty \tag{9}$$

The choice of parameters' values requires some experimental analysis for bat algorithm. Due to this requirement, in this study design of experiments procedure is used to adjust values of some parameters at the beginning of the application. The details of application are presented in Sect. 5.

5 BA Application on SRFLP

Design of experiments (DOE) is an efficient procedure, which helps to investigate the effects of factors (input variables) on a response (output variable) at the same time. Before starting to experiments, the objective(s) of an experiment and response variable are determined and the factors for the study are selected. As mentioned by Montgomery [33], the objectives of the experiments may include the following:

- Determining which variables are most influential on the response *y*.
- Determining where to set the influential *x*'s so that *y* is always near the desired nominal value.
- Determining where to set the influential *x*'s so that variability in *y* is small.
- Determining where to set the influential *x*'s so that the effects of the uncontrollable variables are minimized.

These experiments consist of a series of runs/tests, in which purposeful changes are made to the levels of the factors.

Factorial designs are more efficient type of experiments to dealing with several factors. In this method, factors are varied together instead of one at a time. Therefore by a factorial design, all possible combinations of the levels of the factors are investigated in each complete trial or replication of the experiments.

The effect of a factor is defined to be the change in response produced by a change in the level of factors and called as "*main effect*". If the change in response between the levels of one factor is not same at all levels of the other factors, it means that there is an *interaction* between the factors.

In this study, five parameters of BA algorithm, which are thought to influence the outcome of the SRFLP, was examined by experiments firstly. The parameters were shown as factors and 3^5 full factorial design that is a factorial arrangement with *5* factors each at tree levels was used to investigate the appropriate values of these parameters. The levels of the factors were denoted as low (−1), intermediate (0) and high (1).

Table 2 shows the factors and their levels considered for the full factorial design.

The proposed design replicated twice and $3^5 \times 2 = 486$ treatment combination performed in random order on the data set with 15 departments presented by

Table 2 Algorithm parameters and their levels

FACTORS	LEVELS	VALUES		
		Low (−1)	Intermediate (0)	High (1)
Population size (*n*)	3	30	100	200
Maximum iterations (*N*)	3	100	250	500
F_{max}	3	1	3	5
Alpha (α)	3	0.1	0.5	0.9
Gamma (γ)	3	0.1	0.5	0.9

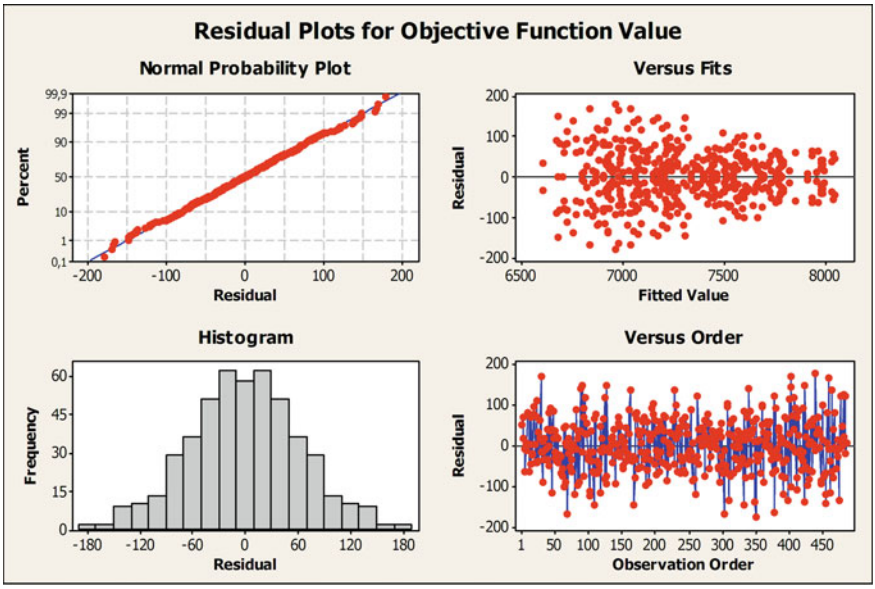


Fig. 2 Residual plots for objective function value

Amaral in 2006 [4]. According to the computational results, analyses were carried out with a statistical software package.

Figure 2 shows the four-in-one residual plot (i.e. normal probability plot of residuals, histogram of residuals, residuals versus fitted values and residuals versus order of the data) for objective function value that was taken as response. Residuals are used in regression and ANOVA analyses to indicate how well our model fits the data. Results indicate that the data are linear, random scattered and the histogram is symmetric bell-shaped. Therefore it can be said that the data are consistent with normality, independence and constant variance. The experimental design is fairly acceptable.

In order to see the effect of algorithm parameters on the objective function value, main effects plot, which is given in Fig. 3 was drawn. It is clearly evident that higher population size and maximum iteration values produce better results. However, the objective function value increases with the increase in F_{max}. Besides, the almost flat lines show that alpha and gamma values do not influence to the objective function value directly.

Figure 4 shows the interaction plot, which examines the interactions between factors. Parallel lines show that there is no interaction effect between the parameters. There is a less significant interaction effect in between F_{max} and alpha while adverse effect is observed in between F_{max} and gamma.

Based on the experimental results, optimal values of parameters are defined as given in Table 3.

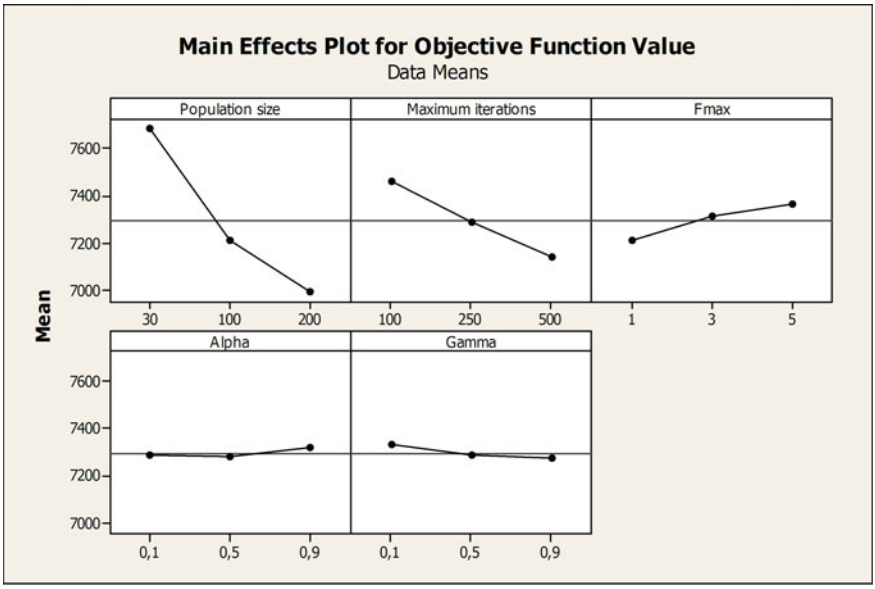


Fig. 3 Main effects plot for objective function value

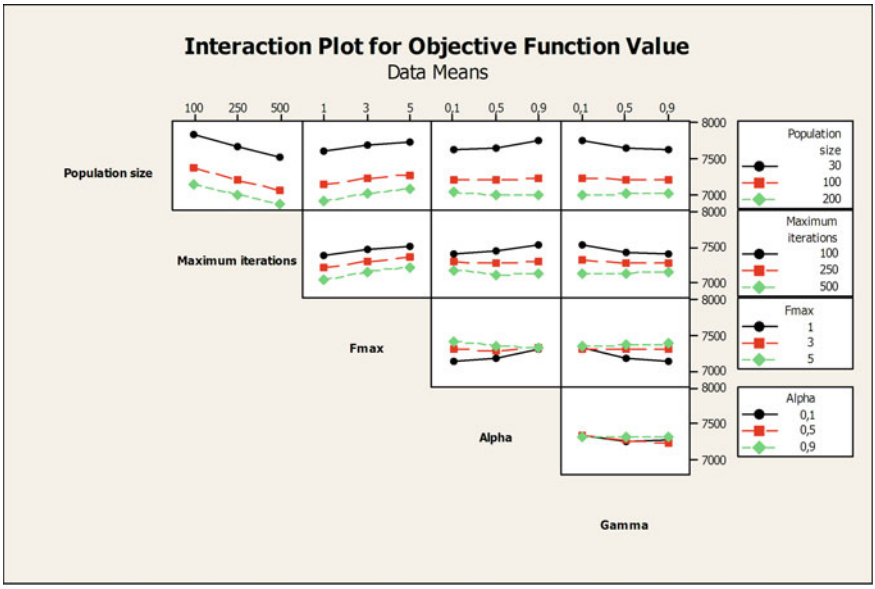


Fig. 4 Interaction plot for objective function value

Table 3 Optimal values of parameters

Parameters	Values
Population size (n)	200
Maximum iterations (N)	500
F_{max}	1
Alpha (α)	0.1
Gamma (γ)	0.9

Table 4 Optimal solutions for the six-problem set

Problem	Number of facilities	Optimal solution
LW5	5	151.0
S8H	8	2,324.5
S10	10	2,781.5
LW11	11	6,933.5
H20	20	15,549.0
H30	30	44,965.0

After defining the optimal settings, six-problem set from literature were used to evaluate the performance of the bat algorithm on the single row facility layout problem. Problems LW5 an LW11 were adopted from Love and Wong [32]. Problems S8H, S10 were found in Simmons [42]. For problems H20 and H30, the flow matrices were from Nugent et al. [35] and the dimensions of the facilities were obtained from Heragu and Kusiak [20]. The clearances between the facilities were assumed to be zero for these six problems. The algorithm was run 500 times for each problem on a PC equipped with 2.4 GHz Intel Core 2 Duo processor and 4 GB RAM.

Table 4 shows the optimal objective function values for the problem set. The optimal solutions for the problems LW5, S8H, S10 and LW11 were calculated by Amaral [4], whereas the others were provided by Anjos and Vanelli [8].

Table 5 shows the results obtained through bat algorithm and five other studies, which proposed some heuristic methods for the single row facility layout problem. In Table 5, the best solution in each row is set as bold. As seen in this table, all methods achieve optimal solution for the problems LW5, S8H, S10 and LW11. The proposed method by Heragu and Alfa [22] provides poor solution for problem H20 compared to others. The heuristic proposed by Datta et al. [13] outperform all the heuristics for the problems LW11, H20 and H30 in terms of the CPU time. Datta et al. [13] and Solimanpur et al. [44] obtain the same CPU time for problems LW5, S8H and S10. The reported result for problem H30 by Solimanpur et al. [44] is extremely low compared to the others. Its optimum value is 44,965.0. Hence, this could be a typo error.

The average objective function value of bat algorithm deviates from optimum solutions by 0, 0.23, 1.96, 2.97, 3.24 and 11.52 % in problems LW5, S8H, S10, LW11, H20 and H30 respectively. The bat algorithm finds optimum solution 345 times for the problem S8H, 254 times for the problem S10, 96 times for the problem

Table 5 Comparison of bat algorithm versus five other heuristics developed for SRFLP

Problems	Number of facilities	Heragu and Alfa [22]		Kumar et al. [28]		Solimanpur et al. [44]		Teo and Ponnambalam [46]		Datta et al. [13]		Bat algorithm		
		OFV	Time[a]	OFV	Time[a]	OFV	Time[a]	OFV	Time[a]	OFV	Time[a]	Best OFV	Average OFV	Average Time[a]
LW5	5	**151.0**	10.351	**151.0**	0.01	**151.0**	**0.00**	**151.0**	0.007	**151.0**	**0.00**	**151.0**	151.00	9.21
S8H	8	**2324.5**	11.803	**2324.5**	0.08	**2324.5**	**0.00**	**2324.5**	0.027	**2324.5**	**0.00**	**2324.5**	2329.82	11.29
S10	10	**2781.5**	19.815	**2781.5**	0.10	**2781.5**	**0.01**	**2781.5**	0.044	**2781.5**	**0.01**	**2781.5**	2835.96	12.58
LW11	11	**6933.5**	29.176	7265.5	0.12	**6933.5**	0.02	**6933.5**	0.057	**6933.5**	**0.01**	**6933.5**	7139.15	14.06
H20	20	15602.0	603.376	**15549.0**	8.60	**15549.0**	2.30	**15549.0**	3.286	**15549.0**	**0.10**	**15549.0**	16052.21	27.68
H30	30	45111.0	585.947	44466.5	91.80	**25000**	37.3	44965.5	82.350	44965.5	**0.74**	49938.5	50143.18	57.43

OFV Objective function value

[a] The CPU time for all methods is in second. The CPU time for [22] was obtained on an VAX 6420 mainframe computer. Kumar et al. [28] tested their proposed heuristic on a Sun 3/260 computer. The CPU time for [44] was obtained on a Pentium III 550 MHz PC. Teo and Ponnambalam [46] obtained the CPU time on a Pentium 4 (Prescott) of 3.36 GHz with 1 MB L2 cache. Datta et al. [13] executed their proposed algorithm in the Fedora 8 Linux environment in a Compaq Presario V6000 notebook having a 1.73 GHz processor with 1.0 GB RAM

LW11 and 84 times for the problem H20 out of 500 runs for each. Besides this, bat algorithm requires more computational time than the other heuristic methods except from method proposed by Heragu and Alfa [22]. Therefore, the performance of bat algorithm needs to be improved.

The bat algorithm balances exploration and exploitation during the search process. But it can get trapped in local minimum by varying loudness and pulse emission rate too quickly. In order to improve the exploration capability, the algorithm may be combined with other metaheuristics.

6 Conclusion

The rapid and promising advancement of metaheuristics in facility layout applications have provided a new perspective on this area. In this study the bat algorithm, which can be considered as a new algorithm in metaheuristics, is presented for solving the single row facility layout problem (SRFLP). The algorithm was designed to arrange a number of facilities on a line with minimum cost. Before application, the optimal settings of the bat algorithm parameters were determined through experimental analysis. 3^5 full factorial design was used to investigate the appropriate settings of the algorithm. Then the performance of the bat algorithm is tested on six-problem set selected from the literature.

Bat algorithms have been applied in almost every area of optimization, classifications, image processing, feature selection, scheduling, data mining etc. However, there is no application in the literature that compares the performances of the bat algorithm on the single row facility layout problem.

In this study, only one layout type is handled to evaluate the performance of the bat algorithm. In order to increase the performance of the bat algorithm for the single row facility layout problem, the algoritm can be initially combined with other metaheuristics. Multi-row layout problem, which includes both the horizontal and vertical material handling cost and distance in objective function, can also be a focus area for future research. Furthermore, some constraints like departments' closeness ratings and backtrack movements can be added to the problem.

References

1. Adolphson, D., Hu, T.C.: Optimal linear ordering. SIAM J. Appl. Math. **25**(3), 403–423 (1973)
2. Alba, E.: Parallel metaheuristics: a new class of algorithms. Wiley, Hoboken (2005)
3. de Alvarenga, A.G., Negreiros-Gomes, F.J., Mestria, M.: Metaheuristic methods for a class of the facility layout problem. J. Intell. Manuf. **11**(4), 421–430 (2000)
4. Amaral, A.R.S.: On the exact solution of a facility layout problem. Eur. J. Oper. Res. **173**(2), 508–518 (2006)

5. Amaral, A.R.S.: An exact approach for the one-dimensional facility layout problem. Oper. Res. **56**(4), 1026–1033 (2008)
6. Amaral, A.R.S.: A new lower bound for the single row facility layout problem. Discrete Appl. Math. **157**(1), 183–190 (2009)
7. Anjos, M.F., Kennings, A., Vannelli, A.: A semidefinite optimisation approach for the single-row layout problem with unequal dimensions. Discrete Optim. **2**(2), 113–122 (2005)
8. Anjos, M.F., Vannelli, A.: Computing globally optimal solutions for single-row layout problems using semidefinite programming and cutting planes. INFORMS J. Comput. **20**(4), 611–617 (2008)
9. Anjos, M.F., Yen, G.: Provably near-optimal solutions for very large single-row facility layout problems. Optimisation Methods Softw. **24**(4), 805–817 (2009)
10. Blum, C., Roli, A.: Hybrid metaheuristics: an introduction. Hybrid metaheuristics, pp. 1–30. Springer, Berlin (2008)
11. Braglia, M.: Heuristics for single-row layout problems in flexible manufacturing systems. Prod. Planning Control **8**(6), 558–567 (1997)
12. Burkard, R.E., Dell'Amico, M., Martello, S.: Assignment problems. Revised Reprint, Siam (2009)
13. Datta, D., Amaral, A.R., Figueira, J.R.: Single row facility layout problem using a permutation-based genetic algorithm. Eur. J. Oper. Res. **213**(2), 388–394 (2011)
14. Djellab, H., Gourgand, M.: A new heuristic procedure for the single row facility layout problem. Int. J. Comput. Integr. Manuf. **14**(3), 270–280 (2001)
15. Drira, A., Pierreval, H., Hajri-Gabouj, S.: Facility layout problems: a survey. Annu. Rev. Control **31**(2), 255–267 (2007)
16. Gandomi, A.H., Yang, X.S., Alavi, A.H., Talatahari, S.: Bat algorithm for constrained optimization tasks. Neural Comput. Appl. **22**(6), 1239–1255 (2013)
17. Glover, F.: Future paths for integer programming and links to artificial intelligence. Comput. Oper. Res. **13**(5), 533–549 (1986)
18. Hall, K.M.: An r-dimensional quadratic placement algorithm. Manage. Sci. **17**(3), 219–229 (1970)
19. Heragu, S.S., Kusiak, A.: Machine layout problem in flexible manufacturing systems. Oper. Res. **36**(2), 258–268 (1988)
20. Heragu, S.S., Kusiak, A.: Efficient models for the facility layout problem. Eur. J. Oper. Res. **53** (1), 1–13 (1991)
21. Heragu, S.S.: Invited review. Recent models and techniques for solving the layout problem. Eur. J. Oper. Res. **57**, 136–144 (1992)
22. Heragu, S.S., Alfa, A.S.: Experimental analysis of simulated annealing based algorithms for the layout problem. Eur. J. Oper. Res. **57**(2), 190–202 (1992)
23. Hungerländer, P., Rendl, F.: Semidefinite relaxations of ordering problems. Math. Program. **140**(1), 77–97 (2013)
24. Koopmans, T.C., Beckmann, M.: Assignment problems and the location of economic activities. Econometrica J. Econometric Soc. 53–76 (1957)
25. Kothari, R., Ghosh, D.: An efficient genetic algorithm for single row facility layout. Optim. Lett. **8**(2), 679–690 (2014)
26. Kouvelis, P., Chiang, W.C.: A simulated annealing procedure for single row layout problems in flexible manufacturing systems. Int. J. Prod. Res. **30**(4), 717–732 (1992)
27. Kouvelis, P., Chiang, W.-C.: Optimal and heuristic procedures for row layout problems in automated manufacturing systems. J. Oper. Res. Soc. **47**(6), 803–816 (1996)
28. Kumar, K.R., Hadjinicola, G.C., Lin, T.L.: A heuristic procedure for the single row facility layout problem. Eur. J. Oper. Res. **87**, 65–73 (1995)
29. Kumar, M.S., Islam, M.N., Lenin, N., Vignesh Kumar, D., Ravindran, D.: A simple heuristic for linear sequencing of machines in layout design. Int. J. Prod. Res. **49**(22), 6749–6768 (2011)
30. Letchford, A.N., Amaral, A.: A polyhedral approach to the single row facility layout problem. Working Paper. The Department of Management Science, Lancaster University (2011)

31. Lin, M.T.: The single-row machine layout problem in apparel manufacturing by hierarchical order-based genetic algorithm. Int. J. Clothing Sci. Techno. **21**(1), 31–43 (2009)
32. Love, R.F., Wong, J.Y.: On solving a one-dimensional space allocation problem with integer programming. INFOR **14**(2), 139–143 (1976)
33. Montgomery, D.C.: Design and analysis of experiments, 5th edn. Wiley, Hoboken (2001)
34. Neghabat, F.: An efficient equipment layout algorithm. Oper. Res. **22**(3), 622–628 (1974)
35. Nugent, C.E., Vollman, T.E., Ruml, J.: An experimental comparison of tecniques for the assignment of facilities to locations. Oper. Res. **16**(1), 150–173 (1968)
36. Ozcelik, F.: A hybrid genetic algorithm for the single row layout problem. Int. J. Prod. Res. **50** (20), 5872–5886 (2012)
37. Picard, J., Queyranne, M.: On the one dimensional space allocation problem. J. Oper. Res. **29** (2), 371–391 (1981)
38. Ponnambalam, S.G., Ramkumar, V.: A genetic algorithm for the design of single row layout in automated manufacturing systems. Int. J. Adv. Manuf. Technol. **18**, 512–519 (2001)
39. Romero, D., Sánchez-Flores, A.: Methods for the one-dimensional space allocation problem. Comput. Oper. Res. **17**(5), 465–473 (1990)
40. Samarghandi, H., Eshghi, K.: An efficient tabu algorithm for the single row facility layout problem. Eur. J. Oper. Res. **205**(1), 98–105 (2010)
41. Samarghandi, H., Taabayan, P., Jahantigh, F.F.: A particle swarm optimization for the single row facility layout problem. Comput. Ind. Eng. **58**(4), 529–534 (2010)
42. Simmons, D.M.: One-dimensional space allocation: an ordering algorithm. Oper. Res. **17**(5), 812–826 (1969)
43. Simmons, D.M.: A further note on one-dimensional space allocation. Oper. Res. **19**, 249 (1971)
44. Solimanpur, M., Vrat, P., Shankar, R.: An ant algorithm for the single row layout problem in flexible manufacturing systems. Comput. Oper. Res. **32**(3), 583–598 (2005)
45. Talbi, E.-L.: Metaheuristics from design to implementation. Wiley, Hoboken (2009)
46. Teo, Y.T., Ponnambalam, S.G.: A hybrid ACO/PSO heuristic to solve single row layout problem. In: Proceeding of the IEEE International Conference on Automation Science and Engineering (CASE), Washington DC, pp. 597–602 (2008)
47. Tompkins, J.A., White, J.A., Bozer, Y.A., Frazelle, E.H., Tanchoco, J.M.A., Trevino, J.: Facilities planning, 2nd edn. Wiley, New York (1996)
48. Yang, X.S.: A new metaheuristic bat-inspired algorithm. In: Cruz, C., Gonza´lez, J.R., Pelta, D.A., Terrazas, G. (eds.) Nature inspired cooperative strategies for optimization (NISCO 2010) studies in computational intelligence, vol. 284, pp. 65–74. Springer, Berlin (2010)
49. Yang, X.S.: Bat algorithm for multi-objective optimization. Int. J. Bio-Inspired Comput. **3**(5), 267–274 (2011)
50. Yang, X.S., Gandomi, A.H.: Bat algorithm: a novel approach for global engineering optimization. Eng. Comput. **29**(5), 464–483 (2012)
51. Yang, X.S.: Bat algorithm and cuckoo search: a tutorial, artificial intelligence, evolutionary computing and metaheuristics, pp. 421–434. Springer, Berlin (2013)
52. Yang, X.S., He, X.: Bat algorithm: literature review and applications. Int. J. Bio-Inspired Comput. **5**(3), 141–149 (2013)

Discrete Cuckoo Search Applied to Job Shop Scheduling Problem

Aziz Ouaarab, Belaïd Ahiod and Xin-She Yang

Abstract Discrete Cuckoo Search (DCS) algorithm is applied to solve combinatorial optimization problems. In this chapter we discuss how DCS solves the Job Shop Scheduling Problem (JSSP), one of the most difficult NP-hard combinatorial optimization problems. DCS is recently developed by Ouaarab et al. in 2013, based on Cuckoo Search (CS) which was proposed by Yang and Deb in 2009. DCS seeks solutions, in the discrete search space, via Lévy flights and a switching parameter p_a of the worst solution in the population. Its search uses a subtle balance between local and global random walks. This first version of DCS for JSSP is designed without using an advanced local search method or hybrid with other metaheuristics. Our experimental results show that DCS can find the optimum solutions for a number of JSSP instances.

1 Introduction

The aim of a scheduling problem is to find out the optimal feasible schedule in a finite or countable infinite set of schedules by optimizing the defined objective function. One of the well-known scheduling problems is the so-called Job Shop Scheduling Problem (JSSP) [1, 2]. JSSP occurs frequently in many manufacturing and service industries, where it is necessary to optimize the costs of production by minimizing the all production times. Many manufacturing scheduling problems can

A. Ouaarab (✉) · B. Ahiod
LRIT, Associated Unit to the CNRST (URAC 29), Mohammed V-Agdal University,
B.P. 1014, Rabat, Morocco
e-mail: aziz.ouaarab@gmail.com

B. Ahiod
e-mail: ahiod@fsr.ac.ma

X.-S. Yang
School of Science and Technology, Middlesex University, The Burroughs,
London NW4 4BT, UK
e-mail: x.yang@mdx.ac.uk

X.-S. Yang (ed.), *Recent Advances in Swarm Intelligence and Evolutionary Computation*,
Studies in Computational Intelligence 585, DOI 10.1007/978-3-319-13826-8_7

be considered as a JSSP model to find the optimum solution. Therefore, JSSP is both important and challenging, and thus researchers are still trying to develop efficient algorithms and propose some resolutions. However, as an NP-hard optimization problem, all existing algorithms are not able to find the optimum solution within a reasonable time for all problem instances. In fact, many recent studies have attempted to use metaheuristic algorithms to find an optimal or good solution in a good runtime [3].

Metaheuristic algorithms have demonstrated their potential and effectiveness in many applications, and thus have been used to solve a wide range of discrete or continuous optimization problems. As a result, they are becoming the most widely used alternatives to solve NP-hard problems and they can perform better than traditional algorithms for highly nonlinear problems. Two important advantages of metaheuristic algorithms are simplicity and flexibility. They are usually simple to implement, however they can often solve complex problems. In addition, they can be adapted easily (or combined with other methods) to solve many real-world optimization problems, from the fields of operations research, engineering to artificial intelligence [4, 5].

To balance their search between exploring and exploiting the solution space, metaheuristics use several search strategies. Ideally, an algorithm should concentrate the search process on some promising regions by starting the search with a population of initially guessed solutions, and then continues to iterative step-by-step as a sequence of solutions so as to reach the optimal solution as quickly as possible [5, 6]. Among the most popular metaheuristics, especially those inspired by nature, we can have a long list, to name a few, Genetic Algorithms (GA) [7, 8], Tabu Search (TS) [9], Simulated Annealing (SA) [10], Ant Colony Optimization (ACO) [11], Particle Swarm Optimization (PSO) [12], Bee Colony Optimization (BCO) [13], Monkey Search (MS) algorithm [14], Harmony Search (HS) algorithm [15], Firefly Algorithm (FA) [16], Intelligent Water Drops (IWD) [17], Bat Algorithm (BA) [18], Cuckoo Search (CS) [19], Flower Pollination Algorithm (FPA) [20, 21].

Most of these metaheuristics are nature-inspired, mimicking how nature successfully finds ingenious solutions to a very difficult problem (either biological, social or even physical-chemical) under dynamically changing environment. Not all algorithms perform equally well, some may obtain better results than others for a given problem, and there is no universally efficient algorithm to deal with all problems. So many challenges remain, especially for solving tough, NP-hard optimization problems [22].

Starting in the early 80s, as an important approach to solving difficult problems with a good compromise between the cost of resolution and quality of the solution, metaheuristics were applied to job shop scheduling problems. Some metaheuritics such as Tabu Search (TS) algorithm [23] were implemented to adopt the disjunctive graph representation by Taillard [24], and other studies included the TS introduced by Nowicki and Smutnicki in 1996 [25] and the improved version proposed by the same authors in 2005 [26]. Extensive case studies and approaches include the Simulated Annealing approach [27], Genetic Algorithm [28, 29] and its hybridized version with a local search procedure [30], Particle Swarm Optimization combined

with Simulated Annealing [31], Ant Colony Optimization combined with Taboo Search [32] and Tabu Search-simulated annealing hybrid [33], GRASP technique [34, 35], Bee Colony Optimization [36], Particle Swarm Optimization algorithm hybridized with Tabu Search [37].

The aim of this chapter is to apply the Discrete Cuckoo Search to study job shop scheduling problems. Therefore, this chapter is organized as follows: Sect. 2 presents an overview of the JSSP by discussing scheduling problem variants, schedule classes, problem representations and JSSP graphical modelling. Section 3 first briefly describes the standard CS, then discusses the improvements on the CS. Then, the proposed discrete CS is presented in detail. Section 4 solves a set of benchmarks from the OR-Library [38]. Finally, Sect. 5 concludes and summarizes the contributions of this work.

2 Job Shop Scheduling Problem

In manufacturing, production is a system that leads to the creation of a product through the use and transformation of resources. This production system is a set of resources that provide a production activity. To manage production, a set of processes is implemented to ensure to complete the manufacture of products using a set of data and forecasts. So, we must deal with one of the most important problems that affect directly the time and the cost of the production process. This problem is scheduling.

In general, a scheduling problem concerns one of the following:

- A set of tasks or jobs to perform,
- A set of resources or machines to use by these jobs,
- A program to identify, to properly allocate resources to tasks.

Scheduling problems are often defined in terms of four main elements: jobs, resources, constraints and objectives. So, it is important to discuss these four basic elements before studying scheduling problems [39, 40].

Jobs: A job is an elementary unit of the work that uses resources from a start date t_i to an end date c_i, whose implementation requires a time p_i such that $p_i = c_i - t_i$.

For such a problem, a job can be executed by piece or uninterrupted. In the case of a job shop scheduling problem, we consider the first category, each job with a set of operations subject to a number of constraints.

Resources: A resource is all that is intended to be used for performing a task, and available in limited quantities and for a limited time. In the manufacturing context, resources can be machines, workers, equipment, facilities and energy, etc. In a job shop scheduling problem, a resource is a machine.

Constraints: A constraint is the restriction or limit in values. Therefore, the constraints are the limits imposed by the environment.

Objectives: Objectives are the goals that guide the scheduling process. These goals are distinguished in classes according to the schedule time, resources, cost, and energy.

2.1 Scheduling Problem Variants

In scheduling, there are several variants of problem formulations, depending on the constraints, objectives or how we manage resources and the queues in front of these resources. In this context, there are many extensively studied such as the Flow Shop, Flexible Flow Shop, Job Shop, Flexible Job Shop and Open Shop. We now outline them briefly.

2.1.1 Flow Shop

All machines are in series. Each job (say, j) must be processed on each one of the available m machines. By following the same sequence, all the jobs must be carried out sequentially, first on machine 1 of the series, then on machine 2, etc. At each machine, there is a queue of the jobs that come from the previous machine. While waiting in the queue, a job cannot pass another, which means that all queues operate under the First In First Out (FIFO) method [39].

2.1.2 Flexible Flow Shop

In essence, a flexible flow shop problem is the generalization of a flow shop problem where each machine is replaced by a set of identical machines in parallel. All these sets of identical machines are in series. So, all jobs have to be processed first on the first set and so on. At each set a job j has to be processed on only one machine at a time [39].

2.1.3 Job Shop

In a job shop with m machines, each job has to be carried out a predetermined sequence of machines. The sequence of m machines for one job is different from that for another, which cannot be changed during the process. At most one job can be processed on each machine at a time [39].

2.1.4 Flexible Job Shop

The extension of a job shop scheduling problem is called a flexible job shop, and a set of identical machines (in parallel) are used to replace each single machine. In addition, a job is carried out by any machine from the given set and each job has its own sequence of sets. At a parallel set, any job *j* can be processed on only any one machine and no two machines should process the same job at the same time [39].

2.1.5 Open Shop

In the context of an open shop, each job can be done only on each machine, while one machine can only carry out one job at a time. However, unlike the job shop, the sequence of job-processing machines can vary freely and different jobs may have different sequences as well. Some of job processing times may be zero [39].

2.2 Schedule Classes

To order to use the most effective methods to find an optimal solution or schedule, any knowledge about the search space is useful. Therefore, it is important to organize the solutions as classes. Each class has its own feature that is important to know before the construction of the feasible schedules. This will allow us to properly exploit the time of the schedule.

The relations between these classes are inclusion or intersection. If we begin with the largest class which encompassing the other classes, we will find the feasible schedule class. There are many classes, but three classes are emphasized here: active, non-delayed and optimal schedules. It worth mentioning that an active schedule is that for which no operation can be put into an empty hole, while preserving feasibility, without delaying another operation. In this case, no shift to the left is possible. The non-delayed schedules are defined in the case that if and only if no operation, available for processing, is on the queue while a machine is free to run. A non-preemptive, non-delayed schedules are active schedules and the reverse is not necessarily true. A non-delayed schedule class can intersect with the optimal solution class. For details about classes, please refer to more advance literature [39–41].

2.3 Problem Representation

There are different ways of formulating a job shop scheduling problem (JSSP), though in essence a JSSP can be considered as a discrete optimization problem, and such problems can have important applications in manufacturing and engineering.

The primary task concerning the JSSP is to transform manufacturing decisions to detailed implementation instructions, defined by the production manager program so as to minimize the production time and cost. The typically output of the JSSP is a schedule that provides detailed assignments of jobs with specific start and end dates. Obviously, such a schedules must be subject to a set of constraints.

Loosely speaking, each one of the n jobs in $J = \{1, \ldots, n\}$ $(n \geq 1)$, in a JSSP, has to be processed via the path of m machines in $M = \{1, \ldots, m\}$ $(m \geq 1)$, and each job can be processed once and once only, in a given sequence. For each job, the sequence of m machines can be different, and once fixed, it cannot be changed during the process. An operation is equivalent to the processing of one job on a machine in a specific order. In general, each job j $(1 \leq j \leq n)$ can require m combined operations $(o_{j1}, o_{j2}, \ldots, o_{jm})$ so as to complete the work. Obviously, one operation can be processed on only one of the m machines at a given, fixed time. Each machine can only process at most one operation at a time, and a machine that is processing one job cannot be interpreted by another job, The main objective or aim of solving a JSSP is to try to find a feasible schedule of operations so as to minimize the maximum completion time or makespan C_{max}. In essence, the makespan can be defined as the complete time for carrying out all the operations on m machines for all the n jobs. Detailed formulation can be found in [2, 42, 43].

In general, the sequence for one job can be different from another. Obviously, the number of operations for a given job cab be the number of machines in most cases, which means that each job is processed once and only once. For such reasons, the total number of operations is essentially $|O| = n \times m$, or $n \times m$ instances. The following descriptions are mainly based on the formulation by Ponsich et al. [41] and others [31, 44]. Now let $O = \{o_{ji}, j \in \{1, \ldots, n\}$ and $i \in \{1, \ldots, m\}\}$ corresponds to the set of all the operations in a JSSP. There is a fixed processing time p_{ji} for each operation $o_{ji} \in O$. Two conditions are: 1) an operation o_{ji} can only be processed on a machine only if the machine is idle and ready. 2) Each operation o_{ji} can start only if its predecessor operation $o_{j(i-1)}$ is completed. In essence, a schedule sequence can be coded as a vector of completion times for all the operations, that is $(C_{ji}, i \in \{1, \ldots, m\}$ and $j \in \{1, \ldots, n\})$. Thus, we use the following notations,

1. o_{ji}: the operation i of job j,
2. p_{ji}: the processing time of o_{ji},
3. C_{ji}: the completion time of o_{ji}.

A schedule or solution can be generated by permutation, which can lead to the JSSP complexity of $O((n!)^m)$ [41]. The objective of a JSSP is to minimize the overall completion time of all the operations, or the makespan C_{max}, subject to the following constraints:

- Once a machine is processed a job, it cannot be interrupted.
- A machine can only process (at most) one job at a time.
- An operation cannot start if its predecessor has not been completed, and the starting time must be later than the completion time of its predecessor.

Therefore, the scheduling problem can be written in a compact form as

$$\text{Minimize } C_{max} = C_{n\times m}. \tag{1}$$

2.4 JSSP Graphical Modelling

For simplicity in describing the JSSP scheduling process, the so-called disjunctive graph and Gantt chart methods are often used.

2.4.1 Disjunctive Graph

The fundamental ideas of a disjunctive graph were proposed by Roy and Sussmann in 1964 [45]. Briefly speaking, a disjunctive graph representation (Fig. 1) can be denoted as $G = (A, V, E)$ where V is a set of nodes (corresponding to the set of operations O) with a (starting) source S and a (finishing) sink T. In addition, A corresponds to a set of conjunctive arcs showing the sequencing or precedence constraints, while E consists of a set of disjunctive edges, linking two operations to be processed on the same machine. Furthermore, the edge represents the time of the operation.

2.4.2 Gantt Diagram

By far, the most common representation for a JSSP is to use the so-called Gantt chart where an operation is represented by a segment or a horizontal bar with its length being proportional to the time of operation. Consequently, a temporal scale is used to indicate the occupation of all machines by different tasks and idle times as well as the potential unavailability of machines. As shown in Figs. 2 and 3, two types of Gantt

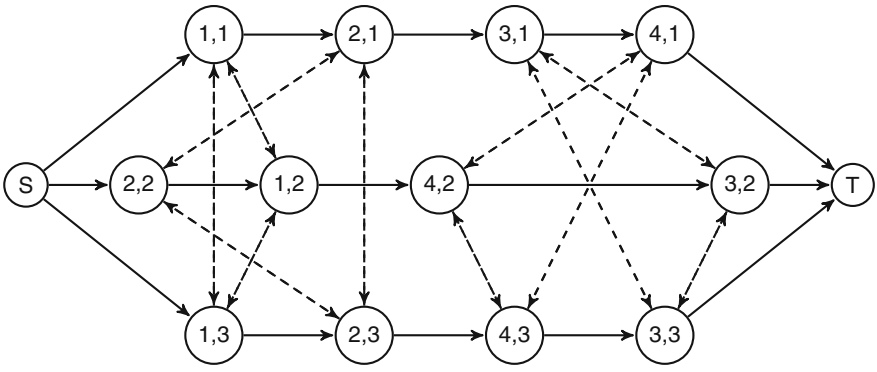


Fig. 1 A JSSP example of a disjunctive graph representation

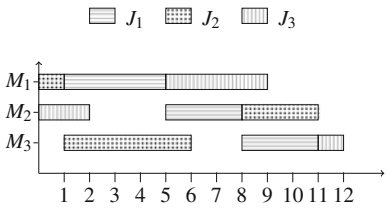


Fig. 2 Machine Gantt chart

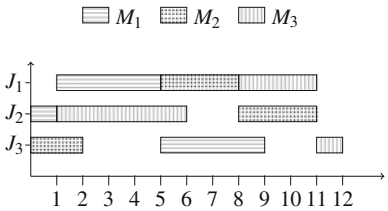


Fig. 3 Job Gantt chart

charts are used. A machine Gantt chart consists of a horizontal line for each machine where the line displays the various operations in sequence periods and periods of idleness of the machines. On the other hand, a job Gantt chart provides the overview the sequence of job operations where each job is represented by a line with the periods of execution of operations and waiting periods for machines [46, 47].

3 Discrete Cuckoo Search for JSSP

Now the discrete cuckoo search approach for solving the JSSP is presented in detail.

3.1 Cuckoo Search

Cuckoo Search (CS) [19] is inspired by the brood parasitism behaviour of some cuckoo species [48]. CS uses Lévy flights [49] to enhance the search both locally and globally where a solution in the search space is represented by a nest or an egg. Lévy flights are a model of random walks characterized by their step lengths which obey a power-law distribution. Most of the Lévy steps are relatively small, with occasional large jumps.

The standard CS was developed by Yang and Deb in 2009, initially designed for solving multimodal functions. The main ideas can be summarized as the following

ideal rules: (1) Each cuckoo lays one egg at a time and selects a nest randomly; (2) The best nest with the highest quality egg can pass onto the new generations; (3) The number of host nests is fixed, and the egg laid by a cuckoo can be discovered by the host bird with a probability $p_a \in [0, 1]$.

Algorithm 1 Improved Cuckoo Search

```
Objective function f(x), x = (x_1, ..., x_d)^T
Generate initial population of n host nests x_i (i = 1, ..., n)
while (t <MaxGeneration) or (stop criterion) do
    Start searching with a fraction (p_c) of smart cuckoos
    Get a cuckoo randomly by Lévy flights
    Evaluate its quality/fitness F_i
    Choose a nest among n (say, j) randomly
    if (F_i > F_j) then
        replace j by the new solution;
    end if
    A fraction (p_a) of worse nests are abandoned and new ones are built;
    Keep the best solutions (or nests with quality solutions);
    Rank the solutions and find the current best
end while
Postprocess results and visualization
```

Here, p_a is a switching parameter that guides the search style in the solution space. CS can perform a balanced choice between local and the global explorative random walks. The local random walk can be written as

$$x_i^{t+1} = x_i^t + \alpha s \otimes H(p_a - \varepsilon) \otimes (x_j^t - x_k^t), \tag{2}$$

where x_j^t and x_k^t are two different solutions selected by a permutation randomly, $H(u)$ is a Heaviside function, ε is a random number drawn from a uniform distribution, and s is the step size. On the other hand, the global random walk is carried out by using Lévy flights

$$x_i^{t+1} = x_i^t + \alpha L(s, \lambda), \tag{3}$$

where

$$L(s, \lambda) = \frac{\lambda \gamma(\lambda) \sin(\pi \lambda / 2)}{\pi} \frac{1}{s^{1+\lambda}}, (s \gg s_0 > 0). \tag{4}$$

Here $\alpha > 0$ is the step size scaling factor, which should be related to the scales of the problem of interest, and s_0 is a small constant (s_0 = 0.001–0.1 can be used). Lévy flights have an infinite variance with an infinite mean [5, 19].

3.2 Discrete Cuckoo Search

In the continuous space, a solution vector is generally represented by its real coordinates. In fact, to move from the actual position to a new one, we change only the current values of the coordinates. A small step is done by a small modification on the coordinate values and a big change of the coordinate values generates a big jump.

In the discrete space, a solution can be denoted by a set of acceptable combinations or permutations that are positioned, according to the order and the relations between their components. So, the meaning of locations and distances changes and becomes more complex. To locate a solution in the discrete space in terms of using permutations, we need to verify that the new found solution is acceptable. In many problems, finding a feasible solution is relatively difficult when we consider all the constraints of this problem. On the other hand, moving in this search space requires the proper definition of distances and neighbourhood, and thus requires proper operators and perturbations according to the type of problem.

This discrete version of Cuckoo Search (DCS) algorithm, developed by Ouaarab et al. [6, 50], is designed and adapted to solve discrete optimization problems such as the Travelling Salesman Problem [50, 51], Quadratic Assignment Problem and Job Shop Scheduling Problem. For solving a JSSP, DCS adopts its own procedure without any subordinate heuristic used generally to improve results. DCS is applied to solve the JSSP in order to show its performance against other metaheuristics rather than hybrids. In the DCS, Lévy flights have a control on all moves in the solution space. A global random walk is performed to explore the space. In fact, we will show how to represent a solution in the space and how to move from the current solution to another one using Lévy flights. The switching parameter p_a is also used to diversify the search to another unexplored new areas.

DCS improves its search strategy by inspiring a new notion from the way how cuckoos exploit and explore new nests. Such cuckoo 'intelligence' can find much better nests/solutions. Studies showed that cuckoos can engage a kind of surveillance, before and after brooding, on nests that are likely to be a host [48]. In this case, we are talking about a second level of local search performed by a fraction of intelligent cuckoos around the current solutions. The new mechanism can be introduced in the CS algorithm by a new population fraction p_c. The population is structured around three types of cuckoos: (1) A cuckoo, seeking new solutions that are much better than the solution of an individual, randomly selected, in the population; (2) A fraction p_a of cuckoos, seek new solutions far from the best solution in the population; (3) The new fraction p_c [50].

To move from a current solution to another, we now use a step that will be associated to the value generated by Lévy flights. Concerning the solution space of the studied problem, we design a model of operators and thus associate each operator with a specific step. To facilitate the generation of these steps via Lévy

flights, we calculate them in an interval between 0 and 1. Therefore, according to the value given by Lévy flights in this interval we can choose the appropriate step.

Obviously, two important components in the application of DCS on JSSP are: solutions and moves. Each combinatorial optimization problem can have a different solution/search space. So how to define a solution can be tricky. How to move from one solution to another can be even harder. In a JSSP, a schedule/solution can be interpreted by DCS as one egg in a nest, which represents an individual in the population.

3.3 JSSP Solution

To illustrate this idea further, we use a simplified example. Considering an instance of a JSSP, Table 1 contains (for example) 3 jobs and 4 machines where each job operation is associated with its machine and processing time. Here, a solution of this JSSP instance is a permutation of operations that can be generated randomly for the initial solution as shown in Fig. 4. The solution space is all the possible solutions/permutations for this JSSP instance.

To generate a JSSP solution from an integer series (or a random integer series in the case of random solutions), we can take each integer from this series and apply the operation: $(i \ mod \ n) + 1$, where i is a series integer and n is the number of jobs. In the current example $n = 3$, so the result is a permutation of jobs, or the permutation of the operations as showed in Fig. 4.

From Table 1 and the solution shown in Fig. 4, we can design a Gantt chart (Fig. 5) for this solution (schedule). In the DCS, a schedule with the good (minimum) completion time is selected to generate the next generation schedules.

Table 1 A 3 × 4 JSSP problem

Jobs	Machine sequence				Processing times			
1	1	2	3	4	$p_{11} = 10$	$p_{12} = 8$	$p_{13} = 4$	$p_{14} = 2$
2	2	1	4	3	$p_{21} = 8$	$p_{22} = 3$	$p_{23} = 5$	$p_{24} = 6$
3	1	2	4	3	$p_{31} = 4$	$p_{32} = 7$	$p_{33} = 3$	$p_{34} = 5$

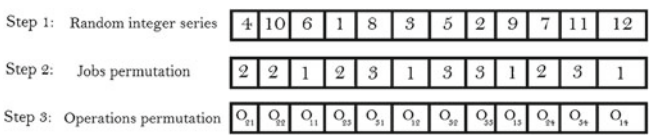


Fig. 4 Procedure of generating a random initial solution

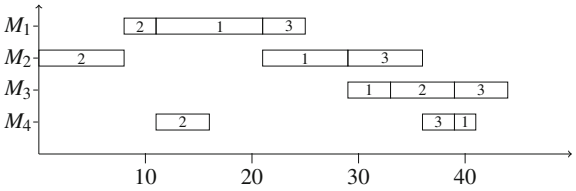


Fig. 5 Gantt chart for the solution shown in Fig. 4

3.4 Moving in Space

In order to move from one solution to another in the combinatorial search space, the move can be generated by considering step lengths or topologies. A step length is represented by the distance between solutions in a given search space. In terms of topology, one solution is moved to another by passing from one topology to another one. This case concerns a different kind of moves or operators.

In this application of the DCS to solve a JSSP, we propose three different moves or operators that are controlled by or associated with Lévy flights. The first move operator is the insertion move (Fig. 6) that removes the integer in an indicated position '2' and inserts it in the second indicated position '6'. The second move operator is the **swapping move** operator (Fig. 7) that swaps to integers which are located respectively in position '2' and '6'. The third operator is the **inversion move** operator (Fig. 8) that reverses integers' order between two positions, i.e., '2' and '6' in the example.

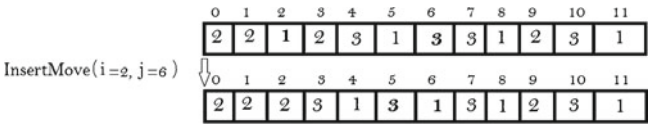


Fig. 6 Insert move operator

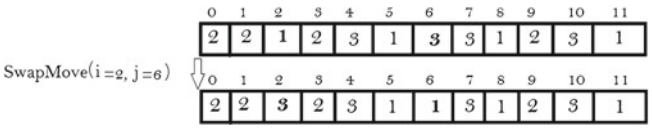


Fig. 7 Swap move operator

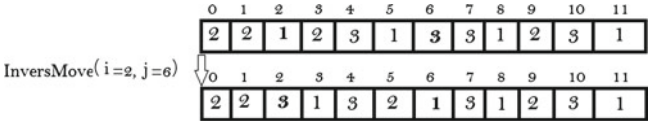


Fig. 8 Inverse move operator

These moves are associated with random values following a Lévy distribution. In fact, a balance search in JSSP combinatorial space should be between restricted and large areas or topologies when appropriate.

4 Experimental Results

In order to validate the proposed approach and show the performance, the DCS algorithm has been implemented and tested on 16 instances among the benchmarks taken from the OR-Library [38]. The DCS algorithm has been coded in JAVA language under the 32 bits Seven Operating System, and simulated on a laptop with the configurations of Intel(R) Core^{TM} 2 Duo 2.00 GHz, and 2 GB of RAM. The parameters used are summarized in Table 2, and the parameters were based on some preliminary parametric studies of p_a, p_c and the population size n.

In each case study, 10 independent runs of the DCS algorithm have been carried out and the test results are summarized in Table 3. The first column shows the name of the instance and the best known solution in parenthesis, the column 'best' shows the completion times of the best solution found by the algorithms and the column 'average' denotes the average solution completion time of the independent runs of the algorithms, and the column 'PDav(%)' denotes the percentage deviation of the average solution completion time over the optimal solution completion time of 10 runs, as shown in Eq. (5):

$$\text{PDav}(\%) = \frac{\text{average solution length} - \text{best known solution length}}{\text{best known solution length}} \times 100 \quad (5)$$

All numbers shown in bold in Table 3 mean that the found solutions have the same completion time over all the 10 runs.

As shown in Table 3, after testing some JSSP instances, comparisons have been made (for all the tested instances) with the results by the recently published discrete version of PSO for JSSP [31]. Their discrete PSO did not use any advanced local search methods or subordinate heuristics, which is the same case as in DCS. So, we can easily compare these two methods in terms of performance fairly.

Table 2 Parameter settings for the DCS algorithm

Parameter	Value	Meaning
n	30	Population size
P_a	0.2	Portion of bad solutions
p_c	0.6	Portion of smart cuckoos
MaxGeneration	300	Maximum number of iterations
α	0.01	Step size
λ	1	Index

Table 3 Computational comparison between PSO and DCS algorithms

	DCS			PSO [31]		
Instance (opt)	Best	Average	*PDAv* (%)	Best	Average	*PDAv* (%)
Abz5 (1234)	1,239	1,239.6	0.00	–	–	–
Abz6 (943)	**943**	946.4	0.00	–	–	–
Ft06 (55)	**55**	**55**	**0.00**	55	56.1	0.02
Ft10 (930)	945	966.8	0.03	985	1,035.6	0.11
Ft20 (1165)	1,173	1,178.8	0.01	1,208	1,266.9	0.08
La01 (666)	**666**	**666**	**0.00**	666	668.6	0.00
La02 (655)	**655**	**655**	**0.00**	–	–	–
La03 (597)	604	607	0.01	–	–	–
La04 (590)	**590**	**590**	**0.00**	–	–	–
La06 (926)	**926**	**926**	**0.00**	926	926	0.00
La11 (1222)	**1,222**	**1,222**	**0.00**	1,222	1,222	0.00
La16 (945)	946	967.7	0.02	956	986.9	0.04
La21 (1046)	1,055	1,070.1	0.02	1,102	1,128.4	0.07
La26 (1218)	**1,218**	**1,218**	**0.00**	1,263	1,312.6	0.07
La31 (1784)	**1,784**	**1,784**	**0.00**	1,789	1,830.4	0.02
La36 (1268)	1,297	1,313.6	0.03	1,373	1,406.2	0.10

Based on the three columns [Best, Average and *PDAv* (%)] in Table 3, we can see that DCS performs better than PSO in the tested instances, which is confirmed by Fig. 9 by indicating the lower curve associated with DCS. Hence, DCS is more appropriate to be adapted to solve JSSPs with good results. Such good performance can be explained basically by the balance performed by CS to explore and exploit the search space via Lévy flights and the switch parameter p_a. The performance is also enhanced by the added improvement presented by the new smart cuckoo fraction p_c.

These results confirm that this first version of DCS is more adaptable to this kind of combinatorial optimization problems. The proposed DCS can also provide a

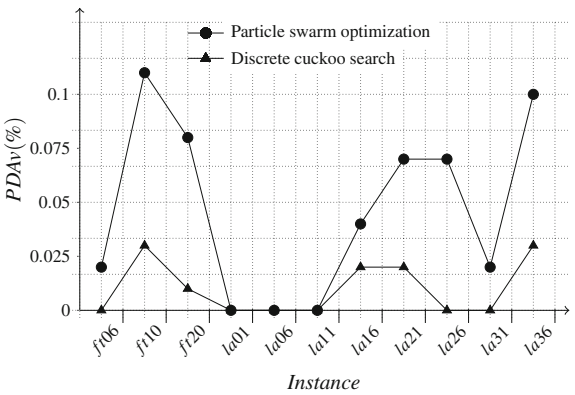


Fig. 9 PDAv (%) for 11 OR-Library instances

good balance between intensification and diversification. The use of Lévy flights, the reduced number of parameters and the introduced fraction p_c of smart cuckoos has enhanced the performance of the population by improving its structure. One further advantage of the improved DCS is the level-independence varieties for the best positions. Each cuckoo (according to its category) in the population can follow one of the three different strategies (discussed in the DCS part). So, it is more likely to find good solutions in the areas unexplored by other metaheuristics.

It can be expected that our proposed approach can be easily improved by adding a good local search method or by hybridization with other metaheuristics.

5 Conclusion

We have discussed a class of well-known job shop scheduling problems that are important in real-world applications. In this chapter, we have prosed an improved discrete version of Cuckoo Search (CS) to solve this NP-Hard problem. The new discrete CS (DCS) developed to solve combinatorial optimization problems, can enhance the performance of CS by reconstructing the population and introducing a new kind of intelligent cuckoos category. Such adaptation to the JSSP is based on a study of terminology interpretation used in CS and in its inspiration sources. It provides an effective to generate JSSP solutions and to move from one solution to another new one, using new move operators in terms of distances and neighbourhood.

DCS has been implemented and its performance has been tested on sixteen benchmark JSSP instances. Its performance has been compared with the discrete PSO [31]. The results of the comparison have shown that the discrete CS outperforms the discrete PSO for solving JSSP instances. The good performance of DCS can be explained by the management of intensification and diversification through Lévy flights and the structure of the population. The new population structure contains a variety of cuckoos that use multiple research methods adopted by p_a and p_c fractions. So, the combination of more than one search components and the high level use of the best solution from the population provides an effective approach for problem solving for combinatorial problems. The aim of such adaptation is to give a good example of mimicking nature and try to design new algorithms to solve very complex problems more efficiently. It is also important to link the efficiency of the algorithms with the simplicity of its implementation.

It can be expected that it will be useful to investigate how DCS can be improved and hybridized with other methods. DCS shows, in some recent studies that it can be adapted to solve the Traveling Salesman Problem and Quadratic Assignment Problems, which confirms that DCS can provide more flexibility to solve (or by hybridization) more combinatorial optimization problems. In addition, DCS has fewer number of parameters and thus it is easier to implement and to tune these parameters. Furthermore, further studies can be fruitful when focusing on the applications of DCS into other combinatorial problems.

References

1. Applegate, D., Cook, W.: A computational study of the job-shop scheduling problem. ORSA J. comput. **3**(2), 149–156 (1991)
2. Manne, A.S.: On the job-shop scheduling problem. Oper. Res. **8**(2), 219–223 (1960)
3. Blum, C., Roli, A.: Metaheuristics in combinatorial optimization: Overview and conceptual comparison. ACM Comput. Surv. (CSUR) **35**(3), 268–308 (2003)
4. Gandomi, A.H., Yang, X.S., Talatahari, S., Alavi, A.H.: Metaheuristic applications in structures and infrastructures. Newnes (2013)
5. Yang, X.S.: Nature-inspired metaheuristic algorithms. Luniver Press, Bristol (2010)
6. Ouaarab, A., Ahiod, B., Yang, X.S.: Improved and discrete cuckoo search for solving the travelling salesman problem. In: Cuckoo Search and Firefly Algorithm, pp. 63–84. Springer, Berlin (2014)
7. Davis, L., et al.: Handbook of genetic algorithms, vol. 115. Van Nostrand Reinhold, New York (1991)
8. Sivanandam, S., Deepa, S.: Genetic Algorithm Optimization Problems. Springer, Berlin (2008)
9. Glover, F., Laguna, M.: Tabu search. Springer, Berlin (1999)
10. Van Laarhoven, P.J., Aarts, E.H.: Simulated annealing. Springer, Berlin (1987)
11. Dorigo, M., Blum, C.: Ant colony optimization theory: A survey. Theoret. Comput. Sci. **344** (2), 243–278 (2005)
12. Kennedy, J.: Particle swarm optimization. In: Encyclopedia of Machine Learning, pp. 760–766. Springer, Berlin (2010)
13. Karaboga, D., Basturk, B.: On the performance of artificial bee colony (abc) algorithm. Appl. Soft Comput. **8**(1), 687–697 (2008)
14. Mucherino, A., Seref, O.: Monkey search: a novel metaheuristic search for global optimization. In: Data Mining, Systems Analysis and Optimization in Biomedicine, vol. 953, pp. 162–173. AIP Publishing, New York (2007)
15. Geem, Z.W., Kim, J.H., Loganathan, G.: A new heuristic optimization algorithm: harmony search. Simulation **76**(2), 60–68 (2001)
16. Yang, X.S.: Firefly algorithms for multimodal optimization. In: Stochastic algorithms: foundations and applications, pp. 169–178. Springer, Berlin (2009)
17. Shah-Hosseini, H.: The intelligent water drops algorithm: a nature-inspired swarm-based optimization algorithm. Int. J. Bio-Inspired Comput. **1**(1), 71–79 (2009)
18. Yang, X.S.: A new metaheuristic bat-inspired algorithm. In: Nature Inspired Cooperative Strategies for Optimization (NICSO 2010), pp. 65–74. Springer, Berlin (2010)
19. Yang, X.S., Deb, S.: Cuckoo search via lévy flights. In: Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on, pp. 210–214. IEEE, New York (2009)
20. Yang, X.S.: Nature-Inspired Optimizaton Algorithms. Elsevier (2014)
21. Yang, X.S., Karamanoglu, M., He, X.: Flower pollination algorithm: a novel approach for multiobjective optimization. Eng. Optim. **46**, 1222–1237 (2014)
22. Wolpert, D.H., Macready, W.G.: No free lunch theorems for optimization. Evolut. Comput. IEEE Trans. **1**(1), 67–82 (1997)
23. Dell'Amico, M., Trubian, M.: Applying tabu search to the job-shop scheduling problem. Ann. Oper. Res. **41**(3), 231–252 (1993)
24. Taillard, E.D.: Parallel taboo search techniques for the job shop scheduling problem. ORSA J. Comput. **6**(2), 108–117 (1994)
25. Nowicki, E., Smutnicki, C.: A fast taboo search algorithm for the job shop problem. Manage. Sci. **42**(6), 797–813 (1996)
26. Nowicki, E., Smutnicki, C.: An advanced tabu search algorithm for the job shop problem. J. Sched. **8**(2), 145–159 (2005)
27. Van Laarhoven, P.J., Aarts, E.H., Lenstra, J.K.: Job shop scheduling by simulated annealing. Oper. Res. **40**(1), 113–125 (1992)

28. Davis, L.: Job shop scheduling with genetic algorithms. In: Proceedings of an International Conference on Genetic Algorithms and Their Applications, vol. 140 (1985)
29. Della Croce, F., Tadei, R., Volta, G.: A genetic algorithm for the job shop problem. Comput. Oper. Res. **22**(1), 15–24 (1995)
30. Gonçalves, J.F., de Magalhães Mendes, J.J., Resende, M.G.: A hybrid genetic algorithm for the job shop scheduling problem. Eur. J. Oper. Res. **167**(1), 77–95 (2005)
31. Lin, T.L., Horng, S.J., Kao, T.W., Chen, Y.H., Run, R.S., Chen, R.J., Lai, J.L., Kuo, I., et al.: An efficient job-shop scheduling algorithm based on particle swarm optimization. Expert Syst. Appl. **37**(3), 2629–2636 (2010)
32. Huang, K.L., Liao, C.J.: Ant colony optimization combined with taboo search for the job shop scheduling problem. Comput. Oper. Res. **35**(4), 1030–1046 (2008)
33. Zhang, C.Y., Li, P., Rao, Y., Guan, Z.: A very fast ts/sa algorithm for the job shop scheduling problem. Comput. Oper. Res. **35**(1), 282–294 (2008)
34. Aiex, R.M., Binato, S., Resende, M.G.: Parallel grasp with path-relinking for job shop scheduling. Parallel Comput. **29**(4), 393–430 (2003)
35. Binato, S., Hery, W., Loewenstern, D., Resende, M.: A grasp for job shop scheduling. In: Essays and Surveys in Metaheuristics, pp. 59–79. Springer, Berlin (2002)
36. Chong, C.S., Low, M.Y.H., Sivakumar, A.I., Gay, K.L.: A bee colony optimization algorithm to job shop scheduling. In: Simulation Conference, 2006. WSC 06. Proceedings of the Winter, pp. 1954–1961. IEEE, New York (2006)
37. Sha, D., Hsu, C.Y.: A hybrid particle swarm optimization for job shop scheduling problem. Comput. Ind. Eng. **51**(4), 791–808 (2006)
38. Beasley, J.E.: Or-library: distributing test problems by electronic mail. J. Oper. Res. Soc. pp. 1069–1072 (1990)
39. Pinedo, M.L.: Scheduling: theory, algorithms, and systems. Springer, Berlin (2012)
40. T'kindt, V., Scott, H., Billaut, J.C.: Multicriteria scheduling: theory, models and algorithms. Springer, Berlin (2006)
41. Ponsich, A.: Coello Coello, C.A.: A hybrid differential evolution—tabu search algorithm for the solution of job-shop scheduling problems. Appl. Soft Comput. **13**(1), 462–474 (2013)
42. Błażewicz, J., Domschke, W., Pesch, E.: The job shop scheduling problem: Conventional and new solution techniques. Eur. J. Oper. Res. **93**(1), 1–33 (1996)
43. Jain, A.S., Meeran, S.: Deterministic job-shop scheduling: Past, present and future. Eur. J. Oper. Res. **113**(2), 390–434 (1999)
44. Cheng, T., Peng, B., Lü, Z.: A hybrid evolutionary algorithm to sovle the job shop scheduling problem. Ann. Oper. Res. pp. 1–15, 2013. http://link.springer.com/article/10.1007
45. Roy, B., Sussmann, B.: Les problemes d'ordonnancement avec contraintes disjonctives. Note ds **9** (1964)
46. Esquirol, P., Lopez, P., professeur de robotique) Lopez, P.: L'ordonnancement. Économica (1999)
47. Herrmann, J.W.: Handbook of production scheduling, vol. 89. Springer, Berlin (2006)
48. Payne, R.B.: The cuckoos, vol. 15. Oxford University Press, Oxford (2005)
49. Shlesinger, M.F., Zaslavsky, G.M., Frisch, U.: Lévy flights and related topics in physics. In: Levy flights and related topics in Physics, vol. 450 (1995)
50. Ouaarab, A., Ahiod, B., Yang, X.S.: Discrete cuckoo search algorithm for the travelling salesman problem. Neural Comput. Appl. pp. 1–11 (2013)
51. Ouaarab, A., Ahiod, B., Yang, X.S.: Random-key cuckoo search for the travelling salesman problem. Soft Comput. pp. 1–8 (2014)

Cuckoo Search and Bat Algorithm Applied to Training Feed-Forward Neural Networks

Milan Tuba, Adis Alihodzic and Nebojsa Bacanin

Abstract Training of feed-forward neural networks is a well-known and important hard optimization problem, frequently used for classification purpose. Swarm intelligence metaheuristics have been successfully used for such optimization problems. In this chapter we present how cuckoo search and bat algorithm, as well as the modified version of the bat algorithm, were adjusted and applied to the training of feed-forward neural networks. We used these three algorithms to search for the optimal synaptic weights of the neural network in order to minimize the function errors. The testing was done on four well-known benchmark classification problems. Since the number of neurons in hidden layers may strongly influence the performance of artificial neural networks, we considered several neural networks architectures for different number of neurons in the hidden layers. Results show that the performance of the cuckoo search and bat algorithms is comparable to other state-of-the-art nondeterministic optimization algorithms, with some advantage of the cuckoo search. However, modified bat algorithm outperformed all other algorithms which shows great potential of this recent swarm intelligence algorithm.

Keywords Cuckoo search · Bat algorithm · Training neural networks · Swarm intelligence · Nature-inspired algorithms · Metaheuristic optimization

M. Tuba (✉) · N. Bacanin
Megatrend University Belgrade, Bulevar Umetnosti 29, 11070 N. Belgrade, Serbia
e-mail: tuba@ieee.org

N. Bacanin
e-mail: nbacanin@megatrend.edu.rs

A. Alihodzic
University of Sarajevo, Studentski trg 16, 71000 Sarajevo, Bosnia and Herzegovina
e-mail: adis_mtkn@yahoo.com

X.-S. Yang (ed.), *Recent Advances in Swarm Intelligence and Evolutionary Computation*,
Studies in Computational Intelligence 585, DOI 10.1007/978-3-319-13826-8_8

1 Introduction

Artificial neural network (ANN) is a computational model inspired by central nervous system which is capable of machine learning and pattern recognition. Like other machine learning methods, neural networks have been used for different optimization and mathematical problems such as clustering [1], text categorization [2], pattern recognition [3], weather forecasting [4], bankruptcy forecasting [5] etc. A neural network consists of two types of components: processing elements (neurons) and connections (links) between them. Every link has a weight parameter associated with it. The neural network is generally presented as a system of interconnected artificial neurons which can compute values from their inputs. Namely, each neuron receives signals from the neighboring neurons connected to it, processes the information, and produces an output. The neurons of neural networks can process input data in different ways and also can have various interconnections. Different structures of the artificial neural networks can be made by using different processing elements and by the specific manner in which they are connected. Artificial neural networks play an important role as classifiers in classification of non-separable data.

The design of ANN was motivated by the structure of a real brain, but the processing elements and the architectures used in ANN have gone far from their biological inspiration. Finding an optimal neural network structure as well as weight values presents a difficult optimization problem. Among many different ANN models, the multilayer feed-forward neural network (MLFF) is one which consists of neurons, that are ordered into layers where the first layer is called the input layer, the last layer is the output layer, and the layers between are hidden layers. These artificial networks have been mainly used due to their well-known universal approximation capabilities [6]. Successful application of the MLFF ANN to any problem will depend on the choice of structures used in training as well as the training algorithms.

A well-known algorithm named back-propagation (BP) algorithm is used for optimal network performance for training the MLFF ANN. BP Algorithm is a gradient descent optimization method that searches the synaptic weight coefficients of the MLFF ANN and minimizes the calculated gradient of the error with respect to the weight coefficients for a given input by propagating error backwards through the network. Since the error function of MLFF ANN is a multi-modal function, this method, although best in its class, has two important drawbacks: slow convergence speed and getting stuck into local minima easily.

The Levenberg-Marquardt algorithm (LMA) [7] combines the advantages of gradient-descent method and Gauss-Newton method. It is a very popular algorithm, but it finds only a local optimum. Therefore, many global optimization methods have been proposed to overcome the weakness of gradient-based techniques. Swarm intelligence algorithms like ant colony optimization [8–11], artificial bee colony [12–15], seeker optimization algorithm [16, 17], firefly algorithm [18–23] etc. have been successfully used for such hard optimization problems. Some nature inspired metaheuristics like genetic algorithms (GA) [24], particle swarm optimization (PSO)

[25], differential evolution (DE) [26], artificial bee colony (ABC) [27, 28] and firefly algorithm [29] were used to train the MLFF networks as an alternative to BP algorithm.

In order to overcome the disadvantages of the conventional BP, this chapter proposes two relatively new meta-heuristic search algorithms, called cuckoo search (CS) algorithm [30–34] and bat algorithm (BA) [35–37]. The CS, BA, and modified BA results obtained by using sigmoid transfer function are compared with the results obtained by Levenberg-Marquardt algorithm (LMA), artificial bee colony (ABC), and genetic algorithm (GA) from the paper [27], as well as with the results reported by the firefly algorithm [29]. The results in the papers [27] and [29] are also obtained by using sigmoid transfer function for the same sample data. Since the choice of number hidden neurons affects on the performance of neural network [38], we will perform the comparison of the results provided by the CS, BA and MBA for different number of neurons in the hidden layers.

Cuckoo search (CS) is a novel swarm intelligence metaheuristic algorithm introduced by Yang and Deb [32] which imitates animal behavior and is useful for global optimization [39]. Different approaches based on the CS algorithm were successfully applied to solve various optimization problems, such as engineering optimization problems [40], Knapsack problems [41], image thresholding [31] etc. Also, different versions of the implementation of the cuckoo search algorithm were provided [42].

Bat algorithm (BA) is also a novel swarm intelligence metaheuristic algorithm introduced by Yang [35] based on so-called echolocation of the bats. The primary purpose for bat's echolocation is to serve as a hunting strategy. A comprehensive review about swarm intelligence involving the bat algorithm is performed by Zhang and Wang [43] and Huang and Zhao [44]. Furthermore, Tsai proposed an evolving bat algorithm to improve the performance of standard BA with better efficiency [45]. Bat algorithm was successfully used for other problems [46, 47]. In this paper, its application to the training of MLFF networks is investigated.

Basically, the bat algorithm is powerful at intensification, but at times it may get trapped into some local optima, so that it may not perform diversification very well. In order to enhance the search performance of the bat algorithm, and also to avoid trapping into local optima, we introduce additional four modifications and tuning of the BA that almost always uniformly improve the results during the training of the neural networks. As a result, we propose a modified bat algorithm (MBA) which speeds up the global convergence rate of the BA while preserving its attractive characteristics.

Generally, metaheuristic search algorithms are often nature-inspired, and they are now the most practical approach for solving complicated optimization problems where traditional approaches are inefficient. The challenges of research in computational optimization are to find the right algorithms most suitable for a given practical problem to obtain acceptable solutions in a reasonable time with a limited amount of resources. In this chapter we present the use of metaheuristics for computational optimization and application to the classification problems, focusing

on bridging a significant gap between theory and the practical use of such algorithms in bio-inspired computing.

In this chapter, the main focus was on three things: the speed of convergence analysis of metaheuristics, the stability of used metaheuristics, as well as the classification accuracy obtained by mentioned metaheuristics. Namely, the speed of convergence is measured by counting the number of evaluations required for classification of a problem, the stability is measured by standard deviation, and the classification accuracy is expressed as the percentage of correct and false classification. In order to show the key points of the swarm intelligence based algorithms, in this chapter the following metaheuristics are discussed: Levenberg-Marquardt algorithm (LMA) search method, bat algorithm (BA), cuckoo search (CS) algorithm and modified bat algorithm (MBA).

The remainder of the chapter is organized as follows. Section 2 presents the structure of the artificial neural networks and introduces the Levenberg-Marquardt back-propagation algorithm. In Sects. 3 and 4 two recent population based metaheuristic algorithms are shown: cuckoo search algorithm (CS) and bat algorithm (BA). In Sect. 5, our proposed modified bat algorithm (MBA) is presented. In Sect. 6, the benchmark problems considered in this paper are described, as well as the discussion and analysis of the obtained experimental results. Finally, our conclusions are presented in Sect. 7.

2 Artificial Neural Networks

The basic MLFF neural network generates its output by transforming input data. In this work, we used MLFF neural network (see Fig. 1) that has three layers known as input, hidden and output layers, respectively. Each of the layers, input, hidden or output, consists of a certain number of nodes or neurons that are interconnected by links, so each processing node calculates a weighted sum of the nodes in the preceding layer to which it is connected. This weighted sum further transformed through the transfer or activation function to calculate output which is fed to the neurons in the next layer. Each link has a scalar weight associated with it that is adjusted during the training process.

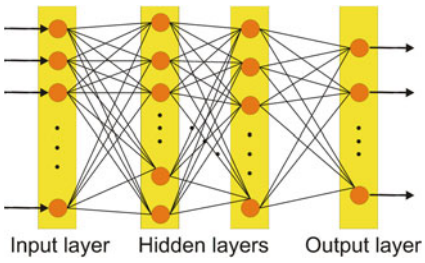


Fig. 1 Typical structure of a multi-layer feed-forward neural network

In the MLFF neural network, training data during the learning time move forward in only one direction from the input nodes, through several hidden nodes (if any) and to the output nodes. The output of the i-th node is calculated as

$$O_i = f_l\left(\sum_{i=1}^{n} w_{i,j} I_j + b_i\right), \tag{1}$$

where I_j is the input to the node j, b_i is the bias levels of a neural network, $w_{i,j}$ is the synaptic weight for the connection linking node i to j, and f_l is the transfer (activation) function which generates the output from nodes. The transfer function is usually a nonlinear function such as a sigmoid function [6], and is defined as

$$f_l = \frac{1}{1 + \mathrm{e}^{-x}}. \tag{2}$$

Input-output samples or training data T_s that are used during the training process of a neural network can be defined as

$$T_s = \{(x_s, d_s) : x_s \in R^n, d_s \in R^m\}, s = 1, \ldots, p \tag{3}$$

where x_s is the input sample, d_s is the desired sample, and p is the total number of input-output samples.

Given training data T_s, the task of evolutionary algorithms is to during the learning of the MLFF neural network, compute its free parameters so that the actual output due to x_s is close enough to d_s for all s in statistical sense. In this sense, in order to minimize the learning error function, the mean-square error (MSE) is used as the cost function, which is defined by

$$MSE = \frac{1}{p \cdot m} \sum_{s=1}^{p} \sum_{i=1}^{m} (d_{s,i} - O_{s,i})^2, \tag{4}$$

where m denotes the number of MLFF ANN outputs, and $O_{s,i}$ is the actual output value, both for the output i and the input sample s.

In nature inspired algorithms, operators are used to change the population, i.e., the vector of connection weights and the learning error obtained by the MLFF neural networks is used as a cost function which guides the selection.

The Levenberg-Marquardt algorithm (LMA) [7] combines the advantages of gradient-descent method and Gauss-Newton method, where backpropagation is a steepest descent algorithm, i.e., first-order gradient-based method and Gauss-Newton method [48] is a second-order curvature-based method. LMA starts with initial guess w_0, and w is adjusted by δ only for downhill steps:

$$\delta = (J^T(w)J(w) + \lambda I)^{-1} J^T(w) r(w), \tag{5}$$

where $J(w)$ is the Jacobian matrix of the residual vector $r(w)$ evaluated in w, λ is damping parameter and I is the identity matrix. The details of the Levenberg-Marquardt algorithm are:

1. *Initialization*. Set the generation counter $t = 1$; Initialize values for the parameter w, the LMA parameter λ as well as λ_{up} and λ_{down} to be used to adjust the damping parameter. Evaluate the residuals r and the Jacobian J at the initial parameter guess.
2. **Repeat**. Calculate the matrix $G = J^T J + \lambda I$, and the cost gradient $C = J^T r$, $\nabla C = 0.5r^2$.
3. *Evaluation*. Evaluate the new residuals r_{new} at the point given by $w_{new} = w - g^{-1}\nabla C$, and calculate the cost at the new point, $C_{new} = 0.5r_{new}^2$.
4. **If** $(C_{new} < C)$ **then** accept the step $w = w_{new}$, and set $r = r_{new}$ and $\lambda = \frac{\lambda}{\lambda_{down}}$. **Otherwise**, reject the step, keep the old parameter guess w and the old residuals r, and set $\lambda = \lambda \cdot \lambda_{up}$.
5. *Check for convergence*. If the method has converged, return w as the best-fit parameters. If the method has not yet converged but the step was accepted, evaluate the Jacobian J at the new parameter values. Then go to Step 2.

3 Cuckoo Search Algorithm

Cuckoo search (CS), swarm intelligence algorithm introduced by Yang and Deb [32], is based on the brood parasitism of some cuckoo species that lay their eggs in the nests of other host birds. Furthermore, the CS algorithm is enhanced by the so-called Levy flight used instead of simple isotropic random walk. It was successfully applied to a number of very different problems like structural engineering optimization [33], test effort estimation for testing the software [49], planar graph coloring problem [34], etc. The latest survey is [50].

In the CS algorithms, a nest represents a solution, while a cuckoo egg represents a new solution. The goal is to use the new and potentially better solutions to replace worse solutions in the population. In order to implement the CS algorithm, the following three idealized rules are used:

- Each cuckoo can lay only one egg at a time and choose random nest for laying their eggs;
- Greedy selection process is applied, so only the eggs with highest quality are passed to the next generation;
- Available host nests number is fixed. Host bird discovers cuckoo egg with probability p_a from [0, 1]. If cuckoo egg is discovered by the host, the host bird can either throw the egg away or leave the nest, and create a completely new nest.

This last assumption can be approximated by the fraction p_a of the n nests that are replaced with new random solutions. In the basic form of the CS algorithm, each nest contains one egg. This algorithm can be extended to more complicated cases in which each nest contains multiple eggs, i.e. set of solutions. In this approach, as well as the BA approach, the CS algorithm tries to find the optimal k-dimensional vector which will minimize the objective function [in our case defined by Eq. (4)]. Based on above approximations and idealization, the basic steps of the cuckoo search algorithm can be described as follows:

1. *Initialization*. Initialize the population of n nests randomly and each nest corresponding to a potential solution to the required problem. Calculate objective function values of all solutions and set the generation counter *cycle* to one.
2. *Calculate new population*. Detects the most successful solution as the best vector of weights. Calculate the step scale factor as:

$$\varphi = \left(\frac{\Gamma(1+\beta) \cdot \sin(\pi \cdot \frac{\beta}{2})}{\Gamma(1+\frac{\beta}{2}) \cdot \beta \cdot 2^{\frac{\beta-1}{2}}} \right)^{\frac{1}{\beta}}, \tag{6}$$

   where β denotes Levy distribution parameter, and γ denotes gamma function. For this step size, recommended value for $\beta = 1.5$ is used. Then each vector of the weights coefficients $\mathbf{t}$ from the search population produces a new solution $\mathbf{v}$ and tests its the objective function value. The new solution $\mathbf{v}$ is defined by:

$$\mathbf{v} = \mathbf{t} + s_k \cdot \left(\frac{r_1 \cdot \varphi}{r_2} \right)^{\frac{1}{\beta}} \cdot (\mathbf{t} - \mathbf{t}_{best}) \cdot r_3, \tag{7}$$

   where r_1, r_2 and r_3 are three normally distributed random numbers, and s_k are the scaling factors. At each computation step, check the boundary conditions of the created new solution. If the objective function value of the new one $\mathbf{v}$ is higher than that of the previous one $\mathbf{t}$, memorize $\mathbf{v}$ and forget the old one. Otherwise, keep the old solution.
3. *Record the best solution*. Memorize the best solution vector $\mathbf{t}_{best}$ with the optimal objective fitness value.
4. *Fraction p_a of worse nests are abandoned and new nests are being built*. For each solution $\mathbf{t}$ apply the crossover operator in the search population by:

$$\mathbf{v} = \begin{cases} \mathbf{t} + rand_1 \cdot (\mathbf{t}_{perm1} - \mathbf{t}_{perm2}), & \text{if } rand_2 < p_a, \\ \mathbf{t}, & \text{otherwise}, \end{cases} \tag{8}$$

where $rand_1$ and $rand_2$ are uniform random numbers in range [0, 1], and $perm_1$ and $perm_2$ are different rows permutation functions applied to nests matrix.

5. *Record the best solution*. Memorize the best solution of the weight coefficients $\mathbf{t}_{best}$ so far, and increase the variable cycle by one.
6. *Check the termination criterion*. Until the termination criteria is satisfied or *cycle* > *MaxGeneration*.

4 Bat Algorithm

Bat algorithm (BA) is a new population based metaheuristic approach proposed by Yang [35] with the latest review in [51]. The algorithm uses the so-called echolocation of the bats. Echolocation is typical sonar which bats use to detect prey and to avoid obstacles. From physics, it is known that pulses reflect from barriers. The bats move by using the time delay between emission and reflection. In order to transform these behaviors of bats to algorithm, Yang used three generalized rules:

- All bats use echolocation to sense distance, and they also know the surroundings in some magical way;
- Bats fly randomly with velocity v_i at position x_i with a fixed frequency f_{min}, varying wavelength λ and loudness A_0 to search for prey. Bats are able to automatically set the wavelength of their transmitted pulses and tune the rate of pulse emission r from [0, 1], depending on the proximity of their target;
- Since the loudness of bats can be changed in several ways, it is supposed that the loudness changes from a positive large value A_0 to a minimum constant value A_{min}.

Based on these approximations and idealization, the basic steps of the bat algorithm can be described as follows:

1. *Initialization*. Set the generation counter $t = 1$; Initialize the population of n bats randomly and each bat corresponding to a potential solution to the given problem; Define loudness A_i, pulse frequency Q_i and the initial velocities v_i $(i = 1, 2, \ldots, N)$; Set pulse rate r_i.
2. *Repeat*. Based on the updated frequency and velocity of the movement of bats, new solutions are created (Eqs. 9–11);

   if ($rand > r_i$) **then**
   Select a solution among the best solutions;
   Generate a local solution around the selected best solution Eq. (12);
   end if
   Make a new solution by flying randomly;
   if ($rand < A_i$ and $f(x_i) < f(x^*)$) **then**
   Accept the new solutions;
   Increase r_i and reduce A_i (Eqs. 13, 14);
   end if
   Rank the bats and find the current best x^*;
   $t = t + 1$

3. *Check the termination criterion*. Until the termination criteria is satisfied or $t > MaxGeneration$.
4. *Finish.* Show the results of processing and make visualization.

In bat algorithm, initialization of the bat population is performed randomly and each bat is defined by its locations x_i^t, velocity v_i^t, frequency f_i^t, loudness A_i^t, and the emission pulse rate r_i^t in a search space of synaptic weights. The new solutions x_i^t are performed by moving virtual bats according to the following equations:

$$f_i = f_{min} + (f_{max} - f_{min}) \cdot \beta, \tag{9}$$

$$v_i^t = v_i^{t-1} + (x_i^{t-1} - x_*) \cdot f_i, \tag{10}$$

$$x_i^t = x_i^{t-1} + v_i^t, \tag{11}$$

where β from the closed interval [0, 1] is a random vector drawn from a uniform distribution. Here x_* is the current global best vector of synaptic weights which is located after comparing all the weights among all the bats. Initially, each bat is randomly assigned a frequency which is drawn uniformly from the interval [f_{min}, f_{max}]. A random walk with direct exploitation is used for the local search that modifies the current best vector of weights according the equation:

$$x_{new} = x_* + \varepsilon \cdot s_k, \tag{12}$$

where ε from the interval [−1, 1] is a is a random number, $s_k = |u_k - l_k|$ are the scaling parameters, u_k is upper bounds and l_k is lower bounds. We can see from the pseudo code of the BA that the intensive local search is controlled by the loudness and pulse rate. As the loudness usually decreases once a bat has found its pray, while the rate of pulse emission increases, the loudness can be chosen as any value of convenience. Mathematically, these characteristics are defined with the following equations:

$$A_i^{t+1} = \alpha \cdot A_i^t, \tag{13}$$

$$r_i^{t+1} = r_i^0 \cdot (1 - e^{-\gamma t}), \tag{14}$$

where α and γ are constants. In this approach, the BA tries to find the optimal k-dimensional vector which will minimize the objective function defined by Eq. (4), whereby k denotes the total number of synaptic weights.

In this chapter, we will introduce some modifications to the basic bat algorithm for training neural networks, so as to further improve its global search performance. For some benchmark problems, acceptable results are obtained by the bat algorithm. However, based on both our experiments and the results published in the papers [46, 47, 52, 53], we noticed that bat algorithm for some problems has somewhat weaker diversification, compared to good intensification. Hence, after some time, it may be stuck into some local optimum for some class of problems, for example

XOR 2-2-1 without bias. Therefore, we propose modified version of bat algorithm (MBA) adopted to more efficient search for synaptic weights during the training neural networks. As we will see in Sect. 5, this modification will be based on the principles of the artificial bee colony algorithm (ABC) and partly on differential evolution (DE) algorithm. We compared our proposed algorithm with the LMA, GA, ABC, FA, CS and BA algorithms, and demonstrated that it always produces better results considering both accuracy and, especially, convergence speed.

5 Modified Bat Algorithm

In order to enhance the search performance of the BA and also to avoid trapping into local optima, we introduce four modifications to the basic bat algorithm. As a result, we propose a modified bat algorithm (MBA) which not only inherits the simplicity and efficiency of the basic BA with a capability of searching for global minimum, but also speeds-up the global convergence rate of the BA while preserving its attractive characteristics.

In the pure bat algorithm [35] exploration and exploitation are controlled by functions in Eqs. (13, 14). Analyzing these functions, we noticed that the algorithm for some problems loses exploration capability as iterations proceed. In order to avoid this problem and establish a good balance between intensification and diversification of BA, we changed the form of pulse rate function (Eq. 13), because it allows to switch from the exploration to exploitation and vice versa. The reason for the introduction of this first modification is that when the loudness usually starts to decrease, while at the same time the rate of pulse emission starts to increase, then the bats start to best approximate their optimum solutions.

The first modification is to modify the pulse emission rate vector $\mathbf{r} = r_i^t$ by

$$r_i^t = r_{max} - r_{min} + \lambda - 10^{\frac{t \cdot (r_{max} - r_{min})}{t_{max}}}, \tag{15}$$

where r_{max} and r_{min} from interval [0.5, 1.0) are maximum and minimum pulse rate emission, respectively, λ is a constant factor from the interval [0, 0.5), t_{max} is the maximal number of generations. It was empirically determined that the best results are obtained for maximum and minimum pulse rates $r_{max} = 0.95$, $r_{min} = 0.7$, $\lambda = 0.3$, initial loudness $A_0 = 0.95$ and $\alpha = 0.9$.

Therefore, thanks to the new form of pulse rate function, by appropriately tunned parameters such as r_{max}, r_{min} and λ, in the early stages of the bat algorithm, a large part of agents (bats) will be redirected to diversification, and the remaining will be responsible for intensification. On the other hand, as iterations progress, some of agents will reorient from the diversification to intensification, hence enabling more exhaustive search in the beginning and more targeted towards end.

In the basic bat algorithm [35], the bats randomly select a range of frequency f from Eq. (9) and based on that frequency, they adjust their own solutions

(positions). It is clear that these frequencies will have the same effect to all dimensions of some solution. Since the increasing or decreasing of the frequency controls the rate of bats moving, the old frequencies from Eq. (9) will reduce the local search performance of the basic bat algorithm. Therefore, in this paper we propose the new equation for the frequency modification as the second modification.

The second modification is to change the frequency f from Eq. (9) according to the Eq. (15), which is given by

$$f_i^t = \begin{cases} 1 - max(f_{min}, min(f_{max}, r_i^t)) \cdot \beta, & \text{if } r_i^t < \frac{|\beta|}{2}, \\ max(f_{min}, min(f_{max}, r_i^t)) \cdot \beta, & \text{otherwise,} \end{cases} \tag{16}$$

where f_{max} and f_{min} from interval [0, 1) are maximum and minimum frequency, and β is a random number generated from normal distribution with zero mean value, and standard deviation which is equal one.

Since the new position of individual bat is generated using its velocity and its own position, in order to avoid premature convergence, we change the old Eq. (10) for the velocity and we propose the new equation as a third modification. The main idea is that the velocity of an individual bat is updated by adjusting its pulse rate emission and frequency by equations Eqs. (15, 16), respectively.

The third modification relates to the modification of the equation for the velocity of an individual bat, and that velocity modification is represented by:

$$v_{i,j}^t = \begin{cases} 0, & \text{if } \gamma > C_r, \\ (x_{i,j}^t - x_{a,j}^t) \cdot f_i^t, & \text{otherwise,} \end{cases} \tag{17}$$

where C_r is a crossover probability from the interval [0, 1), γ and a are the uniformly distributed random numbers from the interval [0, 1) and $a \neq i$, and j is a uniformly distributed random number from the interval [0, d), and d denotes the dimension of a considered problem.

As we can see from the Eq. (17), this equation combines one part of the equation from the ABC algorithm that is used in the procedure *employed bee phase* and adapted version of the differential operator that is similar to the *mutation strategy DE/best/1* in the differential algorithm (DE). Through testing benchmarks in the following sections, it was found that setting parameter of the crossover probability C_r to 0.85 produced the best results.

Although the above modification can improve the performance of the bat algorithm over most of the benchmark problems, several solutions in the XOR 2-2-1 without bias problem may remain stuck in some local minimum. In order to fix this lack of the former modifications, we introduced the fourth modification, which is inspired by launch of the scouts in the *scout phase* of the ABC algorithm.

The fourth modification provides that when some solution gets trapped into a local optima after a certain number of iterations, it will eventually exceed the predetermined number of allowed trials called "*limit*". When a solution unchanged exceeds the *limit* trials, it is redirected to search new space by using the random walk.

6 Experimental Results

In order to estimate the performance of the cuckoo search algorithm, bat algorithm and modified bat algorithm applied to the training of the MLFF neural networks and also to compare with other state-of-the-art algorithms applied to the same problem, four well-known benchmark problems were used. The details of these problems are described in the next subsection.

All training algorithms have been implemented in C# programming language. All tests were done on an Intel Core i7 3770 K, 3.5 GHz with 16 GB of RAM and Windows 8 × 64 Professional operating system. In the experiments, for each benchmark problem experiments were repeated 30 times and training processes were stopped when the mean squared error (MSE) of the outputs associated with inputs was equal to or less than 0.01 or when the *maximum cycle* number has been reached.

6.1 Benchmark Problems

1. **The Exclusive-OR (XOR) Problem** is the first problem used in the experiments. It has a long history in the study of neural networks [54]. Boolean function using for this classification problem maps two binary inputs to a single binary output. **Input-output samples** for this problem are: (0 0; 0 1; 1 0; 1 1) → (0; 1; 1; 0). In the simulations, we used different models of the MLFF neural networks with and without biases as: 2-2-1; 2-3-1; 2-4-1; 2-5-1, 2-10-1.
2. **3-Bit Parity Problem** is the second problem taken into account in the experiments. This problem is a special case of *N*-Bit Parity Problem [49]. Also, the XOR problem is a special case of *N*-Bit Parity Problem for $N = 2$. The *N*-bit parity function maps the set of distinct binary vectors in the set {0, 1}. It is the indicator function that returns 1, when the number of binary units is odd, else it returns 0. **Input-output samples** for 3-Bit Parity problem are: (0 0 0; 0 0 1; 0 1 0; 0 1 1; 1 0 0; 1 0 1; 1 1 0; 1 1 1) → (0; 1; 1; 0; 1; 0; 0; 1). In the experiments, we used the following versions with biases of the MLFF neural networks: 3-3-1, 3-5-1, 3-10-1, 3-20-1, 3-30-1.
3. **4-Bit Encoder/Decoder Problem** is the third problem used in the experiments. This problem has four distinct input samples, each of them has only one bit turned on. This is quite close to real world pattern classification problems, where small changes in the input samples cause small changes in the output samples. **Input-output samples** for this type of problem are: $(0001; 0010; 0100; 1000) \rightarrow (0001; 0010; 0100; 1000)$. In the simulations, we used the following models with biases of the MLFF neural networks: 4-2-4, 4-4-4, 4-5-4, 4-10-4, 4-20-4.
4. **Iris classification problem** is the fourth problem used in the experiments. This problem is used as a benchmark widely in the artificial neural networks field.

It is 4 dimensional pattern classification problem with three classes: *Setosa*, *Versicolor*, and *Virginica*. This problem has 150 input-output samples and can be downloaded from [55]. Each input sample consists of four attributes that can be described as categorical, nominal and continuous. Namely, these attributes are named with: *sepal length*, *sepal width*, *petal length*, and *petal width*. It is realized that the petal width is always smaller than petal length and sepal width is also smaller than sepal length. In the experiments, we used the following models with biases of the MLFF neural networks: 4-5-3, 4-10-3, 4-15-3, and compared the performance for different number of hidden neurons.

In all experiments, for each model of the MLFF neural network, the parameter ranges for all problems are [−100, 100] and [−10, 10], respectively. The total number of weight coefficients with and without bias in the MLFF neural network is defined as $(1 + n_i) \cdot n_h + (1 + n_h) \cdot n_o$ and $(n_i + n_o) \cdot n_h$, respectively. The total number of biases is defined as $n_h + n_o$, where n_i, n_h and n_o denote the total number of neurons in the input, hidden and output layers, respectively.

6.2 Parameter Settings

Since each benchmark problem has its own difficulty, different parameter setups were used for each of them. The MLFF neural network architecture 2-2-1 without biases for XOR benchmark problem is the most difficult problem so that all algorithms were run through more generations for this type of problem. We have the following adjustments of parameters for the following algorithms (GA, ABC and FA are from [29]):

- **GA setup**—In the experiments, the parameters value are the following: single point crossover with the rate of 0.8, uniform mutation with the rate of 0.05 are employed. Generation gap is set to 0.9. The population size in GA is 50 for all problems;
- **ABC setup**—The parameter values utilized by ABC in all the experiments are the following: the value of *limit* is equal to *SN***D*, where *D* is the dimension of the problem, and colony size (2**SN*) is 50 for all problems.
- **FA setup**—In the FA algorithm, the parameter values in the all experiments are: size of firefly population is 50, the value of parameter γ is set to 1.0, the initial values of the attractiveness are set to 1.0, the initial value of paremeter α is 0.9, i.e. the reduction scheme was followed by reducing from 0.9.
- **BA setup**—Parameter values utilized by BA in the all experiments are: size of bat population is 50, the initial values of the pulse rates are set to 0.9, the initial values of the loudness are set to 0.95, minimum and maximum frequencies f_{min} and f_{max} are set to 0 and 2 as well as the constants α and γ are set to 0.9 and 0.99, respectively.
- **CS setup**—In the CS algorithm, the probability of discovering a cuckoo egg p_a is set to 0.5. Although the value 0.25 of parameter p_a was suitable for the most

applications, we have found that the best performance of the neural networks is achieved for this choice of the parameter value. The population size is 50 for all benchmark test problems.

- **MBA setup**—In our proposed MBA approach, the parameter values for the all experiments are: size of bat population is 50, the initial values of the pulse emission rates are set to $r_{min} = 0.70$ and $r_{max} = 0.95$, respectively, the initial values of the loudness are set to 0.95, the crossover probability C_r is set to 0.85, minimum and maximum frequencies f_{min} and f_{max} are set to 0.2 and 0.8, respectively, as well as the constants α, λ, and γ are set to 0.9, 0.3 and 0.99, respectively. The parameter *limit* in the fourth modification of the MBA is set to 150.

The maximum number of evaluations for the following architecture of neural networks: XOR without bias (2-2-1), XOR with bias (2-2-1), XOR with bias (2-3-1), 3-Bit Parity problem with bias (3-3-1), 4-Bit Encoder-Decoder problem with bias (4-2-4) and Iris classification problem are set to 375,000, 5,000, 3,750, 50,000, 50,000, and 25,000 respectively.

6.3 Discussion and Analysis

Comparative results for the LMA, GA, ABC, FA, BA, CS and MBA obtained by using sigmoid activation function are given in Table 1 (results for GA, ABC and FA are from [29]). The best obtained results in Table 1 are highlighted in the bold type.

We can see that the LMA gives the worst results in comparison with the remaining algorithms for almost all test problems, so it will not further continue to participate in the analysis and comparison with other algorithms.

When comparing the GA with the remaining algorithms not taking into account the LMA, from the results in Table 1, we can conclude that the algorithms ABC, BA, CS and MBA produce better results with respect to all statistical parameters as MMSE, SDMSE, MC and SR. Comparing it with the FA, we can notice from Table 1 that both GA and FA show similar performances for problems XOR 2-2-1 without bias and Enc.-Dec. 4-2-4 problem. For the remaining two test problems, FA obtained better results, especially for the XOR 2-2-1 with bias. Also, from the mean of cycle numbers, it can be seen that FA converges significantly faster than GA.

Since the ABC produced better results compared to the LMA and GA, it remains to see how the ABC behaves in the comparison with remaining algorithms. When comparing ABC with respect to FA, we can see that ABC performs better on problems XOR 2-2-1 without bias, 3-Bit Parity problem and Enc.-Dec. problem. For the XOR 2-2-1 problem with bias, from the MMSE and MC, it can be seen that FA performs better than ABC, i.e. it converges more than three times faster to the global minimum compared to ABC for this benchmark problem. In the comparison of ABC and BA algorithms, we can see that the ABC outperformed the BA for most of the benchmark problems. For Enc.-Dec. problem, it can be seen that the BA has better MMSE and SDMSE than the ABC. When it comes to the CS algorithm,

Table 1 Results for 30 independent runs of the training MLFF ANN produced by LMA, GA, ABC, FA, BA, CS and MBA using sigmoid transfer function

Algor.	Statistics	Benchmark classification problems			
		XOR	XOR	3 Bit. parity	Enc. Dec.
		MLFF neural network architecture			
		2-2-1 without bias	2-2-1	3-3-1	4-2-4
LMA	MMSE	0.1107	0.0491	0.0209	0.0243
	SDMSE	0.0637	0.0646	0.0430	0.0424
GA	MMSE	0.099375	0.047968	0.028725	0.016400
	SDMSE	0.02785	0.052000	0.032900	0.031300
	MC	7500	77.067	501.1333	400.1333
	SR	0	40	63.3333	86.6667
ABC	MMSE	0.007051	0.006956	0.006679	0.008191
	SDMSE	0.002305	0.002402	0.002820	0.001864
	MC	2717.4	32	179.07	185
	SR	100	100	100	100
FA	MMSE	0.078939	0.004971	0.015399	0.017473
	SDMSE	0.018168	0.002677	0.023297	0.022646
	MC	7500	**9.3**	181.63	216.03
	SR	0	100	83.3	80
BA	MMSE	0.050402	0.006275	0.006924	0.007601
	SDMSE	0.028339	0.003448	0.002194	0.001825
	MC	6678.5	60	325.1	300.1667
	SR	26.66667	86.6667	100	100
CS	MMSE	0.005974	0.005913	0.006565	0.007285
	SDMSE	0.003179	0.002351	0.002120	0.001808
	MC	1519.333	30.96667	212.1	219.86667
	SR	100	100	100	100
MBA	MMSE	**0.005060**	**0.004656**	**0.005432**	**0.006864**
	SDMSE	**0.003147**	**0.002873**	**0.002986**	**0.002338**
	MC	**946.63**	26.233	**137.4**	**184.37**
	SR	**100**	**100**	**100**	**100**

MMSE mean of mean squared errors, *SDMSE* standard deviation of mean squared errors, *MC* mean of cycle numbers, *SR* success rate

then both the ABC and CS algorithms show similar performances for all benchmark problems. Namely, the ABC gives a slightly better results with respect to the MC for the third and fourth benchmark problem, and worse results with respect to the MMSE for all benchmark problems. The CS converges somewhat faster than the ABC for the problem XOR 2-2-1 with and without bias, and it converges slower than the ABC for the remaining problems. From Table 1, it can clearly be shown that the ABC algorithm was completely outperformed by the modified bat algorithm (MBA) for all benchmark problems as well as for all statistical parameters: MMSE, SDMSE, MC and SR.

Since the performance of the FA has already been compared with the performances of ABC, GA and LMA algorithms, now it remains that the results obtained by the FA are compared to the results of the remaining algorithms. Firstly, as we can see from Table 1, the FA in the comparison to the BA produces the worse results with respect the statistical parameter success rate (SR). Namely, the BA has twice reached a 100 % SR, unlike the FA, which succeeded in only one. On other hand, for the XOR 2-2-1 problem with bias, 3 Bit. Parity problem and Enc.-Dec. problem,the FA is superior compared with BA with respect to the MC, and inferior with respect to the MMSE for all benchmark problems except for the XOR 2-2-1 problem with bias. When comparing the FA with the CS, we can see that the CS produces better results than the FA for all statistical parameters except for the MC. The MBA outperforms the FA for all statistical parameters, except for the XOR 2-2-1 problem, where the FA shows the higher rate of the convergence.

From Table 1, it can been seen that the BA is inferior in the comparison with the CS and MBA for all statistical parameters.

As we can see from Table 1, our proposed method (MBA) has the best results compared to the all algorithms for the all test problems. It has the fastest rate of convergence and also has the best MMSE values for all benchmark problems.

Our proposed algorithm (MBA) as well as both the CS and ABC have 100 % success rate for all test problems. We can see that the CS gives better mean square error compared to the ABC for all test problems, while the ABC gives slightly better results compared to the BA for almost all test problems, except for the XOR 2-2-1 with bias and Encoder-Decoder problem.

Since the number of hidden neurons affects the performance of the neural network, the MLFF neural network is trained for different number of hidden neurons considering the XOR, 3-Bit Parity and Encoder-Decoder problems.

The results obtained by the BA, CS and MBA algorithms applied to the XOR problem are shown in Table 2. The maximum numbers of evaluations for all architectures 2-3-1, 2-4-1, 2-5-1 and 2-10-1 are set to 5,000. From Table 2, we can see that all algorithms had a 100 % success rate for all architectures. The CS outperformed the BA with respect to average mean square error, while our proposed method MBA in comparison with the other two algorithms, CS and BA, provides the best results for AMSE as well as for ANE. Also, our proposed method MBA tends to faster convergence when the number of hidden neurons is increased from 3 to 10.

It is shown in Table 3 that for 3-Bit Parity problem all algorithms achieved 100 % accuracy rate and that the MBA has the best average mean square error (AMSE) for all architectures of neural networks. The remaining algorithms also show good performance with respect the mean number of evaluations and the standard deviation. The maximum number of evaluations for the 3–Bit parity and Encoder-Decoder benchmark problems that are discussed in Tables 3 and 4, respectively, is limited to 50,000.

In Table 4, for the Encoder-Decoder problem all three algorithms performed classification with the 100 % accuracy rate. As in the previous cases, the MBA has the best performance compared to other algorithms with respect to the average of mean square error and average number of evaluations (ANE). From the results

Table 2 Average of mean square error (AMSE), standard deviation of mean square error (SDMSE), average number of evaluations (ANE), and accuracy rate (AR) for all training samples over 30 independent runs for BA, CS and MBA using sigmoid transfer function for XOR problem

Algor.	Statistics	Benchmark classification XOR problem			
		MLFF neural network architecture with bias			
		2-3-1	2-4-1	2-5-1	2-10-1
BA	AMSE	0.005672	0.005417	0.004813	0.003937
	SDMSE	0.002758	0.002517	0.002948	0.002844
	ANE	1780.00	1106.50	966.50	430.00
	AR	100	100	100	100
CS	AMSE	0.005169	0.005142	0.004679	0.003480
	SDMSE	0.002829	0.002407	0.002761	0.002364
	ANE	1135.00	861.50	630.00	430.00
	AR	100	100	100	100
MBA	AMSE	**0.003782**	**0.003741**	**0.003157**	**0.002412**
	SDMSE	**0.002846**	**0.002791**	**0.002780**	**0.002310**
	ANE	**831.5**	**608.5**	**555**	**353.5**
	AR	**100**	**100**	**100**	**100**

Table 3 Average, standard deviation, average number of evaluations, and accuracy rate of MSE for all training samples over 30 independent runs for BA, CS and MBA using sigmoid transfer function for 3-bit parity problem

Algor.	Statistics	Benchmark classification 3 bit. parity problem			
		MLFF neural network architecture with bias			
		3-5-1	3-10-1	3-20-1	3-30-1
BA	AMSE	0.006724	0.006022	0.004842	0.004255
	SDMSE	0.002262	0.002525	0.003040	0.002710
	ANE	7986.50	6198.50	4328.50	4066.50
	AR	100	100	100	100
CS	AMSE	0.005548	0.005347	0.004615	0.003942
	SDMSE	0.002901	0.002712	0.002315	0.003042
	ANE	6263.50	4063.50	3191.50	2523.50
	AR	100	100	100	100
MBA	AMSE	**0.005122**	**0.004172**	**0.002913**	**0.002718**
	SDMSE	**0.003109**	**0.002901**	**0.003333**	**0.002335**
	ANE	**2538.5**	**1998.5**	**1638.5**	**1571.5**
	AR	**100**	**100**	**100**	**100**

shown in mentioned tables, we can see that with increasing the number of hidden neurons, we obtain a faster convergence rate and greater stability of the solutions.

Iris datasets from [55] are selected and used to train the MLFF neural network. Out of these 150 samples, 90 samples (30 samples were taken from each class) were used for training, 30 samples were used for validation (10 samples were taken from

Table 4 Average, standard deviation, average number of evaluations, and accuracy rate of the MSE for all training samples over 30 independent runs for BA, CS and MBA using sigmoid transfer function for encoder-decoder problem

Algor.	Statistics	Benchmark classification Enc. Dec. problem			
		MLFF neural network architecture with bias			
		4-4-4	4-5-4	4-10-4	4-20-4
BA	AMSE	0.006137	0.005994	0.005620	0.004964
	SDMSE	0.002708	0.002461	0.002554	0.003247
	ANE	7826.50	7060.00	6906.50	6218.50
	AR	100	100	100	100
CS	AMSE	0.006131	0.005951	0.005474	0.004634
	SDMSE	0.002488	0.002959	0.002491	0.003018
	ANE	5720.00	5671.50	4821.50	4461.50
	AR	100	100	100	100
MBA	AMSE	**0.005644**	**0.005589**	**0.004683**	**0.004358**
	SDMSE	**0.002928**	**0.002688**	**0.003046**	**0.003208**
	ANE	**5523.50**	**5293.50**	**4608.50**	**4357.00**
	AR	**100**	**100**	**100**	**100**

each class) and the remaining 30 samples to test the MLFF neural networks for each class separately. For each class, the first 30 samples are randomly chosen for the training process, the next 10 samples are randomly used for validation and the remaining 10 samples in the class remain to be tested. Since Iris classification problem was not used in [29] only algorithms that we implemented here were applied, namely BA, CS and MBA.

In order to prevent the over-training during training the neural networks, the validation of dataset is done. The results of training and testing neural networks to solve Iris classification problem are presented in Tables 5 and 6, respectively.

For this classification problem, we use MLFF neural network with architecture $4 \times HN \times 3$, where HN is the number of hidden nodes, and we compare the performance of neural networks with $HN = 5$, 10 and 15. Table 5 shows the best, mean and standard deviation of MSE, and the average number of evaluations obtained for each algorithm and for different number of hidden neurons (HN). Each of the 30 independent run stops when the mean square error (MSE) is less than tolerance value 0.01 or when the maximum number of evaluations (25,000) is reached.

In the case when the number of hidden neurons is 10 and 15, respectively, the results in Table 5 show that the MBA gives better results for the average mean square error, standard deviation of the mean square error and average number of evaluations compared to the BA and CS over 30 independent runs for almost all cases of hidden number of neurons. Also, in this case the MBA has better results for

Table 5 Best, average, standard deviation, average number of evaluations of MSE for all training samples over 30 runs for BA, CS and MBA using sigmoid transfer function for classification

HN	Algorithms	Benchmark Iris classification problem			
		MLFF neural network architecture 4 × HN × 3			
		Best of MSE	Average of MSE	Std. De. of MSE	Average num. of evaluations
5	BA	0.015980	0.029381	0.007464	15065.00
	CS	0.016108	0.024712	0.010707	10585.00
	MBA	**0.013546**	**0.023088**	**0.006785**	**7935**
10	BA	0.013077	0.028409	0.006048	18531.67
	CS	0.013082	0.031885	0.007562	9921.67
	MBA	**0.01208**	**0.024892**	**0.006985**	**5635**
15	BA	0.012390	0.030534	0.009083	21023.33
	CS	0.015094	0.026972	0.008258	12263.33
	MBA	**0.012133**	**0.026620**	**0.006345**	**5395**

the Best of MSE compared to the remaining algorithms. As can be inferred from Table 5, the best performance with respect to AMSE for the algorithm MBA are achieved when the number of hidden neurons is equal to 10. The best results are highlighted in bold type. Table 5 shows that MBA and CS have the best convergence rate for all values of the hidden neurons (*HN*).

From Table 6, we can see that with regard to the best classification rate, the algorithms CS, BA and MBA have a 100 % classification rate for all classes and show the best results for all architectures of MLFF neural networks. In this case, all samples belonging the classes *Setosa*, *Versicolor* and *Virginnica* are 100 % classified. With regard to the average classification, we can see from Table 7, that the MBA has the same classification rate as the CS for the architecture 4 × 10 × 3, and better classification rates for the remaining architectures of the neural networks. For architecture 4 × 5 × 3, the CS and BA produce the similar classification results, while for the other models of architecture; the CS gives classification results which are much better than those obtained by the BA. As we can see from 6, as well as for this classification problem, the MBA has a better classification rate compared to the CS and BA. Namely, in 500 generations, for all architectures of neural network, the MBA were classified 74 samples of the total of 75 samples, only one sample was not correctly classified, while the CS algorithm is able to classify two times 73 samples in the case of architectures 4 × 5 × 3 and 4 × 15 × 3, respectively, and once classified 74 samples in the case of architectures 4 × 10 × 3. Thus, CS algorithm is by efficiency immediately after the MBA algorithm, while the BA is by the efficiency in the third place, because only once it was able to classify 72 samples.

Table 6 Best, average, worst classifications for all testing samples over 30 independent runs for CS, BA and MBA using sigmoid transfer function

HN	Algorithms	IRIS plant classes	Test classification data for Iris classification problem for 500 generation		
			MLFF neural network architecture 4 × HN × 3		
			Best classification %	Average classification %	Worst classification %
5	BA	Setosa	100	100	99
		Vericolor	100	92	32
		Virginnica	100	97	79
	CS	Setosa	100	100	100
		Vericolor	100	96	82
		Virginnica	100	97	**87**
	MBA	Setosa	100	100	100
		Vericolor	100	**97**	82
		Virginnica	100	**99**	83
10	BA	Setosa	100	100	96
		Vericolor	100	85	30
		Virginnica	100	97	60
	CS	Setosa	100	100	93
		Vericolor	100	96	73
		Virginnica	100	99	92
	MBA	Setosa	100	100	**100**
		Vericolor	100	96	**82**
		Virginnica	100	99	**87**
15	BA	Setosa	100	100	94
		Vericolor	100	86	0
		Virginnica	100	87	0
	CS	Setosa	100	100	96
		Vericolor	100	92	0
		Virginnica	100	100	87
	MBA	Setosa	100	100	100
		Vericolor	100	**97**	**82**
		Virginnica	100	99	85

Table 7 Average classifications for all testing samples over 30 independent runs for CS, BA and MBA using sigmoid transfer function

HN	Algorithms	Benchmark Iris classification problem							
		MLFF neural network architecture 4 × HN × 3							
		Setosa		Versicolor		Virginnica		Total	
		Class.	Not cl.	Class.	Not cl.	Class.	Not cl.	Class.	Not cl.
5	BA	25	0	23	2	24	1	72	3
	CS	25	0	24	1	24	1	73	2
	MBA	**25**	**0**	**24**	**1**	**25**	**0**	**74**	**1**
10	BA	25	0	21	4	24	1	70	5
	CS	**25**	**0**	**24**	**1**	**25**	**0**	**74**	**1**
	MBA	**25**	**0**	**24**	**1**	**25**	**0**	**74**	**1**
15	BA	25	0	21	4	21	4	67	8
	CS	25	0	23	2	25	0	73	2
	MBA	**25**	**0**	**24**	**1**	**25**	**0**	**74**	**1**

7 Conclusion

Adjustment and application of the cuckoo search, bat algorithm and modified bat algorithm to training of the multilayer feed-forward artificial neural networks is described in this chapter. Four well-known benchmark problems: XOR problem, 3-bit Parity problem, Encoder-Decoder problem and Iris classification problem were used to evaluate the performance of these new learning algorithms using sigmoid activation function. The results are compared with the results reported by Levenberg-Marquardt algorithm, genetic algorithm, artificial bee colony and firefly algorithm. According to the experimental results for all benchmark problems, it can be concluded that cuckoo algorithm produces good results, while the results for pure bat algorithm are somewhat inferior for this class of problems. In order to improve the results obtained by the pure bat algorithm, we introduced four modifications and our proposed modified bat algorithm achieved the best results compared to all other compared algorithms by all statistical parameters. Here we demonstrated great potential of the cuckoo search and bat algorithms for training multilayer feed-forward neural networks by testing on three simple and one more complex standard benchmark problems. Future work may include large-scale benchmark problems, some real-life problems as well as additional modifications to the algorithms to further improve the performance.

Acknowledgments This reserach was supported by Ministry of Education and Science of Republic of Serbia, Grant III-44006.

References

1. Du, K.L.: Clustering: a neural network approach. Neural Netw. **23**(1), 89–107 (2010)
2. Sebastiani, F.: Machine learning in automated text categorization. ACM Comput. Surv. **34**(1), 1–47 (2002)
3. Kim, T.: Pattern recognition using artificial neural network: a review. Inf. Secur. Assur. Commun. Comput. Inf. Sci. **76**, 138–148 (2010)
4. Shrivastava, G., Karmakar, S., Kowar, M.K., Guhathakurta, P.: Application of artificial neural networks in weather forecasting: a comprehensive literature review. Int. J. Comput. Appl. **51** (18), 17–29 (2012)
5. Perez, M.: Artificial neural networks and bankruptcy forecasting: a state of the art. Neural Comput. Appl. **15**(2), 154–163 (2006)
6. Haykin, S.: Neural Networks and Learning Machines. Prentice Hall, New Jersey (2009)
7. Marquardt, D.W.: An algorithm for least squares estimation of non-linear parameters. J. Soc. Ind. Appl. Math. **11**(2), 431–441 (1963)
8. Jovanovic, R., Tuba, M.: Ant colony optimization algorithm with pheromone correction strategy for the minimum connected dominating set problem. Comput. Sci. Inf. Syst.(ComSIS) **10**(1), 133–149 (2013)
9. Tuba, M., Jovanovic, R.: Improved ACO algorithm with pheromone correction strategy for the traveling salesman problem. Int. J. Comput. Commun. Control **8**(3), 477–485 (2013)
10. Jovanovic, R., Tuba, M.: An ant colony optimization algorithm with improved pheromone correction strategy for the minimum weight vertex cover problem. Appl. Soft Comput. **11**(8), 5360–5366 (2011)

11. Jovanovic, R., Tuba, M.: An analysis of different variations of ant colony optimization to the minimum weight vertex cover problem. WSEAS Trans. Inf. Sci. Appl. **6**(6), 936–945 (2009)
12. Bacanin, N., Tuba, M.: Artificial bee colony (ABC) algorithm for constrained optimization improved with genetic operators. Stud. Inf. Control **21**(2), 137–146 (2012)
13. Brajevic, I., Tuba, M.: An upgraded artificial bee colony algorithm (ABC) for constrained optimization problems. J. Intell. Manuf. **24**(4), 729–740 (2013)
14. Subotic, M., Tuba, M.: Parallelized multiple swarm artificial bee colony algorithm (MS-ABC) for global optimization. Stud. Inf. Control **23**(1), 117–126 (2014)
15. Tuba, M., Bacanin, N.: Artificial bee colony algorithm hybridized with firefly metaheuristic for cardinality constrained mean-variance portfolio problem. Appl. Math. Inf. Sci. **8**(6), 2831–2844 (2014)
16. Tuba, M., Brajevic, I., Jovanovic, R.: Hybrid seeker optimization algorithm for global optimization. Appl. Math. Inf. Sci. **7**(3), 867–875 (2013)
17. Tuba, M., Bacanin, N.: Improved seeker optimization algorithm hybridized with firefly algorithm for constrained optimization problems. Neurocomputing **143**, 197–207 (2014). doi:10.1016/j.neucom.2014.06.006
18. Yang, X.S.: Firefly algorithms for multimodal optimization. Stochastic Algorithms: Found. Appl. LNCS **5792**, 169–178 (2009)
19. Fister, I., Fister, I.J., Yang, X.S., Brest, J.: A comprehensive review of firefly algorithms. Swarm Evol. Comput. **13**(1), 34–46 (2013)
20. Yang, X.S.: Multiobjective firefly algorithm for continuous optimization. Eng. Comput. **29**(2), 175–184 (2013)
21. Tuba, M., Bacanin, N.: Upgraded firefly algorithm for portfolio optimization problem. In: Proceedings of the 16th IEEE International Conference on Computer Modelling and Simulation, UKSim-AMSS 2014, pp. 112–117. IEEE, New Jersey (2014)
22. Tuba, M., Bacanin, N.: JPEG quantization tables selection by the firefly algorithm. In: Proceedings of the 4th IEEE International Conference on Multimedia Computing and Systems (ICMCS14), IEEE Catalog Number: CFP14050-CDR, Submission 402, pp. 153–158. IEEE, New Jersey (2014)
23. Bacanin, N., Tuba, M.: Firefly algorithm for cardinality constrained mean-variance portfolio optimization problem with entropy diversity constraint. Sci. World J. **2014** (721521) 16 (2014). doi:10.1155/2014/721521
24. Che, Z.G., Chiang, T.A., Che, Z.H.: Feed-forward neural networks training: A comparison between genetic algorithm and back-propagation learning algorithm. Int. J. Innov. Comput. Inf. Control **7**(10), 5839–5850 (2011)
25. Mendes, R., Cortez, P., Rocha, M., Neves, J.: Particle swarm for feedforward neural network training. In: Proceedings of the International Joint Conference on Neural Networks **2**, 1895–1899 (2002)
26. Ilonen, J., Kamarainen, J.K., Lampinen, J.: Differential evolution training algorithm for feed-forward neural networks. Neural Process. Lett. **17**(1), 93–105 (2003)
27. Karaboga, D., Akay, B., Ozturk, C.: Artificial bee colony (ABC) optimization algorithm for training feed-forward neural networks. Lecture Notes in Computer Science: Modeling Decisions for Artificial Intelligence **4617**, 318–329 (2007)
28. Karaboga, D., Ozturk, C.: Neural networks training by artificial bee colony algorithm on pattern classification. Neural Netw. World **19**(3), 279–292 (2009)
29. Brajevic, I., Tuba, M.: Training feed-forward neural networks using firefly algorithm. In: Proceedings of the 12th International Conference on Artificial Intelligence, Knowledge Engineering and Data Bases (AIKED '13), pp. 156–161 (2013)
30. Yang, X.S., Deb, S.: Engineering optimization by cuckoo search. Int. J. Math. Model. Numer. Optim. **1**(4), 330–343 (2010)
31. Brajevic, I., Tuba, M.: Cuckoo search and firefly algorithm applied to multilevel image thresholding. In: X.S. Yang (ed.) Cuckoo Search and Firefly Algorithm: Theory and Applications, *Studies in Computational Intelligence*, vol. 516, pp. 115–139. Springer International Publishing, Switzerland (2014)

32. Yang, X.S., Deb, S.: Cuckoo search via Levy flights. In: Proceedings of World Congress on Nature & Biologically Inspired Computing (NaBIC 2009), pp. 210–214 (2009)
33. Gandomi, A.H., Yang, X.S., Alavi, A.H.: Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems. Eng. Comput. **29**(1), 17–35 (2013)
34. Zhou, Y., Zheng, H., Luo, Q., Wu, J.: An improved cuckoo search algorithm for solving planar graph coloring problem. Appl. Math. Inf. Sci. **7**(2), 785–792 (2013)
35. Yang, X.S.: A new metaheurisitic bat-inspired algorithm. Stud. Comput. Intell. **284**, 65–74 (2010)
36. Yang, X.S.: Bat algorithm for multi-objective optimisation. Int. J. Bio-Inspired Comput. **3**(5), 267–274 (2011)
37. Yang, X.S., Gandomi, A.H.: Bat algorithm: a novel approach for global engineering optimization. Eng. Comput. **29**(5), 464–483 (2012)
38. Moustafa, A.A., Alqadi, Z.A., Shahroury, E.A.: Performance evaluation of artificial neural networks for spatial data analysis. WSEAS Trans. Comput. **10**(4), 115–124 (2011)
39. Tuba, M., Subotic, M., Stanarevic, N.: Modified cuckoo search algorithm for unconstrained optimization problems. In: Proceedings of the European Computing Conference (ECC 2011) pp. 263–268 (2011)
40. Tuba, M., Subotic, M., Stanarevic, N.: Performance of a modified cuckoo search algorithm for unconstrained optimization problems. WSEAS Trans. Syst. **11**(2), 62–74 (2012)
41. Layeb, A.: A novel quantum-inspired cuckoo search for knapsack problems. Int. J. Bio-Inspired Comput. **3**(5), 297–305 (2011)
42. Bacanin, N.: Implementation and performance of an object-oriented software system for cuckoo search algorithm. Int. J. Math. Comput. Simul. **6**(1), 185–193 (2012)
43. Huang, G.Q., Zhao, W.J., Lu, Q.Q.: Bat algorithm with global convergence for solving large-scale optimization problem. Appl. Res. Comput. **30**(3), 1–10 (2013)
44. Du, Z.Y., Liu, B.: Image matching using a bat algorithm with mutation. Appl. Mech. Mater. **203**(1), 88–93 (2012)
45. Tsai, P.W., Pan, J.S., Liao, B.Y., Tsai, M.J., Istanda, V.: Bat algorithm inspired algorithm for solving numerical optimization problems. Appl. Mech. Mater. **148–149**, 134–137 (2011)
46. Alihodzic, A., Tuba, M.: Improved hybridized bat algorithm for global numerical optimization. In: Proceedings of the 16th IEEE International Conference on Computer Modelling and Simulation, UKSim-AMSS 2014, pp. 57–62 (2014)
47. Alihodzic, A., Tuba, M.: Improved bat algorithm applied to multilevel image thresholding. Sci World J **2014**(176718), 16 (2014). doi:10.1155/2014/176718
48. Battiti, R.: First- and second-order methods for learning: Between steepest descent and newtons method. Neural Comput. **4**(2), 141–166 (1992)
49. Srivastava, P.R., Varshney, A., Nama, P., Yang, X.S.: Software test effort estimation: a model based on cuckoo search. Int. J. Bio-Inspired Comput. **4**(5), 278–285 (2012)
50. Yang, X.S., Deb, S.: Cuckoo search: recent advances and applications. Neural Comput. Appl. **24**(1, SI), 169–174 (2014). doi:10.1007/s00521-013-1367-1
51. Yang, X.S., He, X.: Bat algorithm: literature review and applications. Int. J. Bio-Inspired Comput. **5**(3), 141–149 (2013). doi:10.1504/IJBIC.2013.055093
52. He, X., Ding, W.J., Yang, X.S.: Bat algorithm based on simulated annealing and gaussian perturbations. Neural Comput. Appl. **25**(2), 459–468 (2014)
53. Wang, G., Guo, L.: A novel hybrid bat algorithm with harmony search for global numerical optimization. J. Appl. Math. **2013**, 1–22 (2013)
54. Liu, D., Hohil, M.E., Smith, S.H.: N-bit parity neural networks: new solutions based on linear programming. Neurocomputing **48**(1–4), 477–488 (2002)
55. Newman, D., Hettich, S., Blake, C., Merz, C.: UCI repository of machine learning databases (1998). http://www.ics.uci.edu/mlearn/MLRepository.html

The Potential of the Firefly Algorithm for Damage Localization and Stiffness Identification

Sara Casciati and Lorenzo Elia

Abstract The identification of those differences between the current behavior and the initial state of a structure which are indicative of the presence of damage is one of the aims of structural health monitoring. Since the last decades, considerable research advances have been conducted in the optimization field. In this paper, an objective function that minimizes the discrepancies between the analytical and the experimental modal features obtained from the measurements of the actual dynamic response of a structure is formulated. Once the stiffness parameters are set as design variables, the firefly algorithm is applied to carry out the iterations toward the global minima. Partial solutions are analyzed along different steps of the procedure and identified as local optima by calculating the new stiffness matrices and estimating the corresponding values of the objective function. Eventually, the damage detection and localization are pursued by the comparison between the stiffness matrix identified once the optimization process is finished and the starting one. This procedure is applied to a numerical example, which is representative of a generic structure meshed into finite elements where damage is introduced as a local stiffness reduction.

Keywords Damage localization · Firefly algorithm · Metaheuristic methods · Stiffness identification

S. Casciati
Department of Civil Engineering and Architecture,
University of Catania, piazza Federico di Svevia, Siracusa, Italy

L. Elia (✉)
Department of Civil Engineering and Architecture,
University of Pavia, via Ferrata 3, Pavia, Italy
e-mail: lorenzo.elia@unipv.it

X.-S. Yang (ed.), *Recent Advances in Swarm Intelligence and Evolutionary Computation*,
Studies in Computational Intelligence 585, DOI 10.1007/978-3-319-13826-8_9

1 Introduction

In the structural engineering field, most design optimization problems are non-linear, with complicated constrains and a large number of design variables. In order to avoid multiple local optima when dealing with non-linearity, one should resort to global algorithms only, as stated in Gandoni et al. [1]. Commonly, damage is characterized by changes in the modal parameters, e.g., natural frequencies and mode shapes, as reported in Perera et al. [2]. Therefore, an objective function which minimizes the discrepancies between the analytical and the experimental modal features obtained from the measurements of the actual structural dynamic response is herein formulated. The selected design variables are the local stiffness parameters and the firefly algorithm (FA) is applied to carry out the iterations toward the global minima in a metaheuristic framework. At each step of the proposed procedure, the objective function is computed based on the newly generated stiffness matrix. Finally, damage detection and localization are achieved by comparing the stiffness matrix identified once the optimization process is finished with the starting one. The procedure is applied to a numerical example, which is representative of a generic structure meshed into finite elements where the damage is introduced as a local stiffness reduction.

2 The Firefly Algorithm Approach

A large number of classical and current optimization algorithms is deterministic, as mentioned by Yang in [3]. The gradient-based methods are widely used in most deterministic optimization algorithms, as the Newton-Raphson method. Such algorithms are not applicable when the objective function is very discontinuous. In these cases, a gradient-free algorithm is conveniently applied since it requires only to evaluate the function and not its derivatives. In particular, Yang developed in 2007 the firefly algorithm [4] that is based on the behavior and patterns of fireflies. This algorithm was first introduced for continuous optimization and later extended to discrete problems with applications in different fields, such as structural control or image processing among others. Basically, the firefly algorithm is founded on three main assumptions, as stated by Yang in [5]:

- all the fireflies are assumed to be unisex, therefore a firefly is attracted to another one regardless the sex;
- the attractiveness of a firefly is strictly related to its brightness and decreases with the distance;
- the brightness of a firefly is determined based on the objective function.

The brightness is the main aspect for determining the movement of the fireflies [6]. If the maximum is pursued and two fireflies are considered, the less bright one moves towards the brighter one. The reverse occurs if the minimum is sought. In the total

absence of brightness, the fireflies move randomly. The search of the optimal point, (i.e., the global minimum or maximum of the objective function) is performed by the objective function itself, without depending on the calculation of its gradient. This condition allows not to use the derivatives, thus simplifying the computational burden. Unlike classical gradient methods, the firefly algorithm does not introduce any error when either the variables are discrete or the function is not differentiable [7]. Furthermore, the firefly algorithm depends only on few control parameters and requires a small computational effort with respect to other metaheuristic methods, owed to the elementary operations to solve, as sums and differences.

The procedure starts from a randomly generated population (P) of fireflies, whose size is NP. This population belongs to a well-defined existence domain, which depends on the dimension (d) of the optimization problem. Initially the fireflies are uniformly distributed in the entire search domain at locations $\mathbf{x}_i^{(P)}$ with $i = 1, 2, \ldots, NP$. Each individual of the population is considered as a possible candidate to construct the following generation. Furthermore, the Cartesian distance between any two fireflies $\mathbf{x}_i^{(P)}$ and $\mathbf{x}_j^{(P)}$ is calculated as

$$r_{ij}^{(P)} = \left\| \mathbf{x}_i^{(P)} - \mathbf{x}_j^{(P)} \right\| = \sqrt{\sum_{k=1}^{d} \left(x_{i,k}^{(P)} - x_{j,k}^{(P)} \right)^2} \tag{1}$$

where $\boldsymbol{x}_{i,k}^{(P)}$ represents the k-th component of the vector $\mathbf{x}_i^{(P)}$ of the i-th firefly.

Two important items are the variation of the light intensity and the formulation of the attractiveness. The value of the objective function assessed at the current point of the solution space determines the light intensity. When the cost function minimization is pursued, the current firefly shall move toward a less bight one. The attractiveness is defined as

$$\beta_{ij}^{(P)} \left(r_{ij}^{(P)} \right) = (1 - \beta_{\min}) \mathrm{e}^{-\gamma r_{ij}^{(P)^2}} + \beta_{\min} \tag{2}$$

where $\beta_{\min}$ denotes the minimum attractiveness, $(1-\beta_{\min})$ the attractiveness increase at $r_{ij}^{(P)} = 0$, and γ the light absorption. Thus, the movement of a firefly $\mathbf{x}_i^{(P)}$ that is attracted to another one $\mathbf{x}_j^{(P)}$ less bright but more attractive, is given by

$$\mathbf{x}_i^{(P+1)} = \mathbf{x}_i^{(P)} + \beta_{ij}^{(P)} \left(\mathbf{x}_j^{(P)} - \mathbf{x}_i^{(P)} \right) + \zeta \boldsymbol{\kappa}_i^{(P)} \tag{3}$$

where the second term on the right hand side is due to the attraction, and the last one denotes the randomization term, with ζ and $\boldsymbol{\kappa}_i^{(P)}$ representing the randomization parameter and the vector of random numbers drawn from a Gaussian distribution, respectively.

Usually, in structural health monitoring, a typical issue consists of evaluating the performance of a structure without a priori knowing its actual mechanical properties, but only its response under dynamic excitation [8]. In order to solve the inverse problem, a numerical approach consists of running finite elements analyses where the design variables ought to satisfy the requirement of minimizing the discrepancies with the measured response. Furthermore, for linear systems, the requirement becomes a null difference between the generated and the measured modal parameters. Hence, in such cases, the optimal value of the objective function is known and equal to zero.

The adopted solution strategy based on the firefly algorithm is summarized by the flowchart in Fig. 1 and implemented in MATLAB® [9]. As the iterations of the algorithm progress, the fireflies move toward local optima. By comparing the best solution among the optima, the global one is eventually achieved. In this manner, the convergence of the algorithm is achieved from any large number of fireflies.

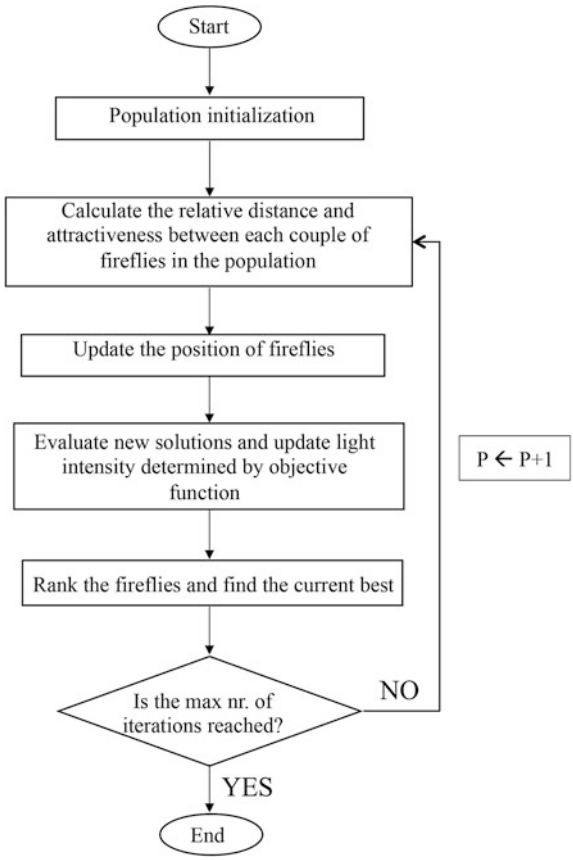


Fig. 1 Flowchart of firefly algorithm

The performance and the accuracy of the method is based on two main steps:

1. the selection of the optimization variables;
2. the formulation of the objective function.

The latter topic is discussed in the following section, for the general structural health monitoring application under study.

The computational burden increases proportionally with the number of design parameter. A numerical example is then used to check whether the method is applicable to high-dimensions problems.

3 The Formulation of the Objective Function

One assumes that the dynamic signature of a structure in its current state is known as either experimentally measured or numerically simulated. Traditional modal analysis tools for linear systems can then be applied to derive its modal features, such as frequencies and mode shapes. The obtained parameters are indicated with the subscript '*exact*' in the following elaborations because they refer to the actual behavior of the structure in its present conditions. In particular, $\bar{\boldsymbol{\omega}}_{exact}$ denotes the $N \times 1$ vector of known natural frequencies, and $\bar{\boldsymbol{\Phi}}_{exact}$ the $N \times N$ matrix of the corresponding modal shapes.

Let $\mathbf{x}$ be a generic $d \times 1$ vector of design parameters in the current population along the firefly algorithm. For easiness of the notation, the superscript tracking the population and the subscript identifying the individual are herein dropped.

The corresponding stiffness matrix, $\mathbf{K}_{gen}(\boldsymbol{x})$, is computed at the beginning of the current step of the algorithm. The frequencies and the mode shapes are then obtained by solving the following eigenvalue-eigenvector problem [8]:

$$\left[\mathbf{K}_{gen}(\boldsymbol{x}) - \omega_{i,gen}^2(\boldsymbol{x})\mathbf{M}\right]\boldsymbol{\phi}_{i,gen}(\boldsymbol{x}) = \mathbf{0}, \quad i = 1,\ldots,N \tag{4}$$

where $\mathbf{M}$ indicates the mass matrix and it is assumed as known and unvaried with respect to the initial state. The resulting eigenvalues are stored in a $N \times 1$ vector $\boldsymbol{\omega}_{gen}(\boldsymbol{x})$, while the eigenvectors in a $N \times N$ matrix $\boldsymbol{\Phi}_{gen}(\boldsymbol{x})$, which has the $N \times 1$ eigenvector $\boldsymbol{\phi}_{i,gen}(\boldsymbol{x})$ as the i-th column.

The objective function is formulated as the norm of the difference between the exact and generated parameters, and is given as follows by using matrix notation:

$$F(\boldsymbol{x}) = \sqrt{\left[\left(\frac{\bar{\boldsymbol{\omega}}_{exact} - \boldsymbol{\omega}_{gen}(\boldsymbol{x})}{\bar{\boldsymbol{\omega}}_{exact}}\right)^{\mathrm{T}} \cdot \mathbf{P} \cdot \left(\frac{\bar{\boldsymbol{\omega}}_{exact} - \boldsymbol{\omega}_{gen}(\boldsymbol{x})}{\bar{\boldsymbol{\omega}}_{exact}}\right)\right]} + wG\left(\bar{\boldsymbol{\Phi}}_{exact}, \boldsymbol{\Phi}_{gen}(\boldsymbol{x})\right) \tag{5}$$

or, equivalently, in explicit scalar form as:

$$F(\boldsymbol{x}) = \sqrt{\sum_{i=1}^{N} \frac{1}{i} \left(\frac{\bar{\boldsymbol{\omega}}_{i,exact} - \boldsymbol{\omega}_{i,gen}(\boldsymbol{x})}{\bar{\boldsymbol{\omega}}_{i,exact}} \right)^2} + w \max_{1 \le j \le N} \left[\frac{\sum_{i=1}^{N} \left(\bar{\boldsymbol{\Phi}}_{ij,exact} - \boldsymbol{\Phi}_{ij,gen}(\boldsymbol{x}) \right)^2}{\sum_{i=1}^{N} \bar{\boldsymbol{\Phi}}_{ij,exact}^2} \right] \tag{6}$$

being $\mathbf{P}$ a $N \times N$ diagonal matrix of weights, where the i-th element $(1/i)$ is selected to prioritize the lower frequencies over the higher ones, which are more affected by the measurements noise.

The first term on the right hand side of Eq. (5) was initially considered alone. The further introduction of the mode shapes results in an improvement for the convergence of the method, as confirmed by the numerical results reported in the following section. A scalar weight, w, is also introduced and preliminary calibrated by manually running the code several times in order to achieve satisfying results. It is worth noting that, when the eigenvectors are considered, the norm of a matrix is not uniquely defined. Indeed, according to [10], the norm of a $N \times N$ matrix $\mathbf{H}$ could be computed as either:

$$l_2\text{norm}(\mathbf{H}) = \left(\text{greatest eigenvalue of } \mathbf{H}^{\mathrm{T}}\mathbf{H}\right)^{1/2} \tag{7}$$

or

$$l_\infty\text{norm}(\mathbf{H}) = \max_{1 \le j \le N} \sum_{i=1}^{N} (H_{ij}) \tag{8}$$

When the $l_2$norm is applied to the j-th column of matrix $\mathbf{H}$, it degenerates to the traditional norm of a vector, i.e.: $l_2\text{norm}(\mathbf{H_j}) \equiv \|\mathbf{H_j}\| = \sqrt{\sum_{i=1,\ldots,N} (H_{ij}^2)}$ with $j = 1, \ldots, N$.

The last term on the right hand side of Eq. (5) is given as the maximum of the ratios between the squares of the $l_2$norm of the vectors $\left[\bar{\phi}_{j,exact} - \phi_{j,gen}(\boldsymbol{x})\right]$ and $\bar{\phi}_{j,exact}$, for $j = 1, \ldots, N$. In mathematical form, one reads

$$G\left(\bar{\boldsymbol{\Phi}}_{exact}, \boldsymbol{\Phi}_{gen}(\boldsymbol{x})\right) = \max_{1 \le j \le N} \frac{\left\| \phi_{j,exact} - \phi_{j,gen}(\boldsymbol{x}) \right\|}{\left\| \bar{\phi}_{j,exact} \right\|^2} \tag{9}$$

whose scalar expression is explicitly represented by the second term in Eq. (6).

Finally, the optimization problem is formulated as follows:

$$\begin{aligned} &\text{minimize } F(\boldsymbol{x}) \\ &\text{under the constraint: } \boldsymbol{x}_{Lb} \leq \boldsymbol{x} \leq \boldsymbol{x}_{Ub} \end{aligned} \tag{10}$$

being $\boldsymbol{x}_{Lb}$ and $\boldsymbol{x}_{Ub}$ the $d \times 1$ vectors containing the lower and the upper bounds of each variable in the design parameters vectors, respectively.

4 The Numerical Example

All numerical studies were performed in the MATLAB® environment, including also the finite element analyses, so that the advantage of transparency at each step of the procedure is achieved.

As shown in Fig. 2, a short cantilever beam of length 10.16 cm and height 5.08 cm is chosen as the case study. The beam is discretized in sixteen, two-dimensional, four-noded (n_e = 4), iso-parametric elements over two layers, under the assumption of plane stress condition. Hence, the mesh consists of 16 elements (m = 16) and 27 nodes (n = 27), with 3 of them that are fixed (n_f = 3), so that the degrees of freedom are $n_{dof} = 2(n-n_f) = 48$.

The material is assumed as isotropic and each element has a Young modulus E = 703.7 MPa, a Poisson ratio v = 0.3, and a mass density $\rho = 7.7\times 10^{-5}\left(\mathrm{N}/\mathrm{mm}^3\right)/\left(\mathrm{mm}/\mathrm{s}^2\right)$.

5 Finite Element Analyses

A classical displacements-based approach is adopted to carry out the finite element analyses [11]. Thus, the local displacements of a single element are expressed as functions of its nodal displacements by an approximate model depending on the selection of the so-called shape function, $\mathbf{N}_e$.

In this context, the element stiffness matrix $\mathbf{K}_e$, of size $2n_e \times 2n_e$, follows the principle of virtual work and is given by

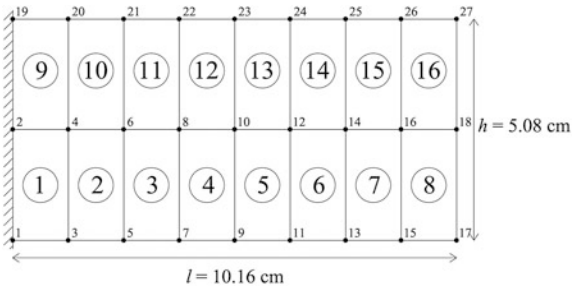


Fig. 2 Finite element discretization of the cantilever beam

$$\mathbf{K}_e = \int_{V_e} \mathbf{B}_e^{\mathrm{T}} \mathbf{D}_e \mathbf{B}_e dV \tag{11}$$

where V_e is the volume of the e-th finite element, $\mathbf{B}_e$ the compatibility matrix that contains the spatial derivatives of the shape functions, and $\mathbf{D}_e$ the matrix of the material constants.

By properly defining the connectivity matrix, $\mathbf{L}_e$, for each e-th element, one can assemble the global stiffness matrix, of size $n_{dof} \times n_{dof}$, as

$$\mathbf{K} = \sum_{e=1}^{m} \mathbf{L}_e^{\mathrm{T}} \mathbf{K}_e \mathbf{L}_e \tag{12}$$

According to this procedure, the continuity of the structure is imposed at the nodes in common to several elements, which have to undergo the same displacements passing from the local to the global reference system. Furthermore, the displacements prevented by the boundary conditions are properly erased during the process of assembling.

Similarly, for a given material of mass density ρ, one defines an element mass matrix as

$$\mathbf{M}_e = \int_{V_e} \rho \mathbf{N}_e^{\mathrm{T}} \mathbf{N}_e \mathrm{d}V \tag{13}$$

and assembles, over all elements, a global mass matrix, of size $n_{dof} \times n_{dof}$, given by

$$\mathbf{M} = \sum_{e=1}^{m} \mathbf{L}_e^{\mathrm{T}} \mathbf{M}_e \mathbf{L}_e \tag{14}$$

Once this process is completed, the resulting stiffness and mass matrices are both symmetric and positive-definite. The $n_{dof} \times 1$ vector of unknown displacements, $\mathbf{u}$, is assumed to be governed by the equations of motion of an undamped system in free vibration: $\mathbf{M}\ddot{\mathbf{u}} + \mathbf{K}\mathbf{u} = \mathbf{0}$. By expressing the displacements in terms of modal coordinates, classical modal analysis is performed.

In the present work, a damage is induced to any e-th element of the structure by simply multiplying the corresponding local stiffness matrix by a non-dimensional quantity, α_e, whose real value falls in the interval between 0 and 1. In other words, a degradation of stiffness in the e element yields to a damaged stiffness matrix which can be expressed as

$$\mathbf{K}_{e,dam} = \alpha_e \mathbf{K}_e \tag{15}$$

Table 1 Different structural configuration and damage scenarios

Structural configuration	Element stiffness coefficients															
	α_1	α_2	α_3	α_4	α_5	α_6	α_7	α_8	α_9	α_{10}	α_{11}	α_{12}	α_{13}	α_{14}	α_{15}	α_{16}
Undamaged	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Damage A	0.8	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Damage B	1	1	1	1	1	1	1	1	0.8	1	1	1	1	1	1	1
Damage C	0.8	1	1	1	1	1	1	1	0.8	1	1	1	1	1	1	1
Damage D	0.4	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Damage E	1	1	1	1	1	1	1	1	0.4	1	1	1	1	1	1	1
Damage F	1	1	1	1	1	1	1	1	1	1	1	0.8	1	1	1	1
Damage G	0.8	0.9	1	1	1	1	1	1	0.7	0.8	1	1	1	1	1	1

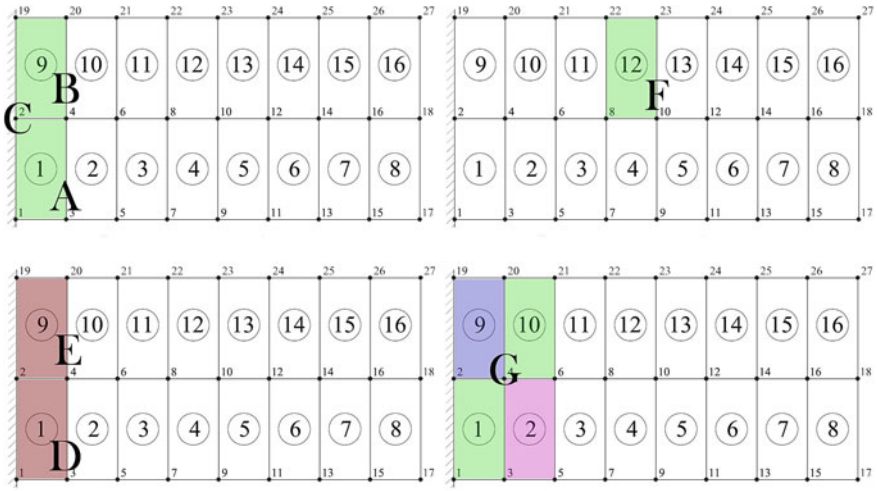


Fig. 3 Different damage scenarios for each structural configuration

with $0 < \alpha_e < 1$, being $\alpha_e = 1$ associated to the undamaged reference condition. The analyses are carried out for different damage scenarios as summarized in Table 1 and Fig. 3.

It is worth noting that the mass matrix can be constructed using two different methods:

1. the herein adopted finite elements method is based on a consistent mass matrix, i.e., a full matrix of non-null inertia terms, that also includes the rotational inertia;
2. the classical assumption that the masses are lumped at the nodal points yields, instead, to a diagonal mass matrix which, for the element e, is given by $\mathbf{M}_{L_e} = (\rho V_e / n_e)\mathbf{I}_e$, with $\mathbf{I}_e$ the identity matrix of size $2n_e \times 2n_e$.

Table 2 First exact nine modal frequencies for different values of the element stiffness coefficients

Structural configuration	Exact values of the first nine circular frequencies (rad/s)								
	ω_1	ω_2	ω_3	ω_4	ω_5	ω_6	ω_7	ω_8	ω_9
Undamaged	66.86	235.38	262.48	562.97	701.09	733.97	956.56	965.88	1,059.69
Damage A	65.41	232.10	258.81	555.75	692.53	727.73	955.39	959.60	1,046.85
Damage B	65.41	232.10	258.81	555.75	692.53	727.73	955.39	959.60	1,046.85
Damage C	63.98	228.67	254.64	548.06	684.12	720.89	953.07	954.65	1,036.58
Damage D	60.03	220.53	248.33	533.26	660.24	710.43	941.83	952.27	999.05
Damage E	60.03	220.53	248.33	533.26	660.24	710.43	941.83	952.27	999.05
Damage F	66.44	233.17	260.35	557.90	699.04	733.11	952.10	955.97	1,047.43
Damage G	61.73	222.33	248.70	540.96	674.26	713.24	944.25	953.27	1,027.94

However, the purpose of this study is limited to the identification of only the stiffness matrix.

The exact values of the natural frequencies and mode shapes are achieved by solving the eigenvalue problem in Eq. (4) for all the structural configurations defined in Table 1. The resulting first nine circular frequencies are reported in Table 2 for each of the considered cases. In the following analyses, the quantities $\bar{\omega}_{exact}$ and $\bar{\Phi}_{exact}$ have size $n_{dof} \times 1$ and $n_{dof} \times n_{dof}$, respectively, where n_{dof} denotes the number of degrees of freedom.

6 Identification of the Stiffness Matrix via FA

All the performed analyses aim to recognize the previously defined damage scenarios (see Table 1), given the identified modal features. Once the frequencies $\bar{\omega}_{exact}$ and the mode shapes $\bar{\boldsymbol{\Phi}}_{exact}$ are introduced in the objective function defined in Eq. (5), the firefly algorithm is applied to solve the optimization problem stated in Eq. (10). The following $d \times 1$ parameters vector collects the coefficients of the element stiffness matrix which have to be identified:

$$\boldsymbol{x} = [\alpha_1\ \alpha_2\ \alpha_3\ \alpha_4\ \alpha_5\ \alpha_6\ \alpha_7\ \alpha_8\ \alpha_9\ \alpha_{10}\ \alpha_{11}\ \alpha_{12}\ \alpha_{13}\ \alpha_{14}\ \alpha_{15}\ \alpha_{16}]^{\mathrm{T}} \tag{16}$$

Theoretically, the existence domain of each parameter is the interval between 0 and 1 on the real line. In order to numerically identify the proper search domain where the convergence of the method is achieved, a preliminary study is carried out on the undamaged structure. For each case defined in Table 1, damage detection and localization are then pursued.

Table 3 Input parameters of the firefly algorithm

Control parameter	Value
NP, Size of the initial population of fireflies	50
I_{max} Maximum number of iterations	1,000–1,500
ζ, randomization parameter	0.5
β_{min}, minimum attractiveness	0.2
γ, absorption coefficient	1

6.1 A Preliminary Study on the Undamaged Configuration

In the following analyses, the entire sets of the eigenvalues and eigenvectors are considered as inputs to the firefly algorithm. Furthermore, in Eq. (4), one assumes $N = n_{dof} = 48$. The undamaged state is preliminary investigated to calibrate the control parameters of the firefly algorithm and to define a proper search domain [12]. In all the analyses, a unit weight, $w = 1$, is assumed for the contribution of the eigenvectors in the objective function. The control parameters of the FA are reported in Table 3.

In particular, the randomization parameter, ζ, the minimum attractiveness, β_{min}, and the absorption coefficient, γ, are kept constant in all the analyses, as suggested in literature and in most implementations [5]. Instead, the size of the initial population, *NP*, and the maximum number of iterations, I_{max}, may change depending on the scale of the optimization problem.

The definition of the search domain of each variable is assigned as an interval centered around the value $x_0 = 1.0$. The lower and upper bounds of this interval are given as $x_{Lb} = (4/5)x_0 = 0.80$ and $x_{Ub} = (6/5)x_0 = 1.20$, respectively. Under these assumptions, the solution in the undamaged case is reached after 224 iterations, with an error close to 0 %.

6.2 Damage Localization Analyses

Several analyses are carried out to localize the damage in the structure under different damage scenarios. For the first and the second damage cases, denoted as A and B in Table 1, the correct solution should localize and quantify the damage in correspondence with elements 1 and 9, respectively, which are both adjacent to the fixed edge of the beam. The path to convergence for damaged case A is shown in Fig. 4 by plotting the successive values of the objective function versus the number of iterations. For case A the convergence to a null value of the objective function is achieved after 196 iterations, and a similar trend is also observed for the case B.

The typical shape of the objective function under consideration is shown in Fig. 5 for the case when the weight w in Eq. (5) is set equal to 0, i.e., there is no eigenvector contribution.

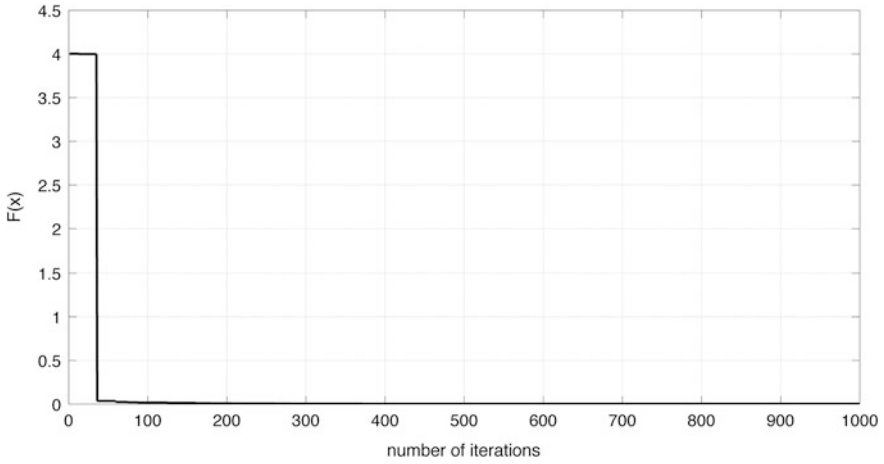


Fig. 4 Path to convergence for damaged case A

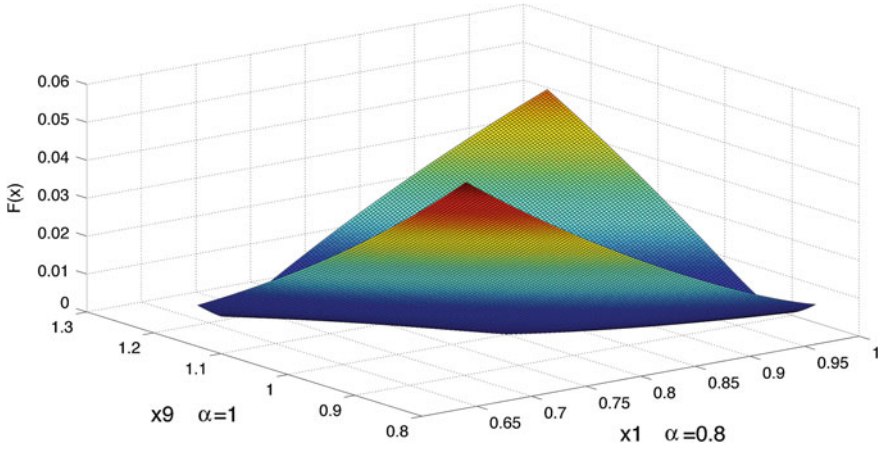


Fig. 5 Objective function for damaged case A with $w = 0$

When, instead, the weight in Eq. (5) is set equal to 1, a curve presenting a very steep slope in the neighborhood of the global minimum is obtained, as shown in Fig. 6. The global optimum is located at the bottom of a 'well', and the challenge consists of entering the 'well' without first getting trapped on a local minimum. Indeed, it is evident from the plots in Figs. 5 and 6 that there exist several situations in which the requirement on the natural frequencies is satisfied, but there is only one point where also the eigenvectors' difference is minimized. This point represents the searched global minimum of the objective function, and it is achieved when the current vector of the design parameters corresponds to the actual one. Therefore,

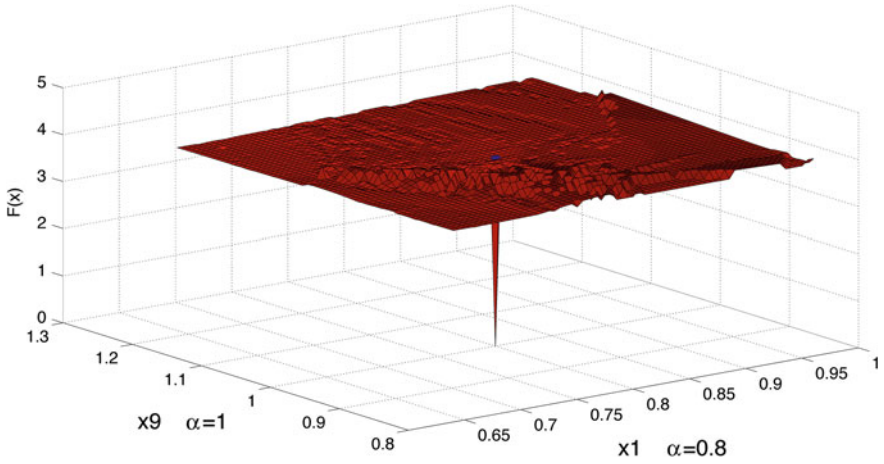


Fig. 6 Objective function for damaged case A with $w = 1$

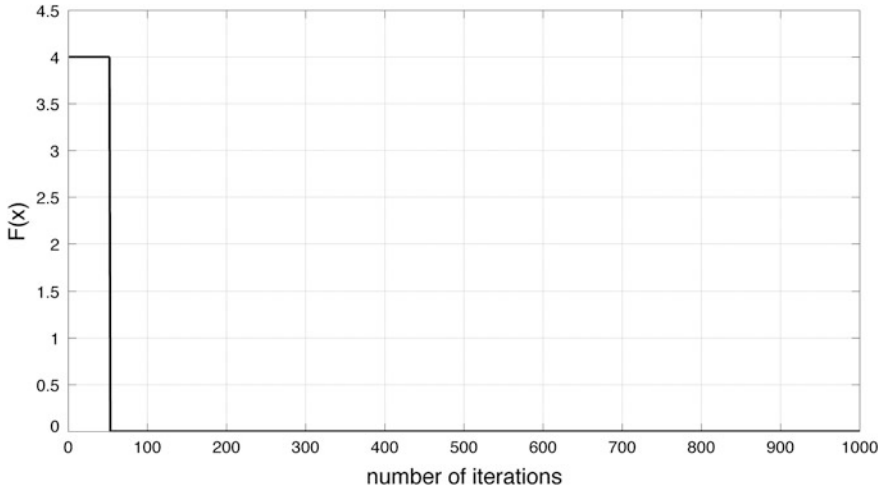


Fig. 7 Path to convergence for damaged case F

the strength of the objective function formulation given in Eqs. (5) and (6) is represented by the existence and the uniqueness of the global minimum.

The detection of multiple damages is also carried out by considering case C, where the damage is symmetrically introduced in both elements 1 and 9. For this case, the convergence is achieved after 375 iterations.

When considering cases D and E, the location of the single damaged element coincides with the one of cases A and B, respectively, but the intensity of damage is higher. As a consequence, the number of iterations necessary for convergence reduces to 76.

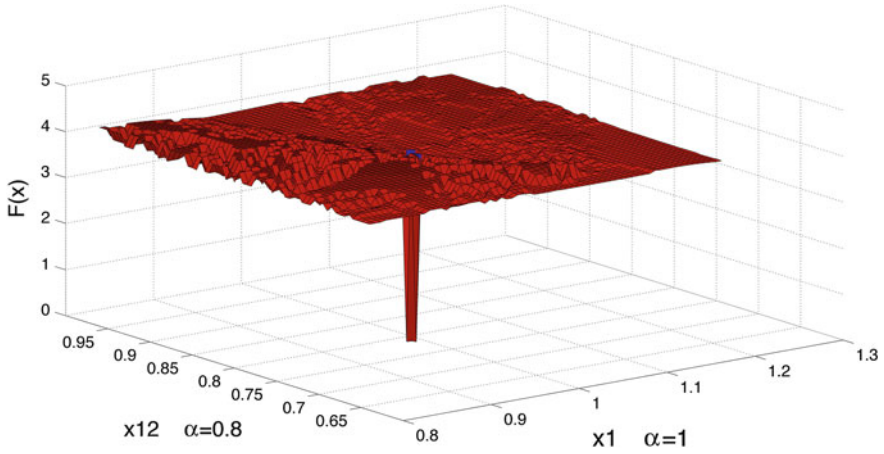


Fig. 8 Objective function for damaged case F with $w = 1$

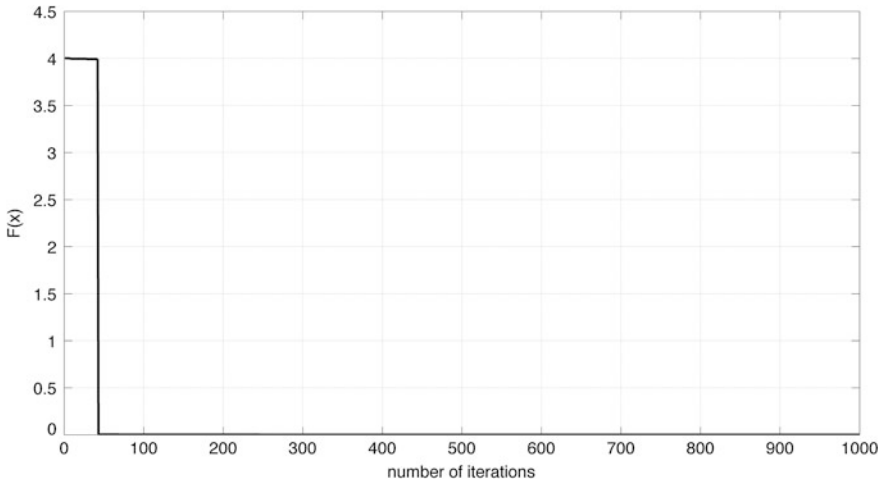


Fig. 9 Path to convergence for damaged case G

To verify that the central elements behave in the same manner as the edge ones, a small damage is introduced in the element 12, as shown in Fig. 3 for case F. The convergence is reached after 217 iterations. The path to convergence and the shape of the objective function with $w = 1$ are shown in Figs. 7 and 8, respectively.

The last analyzed case (labelled as G in Table 1) is characterized by multiple, non-symmetric damage. The exact solution consists of a local stiffness coefficient of 0.8 in elements 1 and 10, 0.9 in element 2, and 0.7 in element 9 (see Fig. 3). With respect to the other cases, an increased maximum number of iterations from 1,000

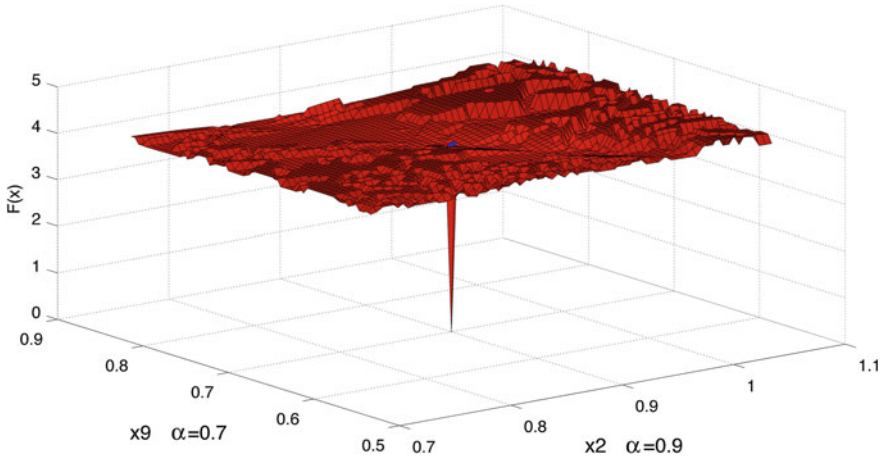


Fig. 10 Objective function for damaged case G with $w = 1$

to 1,500 is assigned, while the population size and the other parameters are unchanged.

The path to convergence and the shape of the objective function for the case G are shown in Figs. 9 and 10, respectively. The performed analysis shows that the convergence can be reached after only 114 iterations.

7 Conclusions

In this chapter, an objective function to minimize the difference between the analytical and the experimental modal characteristics of the dynamic response of a structure is formulated. The so-called firefly algorithm, which belongs to the class of the nature-inspired metaheuristic algorithms, is adopted as the solving technique. Once the damage is introduced [13, 14] in a generic structure as a local stiffness deterioration, the proposed procedure permits to detect and localize it by the identification of the change in the stiffness matrix with respect to the initial and undamaged state as stated in [15] by Casciati and Faravelli and in [16] by Faravelli and Marazzi.

From the results of the analyses performed on a numerical example, the firefly algorithm can converge on global optima with a quite small population of fireflies and a consequent moderate computational burden.

Acknowledgments The authors gratefully acknowledge the financial support provided by the corresponding Athenaeum Research Grants.

References

1. Gandomi, A.H., Yang, X.S., Alavi, A.H.: Mixed variable structural optimization using firefly algorithm. Comput. Struct. **89**, 2336–2535 (2011)
2. Perera, R., Ruiz, A., Manzano, C.: An evolutionary multiobjective framework for structural damage localization and quantification. Eng. Struct. **292**, 2540–2550 (2007)
3. Yang, X.S.: Nature-inspired metaheuristic algorithms, 2nd edn. Luniver Press (2010)
4. Yang, X.S.: Multiobjective firefly algorithm for continuous optimization. Eng. Comput. **29**, 175–184 (2013)
5. Yang, X.S.: Firefly algorithms for multimodal optimization, in stochastic algorithms: foundations and applications, SAGA 2009. Lect. Notes Comput. Sci. **5792**, 169–178 (2009)
6. Casciati, F., Elia, L., Faravelli, L.: Optimization of sensors location for structural monitoring, on the proceedings of OPT-i, international conference on engineering and applied sciences optimization. Kos Island, Greece, (2014)
7. Casciati, S., Elia, L.: Potential of metaheuristic methods for damage localization and stiffness identification, on the proceedings of OPT-i, international conference on engineering and applied sciences optimization. Kos Island, Greece (2014)
8. Casciati, S.: Stiffness identification and damage localization via differential evolution algorithms. Struct. Control Health Monit. **15**, 439–449 (2008)
9. Matlab Matlab user manual. Mathworks Inc., Lowell (2013)
10. Nair, K.K., Kiremidjian, A.S., Law, K.H.: Time series-based damage detection and localization algorithm with application to the ASCE benchmark structure. J. Sound Vib. **291**, 349–368 (2006)
11. Papadimitriou, C.: Optimal sensor placement methodology for parametric identification of structural systems. J. Sound Vib. **278**, 923–947 (2004)
12. Morlier J.: Méthodes d'analyse des déformées modales par traitement du signal pour le diagnostic in situ de structures. Ph.D. Thesis, University of Bordeaux, France (2005) (in French)
13. El-Borgi, S., Choura, S., Ventura, C., Baccouc, M., Cherif, F.: Modal identification and model updating of a reinforcement concrete bridge. Smart Struct. Syst. **1**, 83–101 (2005)
14. Savoia M., Vincenzi L.: Differential evolution algorithm for dynamic structural identification. In: Proceedings of ICOSSAR'05, Rome, Italy, Millpress, Rotterdam (2005)
15. S. Casciati, L. Faravelli, Stiffness matrix estimation via differential evolution algorithm. In: Proceedings of the Third European Workshop on Structural Health Monitoring, Granada, Spain. DEStech Publications, Lancaster, U.S.A. (2006)
16. Faravelli, L., Marazzi F.: Stiffness matrices and genetic algorithm identifiers toward damage detection. In: Proceedings of IABMAS'06, Porto, Portugal (2006)

Synthesizing Cross-Ambiguity Functions Using the Improved Bat Algorithm

Momin Jamil, Hans-Jürgen Zepernick and Xin-She Yang

Abstract The cross-ambiguity function (CAF) relates to the correlation processing of signals in radar, sonar, and communication systems in the presence of delays and Doppler shifts. It is a commonly used tool in the analysis of signals in these systems when both delay and Doppler shifts are present. In this chapter, we aim to tackle the CAF synthesization problem such that the synthesized CAF approximates a desired CAF. A CAF synthesization problem is addressed by jointly designing a pair of waveforms using a metaheuristic approach based on the echolocation of bats. Through four examples, it is shown that such an approach can be used as an effective tool in synthesizing different types of CAFs.

Keywords Cross-ambiguity function · Metaheuristic algorithm · Improved bat algorithm · CAF synthesization

1 Introduction

In a conventional matched filter receiver, the internal reference waveform is a duplicate of the transmitted signal, i.e., the receiver reference waveform is matched to the transmitted signal [1]. However, in radar applications, the appropriate time delay and compression must be taken into account at the receiver side. Therefore, in

M. Jamil (✉)
Automotive Division, Harman International, Becker-Goering Str. 16,
76307 Karlsbad, Germany
e-mail: momin.jamil@harman.com

M. Jamil · H.-J. Zepernick
Blekinge Institute of Technology, 371 79 Karlskrona, Sweden
e-mail: hans-jurgen.zepernick@bth.se

X.-S. Yang
School of Science and Technology, Middlesex University,
London NW4 4BT, UK
e-mail: x.yang@mdx.ac.uk

X.-S. Yang (ed.), *Recent Advances in Swarm Intelligence and Evolutionary Computation*,
Studies in Computational Intelligence 585, DOI 10.1007/978-3-319-13826-8_10

a conventional matched filter receiver, the receiver waveform is a replica of the transmitted signal with appropriate time delay and time compression. However, a conventional receiver is not able to take care of clutter or jamming suppression. In a radar system, clutter appears as signal echoes with different delays or Doppler shifts compared to the signal of interest. In order to suppress impairments due to clutter and interference, it is desirable to minimize these effects at the receiver side. Accordingly, a joint design of the transmit signal and receive filter is desirable such that the signal-to-clutter-plus-interference ratio (SCIR) of the receiver output is maximized at the time of target detection [2]. As a result, an alternative to conventional receivers, known as a general or optimum receiver, was proposed in [1]. This receiver can be used as a trade-off between the signal-to-noise ratio (SNR) for improved SCIR [1]. In an optimum receiver, the internal or reference waveforms (or equivalent filter) may be deliberately mismatched to reduce the sidelobes in the delay-Doppler plane.

The aforementioned joint design for clutter/interference suppression has been addressed in [2–9] and the references therein. However, a joint design of the mismatched filter at the receiver side and the transmit signal leads to a more complex optimization problem that involves either assessing cross-correlation (CC) properties with respect to delay in the case of negligible Doppler shifts or focusing on cross-ambiguity function (CAF) characteristics in the delay-Dopper plane otherwise [4, 10].

A performance measure frequently used to assess waveforms for radar, sonar, and communication applications in the presence of delay and Doppler shifts, known as ambiguity function (AF), was proposed in [11]. An AF is a function of two variables representing correlation properties of a signal in the delay-Doppler plane. It provides a mathematical representation of the response of a matched filter to a received waveform. The waveform design that would yield an optimum AF has been on the forefront of research for many years. In an ideal case, an AF would have the shape of a spike at the origin and zero elsewhere in the delay-Doppler plane. Although such an AF is certainly desirable, in practice, it is not realizable for signals having finite energy. As a result, large efforts have been given to waveform designs that relax the zero sidelobe constraint throughout the delay-Doppler plane to uniformly low sidelobes, while still maintaining a reasonable large value at the origin. In practice, radar waveforms are often designed by minimizing the sidelobes of an auto-correlation function (ACF), i.e., by basically matching pre-defined specifications only to the zero-Doppler cut of an AF [12].

In Woodward [11], the importance of signal designs using waveform synthesis for radar and sonar applications has been stressed. Nevertheless, the search for practical solutions to the synthesis problem still poses a challenge to radar system engineers. A first known mathematical solution to the synthesization problem was presented in [13]. However, this solution has two drawbacks: (i) it requires that the shape of a desired ambiguity function is given in analytical form, (ii) it does not cope with settings where only certain parts of the ambiguity surface are to be approximated, e.g., the clear area in and around a large neighbourhood of the origin. As a consequence, this solution is of limited interest to practical radar applications.

In practice, radar engineers typically have a general idea about the desirable shape of an AF rather than an exact expression of it as a mathematical function. Furthermore, in many scenarios, it is not even necessary to specify the shape of an AF for the entire delay-Doppler plane. In other words, the region where an AF is required to produce small values very much depends on the particular radar application. For example, the Doppler shift may be much smaller compared to the bandwidth of the transmitted waveform which can be in the order of several megahertz. In this case, the AF for Doppler shifts beyond the maximum induced shifts is not required. An alternative approach of constructing a waveform with optimal ambiguity surface in a region around the main lobe of an AF using well-know Hermite waveforms has been presented in [14].

In single-input single-output (SISO) radar systems, the problem becomes to synthesize a single radar waveform that approximates a desired auto-ambiguity function (AAF) of pre-defined magnitude over the delay-Doppler plane. On the other hand, multiple-input multiple-output (MIMO) radar systems or communication systems involve pairs of signals rather than a single waveform. Accordingly, the synthesis problem focuses on the CAF between a pair of signals. In the considered context, the CAF describes the receiver response to a mismatched signal as a function of time and Doppler shift. In particular, the continuous-time CAF is defined as

$$\chi(\tau, f_d) = \int_{-\infty}^{\infty} a(t) b^*(t+\tau) \exp(j2\pi f_d t) dt, \tag{1}$$

where $a(t)$ and $b(t)$ are arbitrary waveforms as a function of time t, τ is delay, f_d denotes Doppler frequency/Doppler shift, $(\cdot)^*$ denotes complex conjugate, and $j = \sqrt{-1}$. In practice, the CAF is applicable for a SISO radar system when $a(t)$ is the transmit signal and $b(t)$ represents the receive filter [15]. Similarly, the CAF is used for a MIMO radar system when both $a(t)$ and $b(t)$ are different transmit signals [16]. In a conventional matched filter receiver, where the receiver reference waveform is matched to the transmitted signal [1], i.e. $a(t) = b(t)$, the CAF becomes an AAF.

Let us now consider two signals $a(t)$ and $b(t)$, consisting of a train of N pulses $s_i(t)$ and $s_j(t)$, respectively, as

$$a(t) = \sum_{i=1}^{N} a_i s_i(t), \tag{2}$$

$$b(t) = \sum_{j=1}^{N} b_j s_j(t), \tag{3}$$

where the coefficients a_i and b_i can be expressed as column vectors of length N as

$$\mathbf{a} = (a_1, a_2, \ldots, a_N)^T, \tag{4}$$

$$\mathbf{b} = (b_1, b_2, \ldots, b_N)^T, \tag{5}$$

and $s_k(t)$; $k = i, j$ denotes a pulse shaping function. For example, a rectangular pulse shaping function is defined as

$$s_k(t) = \frac{1}{\sqrt{T_c}} s\left[\frac{t-(k-1)T_c}{T_c}\right], \quad k = 1, 2, \ldots, N, \tag{6}$$

where T_c denotes the pulse duration and

$$s(t) = \begin{cases} 1, & 0 \le t \le T_c, \\ 0, & \text{elsewhere.} \end{cases} \tag{7}$$

Substituting (2), (3), and (6) into (1), the CAF comprising of pulse shaping functions with respective shifted pulses and corresponding coefficients a_i and b_j can be obtained as

$$\chi(\tau, f_d) = \sum_{i=1}^{N} \sum_{j=1}^{N} a_i b_i^* \int_{-\infty}^{\infty} s_i(t) s_j^*(t+\tau) \exp(j2\pi f_d t) dt, \tag{8}$$

where the integral represents the CAF between pairs of pulse shaping functions, i.e.,

$$\hat{\chi}_{i,j}(\tau, f_d) = \int_{-\infty}^{\infty} s_i(t) s_j^*(t+\tau) \exp(j2\pi f_d t) dt. \tag{9}$$

Clearly, synthesizing a CAF such that it matches a desired CAF of pre-defined magnitude over the delay-Doppler plane is a difficult task. As a result, not many methods, other than solutions based on the least-squares approaches exist, see, e.g., [1, 14, 17–21]. Recently, in [22], an algorithm has been proposed to match a synthesized CAF to a desired CAF of pre-defined magnitude over the delay-Doppler plane. More specifically, this algorithm proposes a joint design of a pair of signals $a(t)$ and $b(t)$, or sequences $\mathbf{a}$ and $\mathbf{b}$ to tackle the CAF synthesization problem. Furthermore, in [23], Jamil et al. proposed a Lévy flight based cuckoo search for a joint sequence design such that their CAF approximates a desired CAF indicating the potential of metaheuristic approaches to solve such challenging sequence design problems.

In view of the above, this chapter considers a joint sequence design using the improved bat algorithm (IBA) of [24] to address the problem of matching a synthesized CAF to a desired CAF of pre-defined magnitude over the delay-Doppler

plane. We hypothesize that a joint design of a pair of sequences **a** and **b** such that their CAF approximates a desired CAF is a global optimization problem (GOP). Apparently, this type of problem is a highly multimodal problem without any a priori information about the location of the optimum solution (unimodal) or solutions (multimodal). Traditional optimization methods that require either a good initial guess or gradient information are unsuitable to solve such problems to optimality. Therefore, nature-inspired population methods, mimicking the behaviour of different species of animals, have been proposed to solve such problems [25–27]. Due to their general applicability and effectiveness, these algorithms have been a popular choice to solve modern optimization problems. These population-based algorithms use population members to explore the problem search space for a possible solution or solutions by maintaining a balance between intensification (exploitation) and diversification (exploration). However, intensification (exploitation) and diversification (exploration) are usually based on a uniform or Gaussian distribution. Lévy flights (LFs) based on the Lévy distribution have been proposed as an alternative to achieve exploitation and exploration strategies.

The remainder of this chapter is organized as follows. In Sect. 2, we briefly introduce the Lévy probability distribution, and the motivation of using LFs in metaheuristic algorithms. In Sect. 3, we present the improved bat algorithm in detail. In Sect. 4, the formulation and solution to the considered synthesis problem is presented. Numerical results are presented in Sect. 5. Finally, Sect. 6 concludes the chapter.

2 Lévy Probability Distribution

2.1 Lévy Distribution

A random process is called stable if the sum of a given number of independent random variables, X_1, X_2, …, X_N, has the same probability density function (PDF) up to location and scale parameters as the individual random variables. A well-known example of a stable random process is a Gaussian process, i.e., the sum of Gaussian random variables also produces a Gaussian distribution which in addition has a finite second moment. A stable random process with infinite second moment produces a so-called α-stable distribution. An α-stable random variable S is defined by its characteristic function as follows [28]:

$$\Phi_{\alpha,\beta} = E[\exp(jzS)] = \exp(-\beta^{\alpha}|z|^{\alpha}) \tag{10}$$

where $E[\cdot]$ denotes the expectation operator, $j = \sqrt{-1}$, $z \in \mathbb{R}$, $\alpha \in (0, 2]$ and $\beta \geq 0$. The Lévy probability distribution belongs to a special class of symmetric α-stable distributions. According to [28], the PDF of a symmetric α-stable random variable is given by the inverse Fourier transform of (10) as

$$L_{\alpha,\beta}(S) = \frac{1}{\pi}\int_0^{\infty} \exp(-\beta z^{\alpha})\cos(zS)dz \tag{11}$$

In (11), the parameters α and β control the shape and the scale of the distribution, respectively. The parameter α takes values in the interval $0 < \alpha \leq 2$ and controls the heaviness of the distribution, i.e., the decay of the tails. The smaller the value of α, the more the accumulation of data in the tails of the distribution. In other words, the random variable values are more likely to be far away from the mean of the distribution. On the other hand, the larger the value of α, the more the accumulation of data near the mean of the distribution. Except for a few special cases, a closed-form expression of the integral in (11) is not known for α in general. The integral in (11) becomes a Cauchy distribution and Gaussian distribution for α = 1 and 2, respectively.

2.2 *Lévy Flight Based Metaheuristic Algorithms*

In recent years, a number of theoretical and empirical studies have tried to explain that foragers such as grey seals [29], microzooplankton [30, 31], reindeer [32], wandering albatrosses [33], fish [34], among many others, adapt LF as an optimal search strategy in search of food. However, it should be mentioned that foragers adapt their search strategy based on the density of prey, sometimes switching between LF and Brownian motion (BM). In metaheuristic and stochastic optimization algorithms, random walks play an important and central role in the exploration of the problem search space. The search performed by metaheuristic algorithms (MAs) is carried out in a way that it can accomplish goals of intensively explored areas of the search space with high-quality solutions and move to unexplored areas of the search space when necessary. Intensification and diversification [35, 36] are two key ingredients to achieve these goals. By maintaining a fine balance between these two components define the overall efficiency of MA. In fact, Lévy flights have already been used to enhance metaheuristic algorithms with promising results in the literature, including the cuckoo search and firefly algorithm [23, 25, 26].

An alternative to a uniform or Gaussian distribution to realize randomization in MA is offered by the Lévy distribution. Not only does the power law behavior of a Lévy distribution reduce the probability of returning to previously visited sites in the problem search space, but it also provides an effective and efficient exploration mechanism of the far-off regions of the function landscape.

3 Improved Bat Algorithm

The bat algorithm (BA) mimicking the echolocation behavior of certain species of bats was presented in [27] and is based on the following set of rules and assumptions:

1. All bats know the difference between food/prey, background barriers, and use echolocation to sense the proximate distance from the prey;
2. In search mode, bats fly randomly with a frequency f_{min} with velocity $\mathbf{v}_i$ at position $\mathbf{x}_i$. During search mode, bats vary wavelength λ (or frequency f) and loudness A_0. Depending on the proximity from the target, bats can automatically adjust the wavelength (or frequency) for their emitted pulses and adjust the rate of pulse emission $r \in [0, 1]$;
3. It is further assumed that the loudness varies from a large (positive A_0) to a minimum value of A_{min};
4. Ray tracing is not used in estimating the time delay and three dimensional topography;
5. The frequency f is considered in a range $[f_{min}, f_{max}]$ corresponding to the range of wavelengths $[\lambda_{min}, \lambda_{max}]$;
6. For simplicity, frequency is assumed in the range $f \in [0, f_{max}]$.

According to [27], by making use of the above rules and assumptions, the standard bat algorithm (SBA) will always find the global optimum. However, in SBA, the bats rely purely on random walks drawn from a Gaussian distribution, therefore, speedy convergence may not be guaranteed [27]. To improve the bat algorithm further, Jamil et al. developed the improved bat algorithm (IBA) [24].

In this section, a brief overview of the IBA [24] is presented which constitutes an improved version of SBA [27]. In IBA, the random motion of bats is replaced by LF instead of using a Gaussian distribution. The motivation for this choice is that the power-law behavior of the Lévy distribution will produce some members of the random population in the distant regions of the search space, while other members will be concentrated around the mean of the distribution. The power-law behavior of the Lévy distribution also helps to induce exploration at any stage of the convergence, making sure that the system will be not trapped in local minima. The Lévy distribution also reduces the probability of returning to the previously visited sights, while the number of visitations to new sights is increased [24, 25, 27]. For a comprehensive review of the bat algorithm and its variants, please refer to [37].

3.1 Motion of the Bats

In IBA, the position or location of each bat is given as $\mathbf{x}_i^t$ and it flies through the D-dimensional search space or solution space with a velocity $\mathbf{v}_i^t$. The position and velocity for bat i are updated at time t, respectively, as

$$\mathbf{v}_i^t = \mathbf{v}_i^{t-1} + (\mathbf{x}_i^{t-1} - \mathbf{x}_i^{best})f_i, \tag{12}$$

$$\mathbf{x}_i^{t+1} = \mathbf{x}_i^{t-1} + \mathbf{v}_i^t \Delta t, \tag{13}$$

where Δt represents the discrete time step of the iteration. However, in mathematical optimization, emphasis is often given to dimensionless variables, and therefore, Δt can be implicitly chosen as 1. Furthermore, the pulse frequency f_i for bat i at position $\mathbf{x}_i$ is given by

$$f_i = f_{min} + (f_{max} - f_{min})\beta, \tag{14}$$

and vectors $\mathbf{x}_i$ and $\mathbf{v}_i$ represent the position and velocity of bat i. In (14), $\beta \in [0, 1]$ is a random number drawn from a uniform distribution, f_{min} and f_{max} denote the minimum and maximum frequency of the emitted pulse [27]. The symbol $\mathbf{x}_i^{best}$ in (12) represents the current best solution found by bat i by comparing all the solutions among all the NP bats.

In IBA [24], once a best solution is selected among the current best solutions, a new solution for each bat is generated using an LF that is based on a Lévy distribution according to

$$\mathbf{x}_i^t = \mathbf{x}_i^{best} + \gamma \cdot \mathbf{L}_\alpha(S). \tag{15}$$

Here, vector $\mathbf{L}_\alpha(S)$ represents a random walk that is generated based on the Lévy distribution for each i (bat) with parameter α. The parameter $\gamma > 0$ scales the random step length and is related to the scales of the problem [25–27]. Specifically, the step size S of the random walk is drawn from a Lévy distribution (with an infinite mean and variance) that is often given in terms of a power-law formula given as [25, 26, 28]

$$L_\alpha(S) \sim \frac{1}{S^{\alpha+1}}, |S| \gg 0, \tag{16}$$

where α is the exponent determining the shape of the tail of the distribution.

3.2 Variation of Loudness and Pulse Rates

In IBA, we use the originally proposed approach of controlling the exploration and exploitation in bats as proposed in [27], i.e., variation of loudness and pulse rates. In order to switch to the exploitation stage when necessary, each bat i varies its loudness A_i and pulse emission rate r_i iteratively as follows:

$$A_i^{t+1} = \Upsilon A_i^{t_0}, \tag{17}$$

$$r_i^{t+1} = r_i^{t_0}[1 - \exp(-\Gamma t)], \tag{18}$$

where $A_i^{t_0}$, A_i^{t+1}, $r_i^{t_0}$, and r_i^{t+1}, respectively, represent initial loudness, updated loudness, initial pulse emission rate, and updated pulse emission rate after each iteration for bat i. Furthermore, Υ and Γ are constants.

4 Problem Formulation

In practice, infinite energy signals do not exist, therefore, ambiguity surfaces that produce a Dirac impulse or a function with ideal delay-Doppler characteristics do not exist. Thus, it is often desirable to design waveforms that exhibit a peak at the origin and produce an almost flat surface in and around a large neighborhood of the origin.

The problem of matching a CAF to a desired CAF can be formulated as a minimization problem and can be solved by using the cyclic approach proposed in [22]. Accordingly, such an optimization problem can be formulated as

$$\min_{\mathbf{a},\mathbf{b}} C(\mathbf{a},\mathbf{b}) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} w(\tau,f_d)\cdot[d(\tau,f_d) - |\mathbf{b}^H\mathbf{X}(\tau,f_d)\mathbf{a}|]^2 d\tau df_d, \tag{19}$$

where $w(\tau, f_d)$ is a weighting function that specifies which area of the CAF in the delay-Doppler plane needs to be emphasized and $(\cdot)^H$ denotes Hermitian transpose. The modulus of the desired CAF is denoted by $d(\tau, f_d)$ which is positive and real-valued, $\mathbf{a}$ and $\mathbf{b}$ are different sequences. In view of (9), the cross-ambiguity matrix of the pulse shaping functions can be written as

$$\mathbf{X}(\tau,f_d) = \begin{bmatrix} \hat{\chi}_{1,1}(\tau,f_d) & \cdots & \hat{\chi}_{1,N}(\tau,f_d) \\ \vdots & \ddots & \vdots \\ \hat{\chi}_{N,1}(\tau,f_d) & \cdots & \hat{\chi}_{N,N}(\tau,f_d) \end{bmatrix}. \tag{20}$$

where $\hat{\chi}_{i,j}(\tau,f_d)$ denotes the CAF between the i-th and j-th pulse shaping function given by (9). Furthermore, the term under the absolute value operator $|\cdot|$ in (19) represents the CAF in (8) in a more compact form as

$$\chi(\tau,f_d) = \mathbf{b}^H\mathbf{X}(\tau,f_d)\mathbf{a}. \tag{21}$$

Due to phase incoherencies, the magnitude of the ambiguity function contains all the information about a signal pertinent to system performance [20]. In order to solve the ambiguity function synthesis problem, the indirect approach introduced in

[20, 38] can be used. Accordingly, auxiliary phases are introduced to the desired ambiguity function $d(\tau, f_d)$ in (19), that is

$$\tilde{C}(\mathbf{a}, \mathbf{b}, \theta(\tau, f_d)) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} w(\tau, f_d) \cdot |d(\tau, f_d) \mathrm{e}^{j\theta(\tau, f_d)} - \mathbf{b}^H \mathbf{X}(\tau, f_d) \mathbf{a}|^2 d\tau df_d. \quad (22)$$

Introducing auxiliary phases $\theta(\tau, f_d)$ makes the integrand in (22) real and positive everywhere. The minimization problem in (22) can then be solved by fixing two arguments of $\tilde{C}(\cdot, \cdot, \cdot)$ and minimizing $\tilde{C}(\cdot, \cdot, \cdot)$ with respect to the third variable [22].

First, let us fix a pair of sequences $\mathbf{a}$ and $\mathbf{b}$ which leads to the auxiliary phase $\theta(\tau, f_d)$ being expressed as [20, 38]

$$\theta(\tau, f_d) = \arg\{\mathbf{b}^H \mathbf{X}(\tau, f_d) \mathbf{a}\}. \quad (23)$$

Second, by fixing the auxiliary phases $\theta(\tau, f_d)$ and sequence $\mathbf{b}$, the criterium $\tilde{C}(\cdot, \cdot, \cdot) \rightarrow \tilde{C}(\mathbf{a})$ can be written as [20, 38]

$$\begin{aligned} \tilde{C}(\mathbf{a}) &= \mathbf{a}^H \mathbf{D}_1 \mathbf{a} - \mathbf{a}^H \mathbf{D}_2 \mathbf{b} - \mathbf{b}^H \mathbf{D}_2^H \mathbf{a} + \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} w(\tau, f_d) |d(\tau, f_d)|^2 d\tau df_d \\ &= (\mathbf{a} - \mathbf{D}_1^{-1} \mathbf{D}_2 \mathbf{b})^H \mathbf{D}_1 (\mathbf{a} - \mathbf{D}_1^{-1} \mathbf{D}_2 \mathbf{b}) + C, \end{aligned} \quad (24)$$

where constant C does not depend on sequence $\mathbf{a}$ and therefore can be ignored. It follows from (24) that the minimizer $\mathbf{a}$ is given as

$$\mathbf{a} = \mathbf{D}_1^{-1} \mathbf{D}_2 \mathbf{b}, \quad (25)$$

where $\mathbf{D}_1 \in \mathbb{C}^{N \times N}$ and $\mathbf{D}_2 \in \mathbb{C}^{N \times N}$, respectively, are given as

$$\mathbf{D}_1 = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} w(\tau, f_d) \mathbf{X}^H(\tau, f_d) \mathbf{b} \mathbf{b}^H \mathbf{X}(\tau, f_d) d\tau df_d, \quad (26)$$

$$D_2 = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} w(\tau, f_d) d(\tau, f_d) \mathrm{e}^{\theta(\tau, f_d)} \boldsymbol{X}^H(\tau, f_d) d\tau df_d. \quad (27)$$

Third, by fixing the auxiliary phases θ (τ, f_d) and sequence $\mathbf{a}$, the criterium $\tilde{C}(\cdot,\cdot,\cdot) \rightarrow \tilde{C}(\mathbf{b})$ can be formulated as [20, 38]

$$\begin{aligned}\tilde{C}(\mathbf{b}) &= \mathbf{b}^H\mathbf{D}_3\mathbf{b} - \mathbf{b}^H\mathbf{D}_2\mathbf{a} - \mathbf{a}^H\mathbf{D}_2^H\mathbf{b} + \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} w(\tau,f_d)|d(\tau,f_d)|^2 d\tau df_d \\ &= (\mathbf{b} - \mathbf{D}_3^{-1}\mathbf{D}_2^H\mathbf{a})^H\mathbf{D}_3(\mathbf{b} - \mathbf{D}_3^{-1}\mathbf{D}_2^H\mathbf{a}) + C,\end{aligned} \tag{28}$$

where constant C does not depend on sequence $\mathbf{b}$ and therefore can be ignored. Then, in view of (28), the minimizer $\mathbf{b}$ can be obtained as

$$\mathbf{b} = \mathbf{D}_3^{-1}\mathbf{D}_2^H\mathbf{a}, \tag{29}$$

where $\mathbf{D}_3 \in \mathbb{C}^{N\times N}$ is given as

$$\mathbf{D}_3 = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} w(\tau,f_d)\mathbf{X}(\tau,f_d)\mathbf{a}\mathbf{a}^H\mathbf{X}^H(\tau,f_d)d\tau df_d. \tag{30}$$

4.1 Proposed Approach

In the proposed approach, the phases of the elements of sequence $\mathbf{a} \in \mathbb{C}^{N\times 1}$ and sequence $\mathbf{b} \in \mathbb{C}^{N\times 1}$, respectively, are denoted by column vectors of length N as

$$\phi_{\mathbf{a}} = [\phi_a(1), \phi_a(2), \ldots, \phi_a(N)]^T, \tag{31}$$

$$\phi_{\mathbf{b}} = [\phi_b(1), \phi_b(2), \ldots, \phi_b(N)]^T. \tag{32}$$

In the context of IBA, each element of the column vectors $\phi_{\mathbf{a}}$ and $\phi_{\mathbf{b}}$ in (31) and (32), respectively, is considered as a single bat generated randomly in the interval $[0, 2\pi]$. The population size (bats) is equal to the length N of the sequences. Then, the corresponding sequences $\mathbf{a}$ and $\mathbf{b}$, respectively, are given as

$$\mathbf{a} = [e^{\phi_a(1)}, e^{\phi_a(2)}, \ldots, e^{\phi_a(N)}]^T, \tag{33}$$

$$\mathbf{b} = [e^{\phi_b(1)}, e^{\phi_b(2)}, \ldots, e^{\phi_b(N)}]^T. \tag{34}$$

Given the above notion of sequence elements being bats, the pseudocode to solve the CAF synthesization problem using IBA can be formulated as in Procedure 1.

Procedure 1: Pseudocode of IBA for CAF synthesization.

---

1. Objective function $\tilde{C}(\theta(\tau, f_d), \mathbf{a}, \mathbf{b})$
2. Initialize A_i,f_i, and r_i.
3. Generate the cross-ambiguity matrix using (20).

for all NP bats

4. Generate an initial population of NP bats (solutions) to generate sequences **a** and **b** using (33) and (34), respectively, or use initially generated sequences.
5. $\theta(\tau, f_d) = \arg\{\mathbf{b}^H \mathbf{X}(\tau, f_d)\mathbf{a}\}$
6. Start with initially generated sequence in Step 4 by applying (25) to generate **a**.
7. Start with initially generated sequence in Step 4 by applying (29) to generate **b**.
8. Evaluate the objective function using (22).

end

9. Store the best objective function value.
10. Keep the current best sequences **a** and **b**.

$t = 1$

while (t < MaxGeneration) or (stop criterion)

$t = t + 1$

11. Update pulse emission rate and loudness using (17) and (18).
12. Update the velocity and frequency using (12) and (14).
13. $\theta(\tau, f_d) = \arg\{\mathbf{b}^H \mathbf{X}(\tau, f_d)\mathbf{a}\}$
14. Start with sequences generated in Step 6 and 7 to generate $\mathbf{a}'$ and $\mathbf{b}'$ by applying (25), (29), and (13).
15. **if** (rand > r)
    Start with sequences in Step 10 to generate $\mathbf{a}'$ and $\mathbf{b}'$ by performing LF using (25), (29), and (15).
    end
16. Apply problem bound constraints, if the sequences generated in Step 14 or Step 15 are outside the interval $[0, 2\pi]$.
17. Re-evaluate the objective function using (22).
18. **if** ($\tilde{C}_{\text{Step 16}} \leq \tilde{C}_{\text{Step 8}} \mid \text{rand} < A$)
    Replace the sequences in Step 6 and Step 7 with $\mathbf{a}'$ and $\mathbf{b}'$.
    $\tilde{C}_{\text{Step 8}} \leftarrow \tilde{C}_{\text{Step 17}}$
    end
19. **if** $\tilde{C}_{\text{Step 17}} \leq \tilde{C}_{\text{Step 9}}$
    Replace the sequences in Step 10 with $\mathbf{a}'$ and $\mathbf{b}'$.
    $\tilde{C}_{\text{Step 9}} \leftarrow \tilde{C}_{\text{Step 17}}$
    end

end while

4.2 Parameter Settings

Universal values of the parameters A^{t_0}, r^{t_0}, Γ, and Υ do not exist for the problems that will be discussed in Sect. 5. This is due to the fact that each problem has a different landscape and dimension. Hence, an effective set of initial values of these parameters require some experimentation. Accordingly, the initial values for these parameters were obtained from trial experiments on the optimization problems that will be considered in Sect. 5. Different initial values for loudness A and pulse emission rate r were taken in the range [0, 1] with increments of 0.1. For each optimization problem, the selected values of A^{t_0}, r^{t_0}, Γ, and Υ produced slightly different rates of convergence as each optimization problem has a different landscape.

In reality, bats increase pulse emission rate r_i and decrease loudness A_i after potential prey has been detected and their approach towards the prey has commenced. In the context of optimization, prey refers to a solution of the problem. As such, an update of loudness A_i and pulse emission rate r_i in (17) and (18), respectively, takes place in the IBA only if a new solution is found. This implies that the virtual bats are moving towards the optimal solution.

The above experimental approach was also adapted to select the values of constants Υ and Γ. The best combination of Υ and Γ was found to be $\Upsilon = \Gamma = 0.5$. The results that will be presented subsequently in Sect. 5 show that this choice of parameters seems to be appropriate for the optimization problems considered. In summary, the parameter settings listed in Table 1 are used in the simulations.

4.3 Calculation of Lévy Step Size

A CAF synthesization problem can be considered as multimodal optimization problem without any a priori information regarding the location of an optimal solution. LFs can be used to generate the random step length S of a random walk

Table 1 Parameter setting for IBA

Parameter	Value
Number of bats (population size), NP	depending on the length of the sequence to be synthesized
Number of generations, G	200
Initial loudness, A^{t_0}	0.1
Initial pulse emission rate, r^{t_0}	0.1
Constants, $\Upsilon = \Gamma$	0.5
Lévy step length, S	1.5
Minimum frequency, f_{min}	0
Maximum frequency, f_{max}	depending on the problem domain size

drawn from a Lévy distribution. The choice of α in (16) determines the probability of obtaining a Lévy random number in the tail of the Lévy distribution. Given that each optimization problem is unique, i.e., has different dimension and landscape, the task of choosing a favorable value of α that generates a suitable step length S becomes difficult. In particular, the search ability of the algorithm may be severely hampered, if an improper value of α is used to generate S.

Given the complex nature of the CAF synthesization problem, it seems that a universal value of α required to generate S for guiding virtual bats in IBA without getting trapped in a local minimum does not exist. Therefore, it is appropriate to carry out a series of experiments in order to find a suitable value of α. For this purpose, four values of α = 1.3, 1.4, 1.5, and 1.6 were selected. For each of these values, 10 independent trials for a fixed number of iterations were performed to minimize (22) for the problems that will be discussed in Sect. 5. It turned out that α = 1.5 produces the best value of criterium (22) in average over the number of trails. Therefore, this value has been used to generate the random step length S for all problems considered in Sect. 5.

4.4 Selection of Scaling Factor

The parameter γ in (15) determines how far the virtual bats in IBA can travel in the search space. An excessively large value of γ causes new solutions to jump outside of the feasible search space and even to fly off to far regions. On the other hand, the search is confined to a rather narrow region, if γ is too small. In the former case, the LF becomes too aggressive, whereas, in the latter case, the LF is not efficient. Therefore, some sort of strategy is needed to scale step length S such that an efficient search process is maintained. In order to avoid the particles/bats flying too far, a small value of parameter γ can be more efficient [25, 26]. A small value of γ may apply for unimodal problems. We hypothesize that the location of an optimal solution to a multimodal problem such as CAF synthesization is not known. As such, selecting a small value of γ will hinder the search process.

Therefore, in order to select an appropriate value of γ, we have adopted the experimental approach described in Sect. 4.3 and conducted a series of trials with different values of γ. We have conducted 10 independent trails for a fixed number of iterations that were performed to minimize (22) for Example 2 in Sect. 5 using $\gamma = 0.01, 0.05, 0.1, 0.5, 0.7$ and 0.9. The best peak-to-average power ratio (PAR) for each of these values produced by IBA for each run was recorded. It was found that γ = 0.05 produces the best PAR and hence was subsequently used as a basis for the examples presented in Sect. 5.

5 Numerical Results

Let us consider sequences of length $N = 50$ for Examples 1 and 2, sequence length $N = 53$ for Example 3 and sequence length $N = 31$ for Example 4. Each element of the considered sequences corresponds to the phase-coded amplitude of a rectangular pulse shape function of duration T_c. Thus, the duration of a sequence is given as $T = N \cdot T_c$. Furthermore, τ denotes the delay by which a transmitted signal is returned from a target and f_d denotes the Doppler frequency induced by a moving target. In the sequel, we utilize normalized delay τ/T_c and normalized Doppler frequency $f_d \cdot T$, respectively. In what follows, we illustrate by way of four examples that IBA is able to jointly design sequences **a** and **b** such that a desired CAF is synthesized.

5.1 Example 1

In this example, we aim at synthesizing a CAF with a diagonal ridge while being zero elsewhere. This type of CAF is desirable when a filter bank is too expensive to cope with different Doppler frequencies and tolerance to Doppler shifts is needed. The weighting function is $w(\tau, f_d) = 1$ for all (τ, f_d) in (19) and the sequence **a** has constant modulus, i.e., each element of **a** takes on the value of one and hence PAR = 1. The desired CAF is shown in Fig. 1 and the corresponding synthesized CAF obtained by using IBA is shown in Fig. 2. As can be seen from Fig. 2, the CAF synthesized by IBA approximates the desired CAF in Fig. 1.

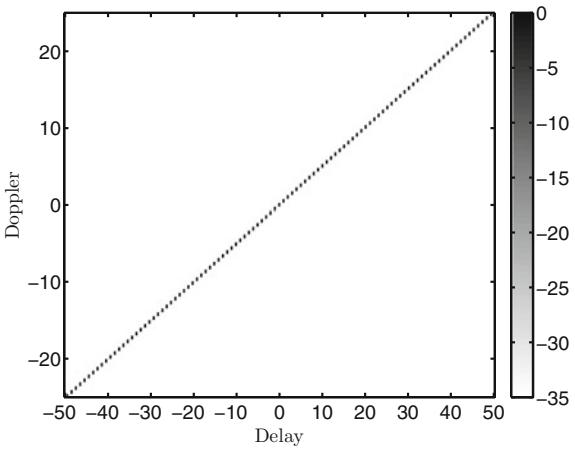


Fig. 1 Desired CAF with diagonal ridge

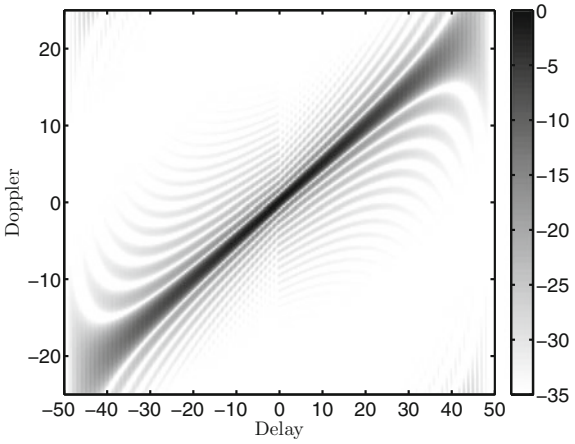


Fig. 2 Synthesized CAF using IBA for $\gamma = 0.05$ (PAR = 1)

5.2 Example 2

The synthesization of an ideal thumbtack CAF, i.e., narrow peak at the origin and zero sidelobes in the rest of the delay-Doppler plane is not possible due to the volume property of CAFs. Therefore, in this example, we aim at synthesizing a CAF with a clear area in and around a large neighbourhood of the origin using the following CAF modulus:

$$d(\tau, f_d) = \begin{cases} N, & \text{for } (\tau, f_d) = (0,0), \\ 0, & \text{elsewhere}, \end{cases}$$

and weighting function

$$w(\tau, f_d) = \begin{cases} 1, & \text{for } (\tau, f_d) \in \Omega_{ds} \backslash \Omega_{\sim ds}, \\ 0, & \text{elsewhere}, \end{cases}$$

where $\Omega_{ds} = \left\{ [-10T_c, 10T_c] \times \left[-\frac{2}{T_c}, \frac{2}{T_c} \right] \right\}$ is the selected region of interest of the synthesized CAF. In order to compensate for sharp changes in the desired CAF $d(\tau, f_d)$ near the origin, the area of the main lobe $\Omega_{\sim ds} = \{[-T_c, T_c] \backslash \{0\} \times [-\frac{1}{T_c}, \frac{1}{T_c}] \backslash \{0\}\}$ near the origin has to be excluded [22]. Recall that the Doppler shift f_d induced on the signal in practice is often much smaller compared to the bandwidth of the transmitted signal. Therefore, the weighting function $w(\tau, f_d)$ outside the maximum induced Doppler shift f_d can be set to zero.

The need of this type of CAF arises in applications such a geolocation of signals, where the CAF is used to calculate the time of difference of arrival and frequency difference of arrival of the emitted signal using two receivers [39]. The two collector architecture offers the opportunity to compare the reception of a likely similar

radar pulse using cross-correlation concepts with respect to delay. Thus, one collector will see the radar pulse as $a(t)$ and the other collector will see it as $b(t + \tau)$. Also, it is assumed that one collector is moving with some relative velocity to the other collector which supports measuring the frequency of the received pulse at slightly different frequencies [39].

Furthermore, a CAF with a clear area around a large neighbourhood of the origin also arises in situations, when it is not possible to design a sequence or set of sequences that yield zero sidelobes over the entire delay-Doppler plane. Therefore, it is desirable to design a reference waveform or equivalent filter at the radar receiver end. Such a receiver, is called an optimum receiver [1] in which the internal or reference waveform (or equivalent filter) is deliberately mismatched compared to the transmitted waveform in order to reduce the sidelobes in the delay-Doppler plane.

Figure 3 shows the CAF of sequences **a** and **b** generated by IBA with $\gamma = 0.05$. The desired sidelobe free area can be observed within the rectangular area close and around the origin. The sidelobe free region in Fig. 3a is due to the fact that the amplitude of the generated sequences **a** or **b** is not constrained. This may result in sequences with relatively high PAR and relatively low sidelobe levels. The PAR of sequences **a** and **b** for the CAF shown in Fig. 3a were found to be $PAR_{\mathbf{a}} = 3.9$ and $PAR_{\mathbf{b}} = 7.2$, respectively. It is noted that low sidelobe levels are desirable in radar applications to avoid masking of main peaks of secondary targets, even if the targets are well separated. Moreover, in case of a multiple target environment, the sum of all sidelobes may build up to a level sufficient to mask even relatively strong targets.

The widespread use of solid state power amplifiers and digitization can have a significant impact on the overall performance of radar, sonar, and communication systems. For example, the transmission of a signal or a waveform of arbitrary amplitude is not possible due to the limitations of power amplifiers and analog-to-digital converters. As a result, it is often desirable that transmit signals or waveforms have a constant amplitude or a low PAR. One of the possibilities that allows the consideration of waveforms with variable amplitude is to work with a pair of waveforms, i.e., the transmitted signal of constant amplitude and the reference signal of arbitrary amplitude that is used during signal processing at the receiver [40]. Therefore, to constrain the PAR of a transmit waveform with PAR = 1, the following additional operation may be employed in the IBA algorithm in Procedure 1 after Step 14 (see also [22]):

$$s_n \leftarrow \exp[j \arg(s_n)]. \tag{35}$$

However, inducing such a constraint further complicates the design of waveforms with prescribed ambiguity surfaces. Using (35), somewhat higher sidelobes can be observed in the results shown in Fig. 3b. The normalized zero-Doppler cut through the CAF for the unconstrained design (PAR > 1) and constrained design (PAR = 1) are shown in Fig. 4.

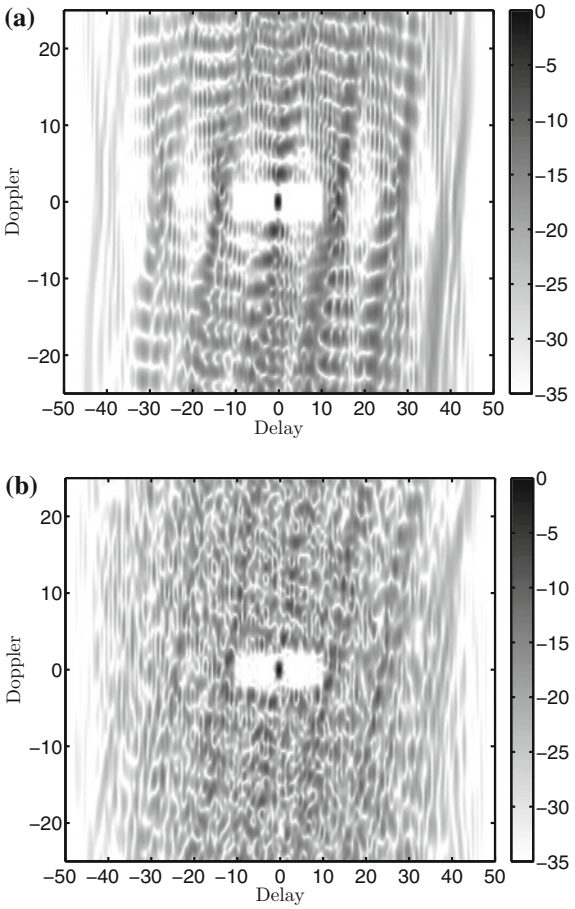


Fig. 3 CAF synthesization without and with PAR constraint (35) for $\gamma = 0.05$: **a** Synthesized CAF of random sequences of length N = 50 (PAR = 3.9), **b** Synthesized CAF of random sequences of length N = 50 (PAR = 1)

5.3 Example 3

In this example, we aim at synthesizing a CAF for Björck sequences of length $N = 53$. In particular, Björck sequences of length $N = P$, where P is a prime number and $P \equiv 1(\text{mod}4)$, are defined as

$$\mathbf{B}(k) = \exp\left(j2\pi\theta\left(\frac{k}{P}\right)\right), \quad \theta = \arccos\left(\frac{1}{1+\sqrt{P}}\right), \tag{36}$$

where $\left(\frac{k}{P}\right)$ denotes the Legendre symbol which is defined as

$$\left(\frac{k}{P}\right) = \begin{cases} 1, & \text{if } \mathrm{k} \equiv 0 \pmod{\mathrm{P}}, \\ 1, & \text{if } \mathrm{k} \equiv m^2 \pmod{\mathrm{P}} \text{ for } \quad \mathrm{m} \in \mathbb{Z}, \\ -1, & \text{if } \mathrm{k} \not\equiv m^2 \pmod{\mathrm{P}} \text{ for } \quad \mathrm{m} \in \mathbb{Z}. \end{cases}$$

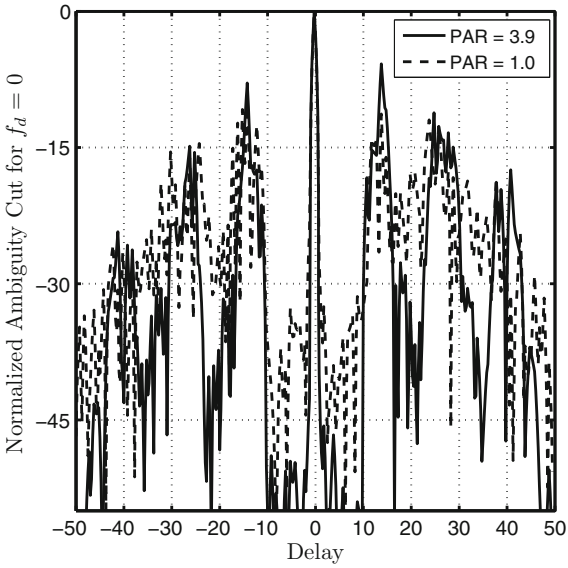


Fig. 4 Normalized zero-Doppler cut of the CAFs of Fig. 3a, b without and with constraint (35), respectively

The synthesized CAFs for the case of Björck sequences which approximates a desired thumbtack CAF without and with using constraint (35) are shown in Fig. 5a, b, respectively. The normalized zero-Doppler cuts through the CAFs for the unconstrained design (PAR > 1) and the constrained design (PAR = 1) are shown in Fig. 6. A sidelobe level below −45 dB respective −30 dB can be observed for these cases.

5.4 Example 4

Finally, we synthesize a CAF for the case of Oppermann sequences [41] of length $N = 31$. The phase $\varphi_k(i)$ of the i-th element $u_k(i)$ of the k-th Oppermann sequence $\mathbf{u}_k = [u_k(0), u_k(1), \ldots, u_k(N-1)]$ of length N is defined as

$$\varphi_k(i) = \frac{\pi}{N}[k^m(i+1)^p + (i+1)^n + k(i+1)N], \tag{37}$$

where $1 \leq k \leq N-1$, $0 \leq i \leq N-1$ and integer k is relatively prime to the length N. The parameters m, n and p in (37) take on real values and define a family of Oppermann codes.

The CAF for the case of Oppermann sequences with parameters m, $p = 1$ and $n = 3$ is shown in Fig. 7. The synthesized CAF of these Oppermann sequences is

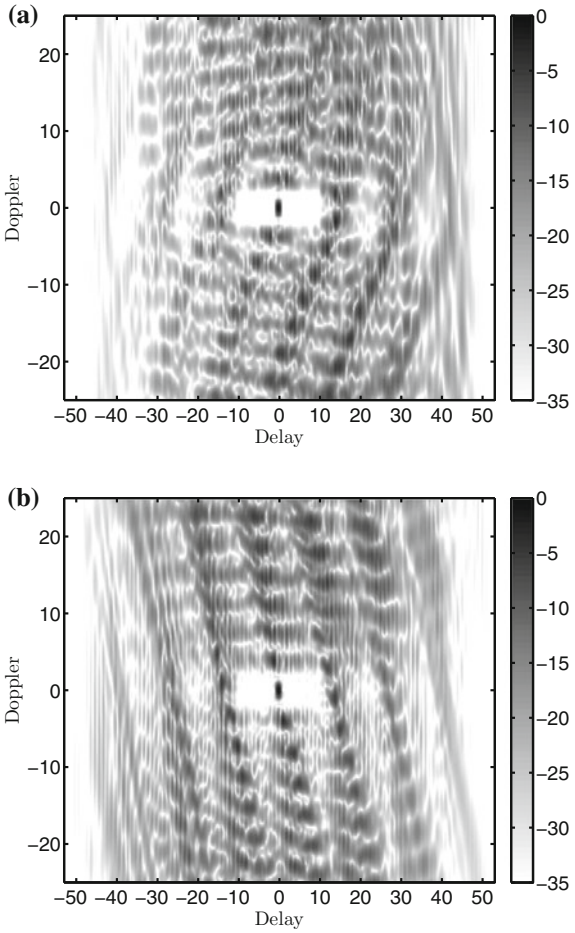


Fig. 5 CAF synthesization without and with (35) for $\gamma = 0.05$: **a** Synthesized CAF for Björck sequence of length N = 53 (PAR = 4.9), **b** Synthesized CAF for Björck sequence of length N = 53 (PAR = 1)

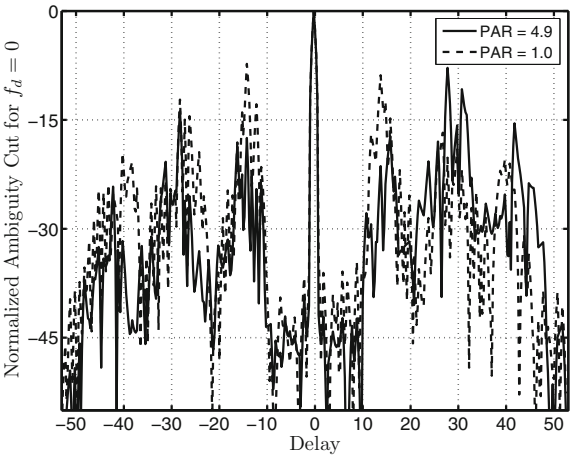


Fig. 6 Normalized zero-Doppler cut of the CAFs shown in Fig. 5a, b without and with constraint (35), respectively

shown in Fig. 8, which approximates the desired CAF with a sidelobe free area around the neighbourhood of the origin. Relatively low sidelobe levels can be observed within the rectangular area close and around the origin with improved delay-Doppler characteristics compared to the CAF of the original sequence that is shown in Fig. 7. The zero-Doppler cut of the synthesized Oppermann sequence is shown in Fig. 9 which exhibits a low sidelobe level with respect to delay compared to the original Oppermann sequences.

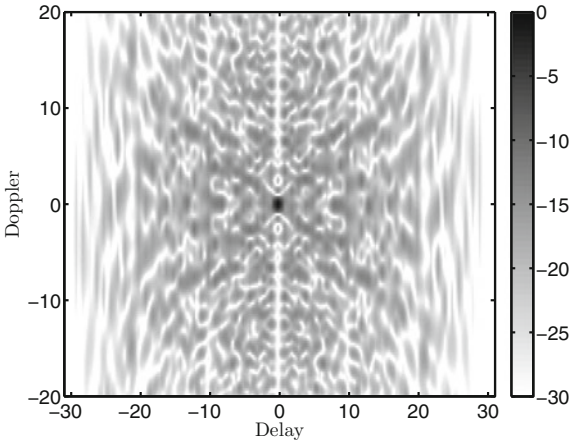


Fig. 7 Synthesized CAF of Oppermann sequences with parameters $m, p = 1$, and $n = 3$

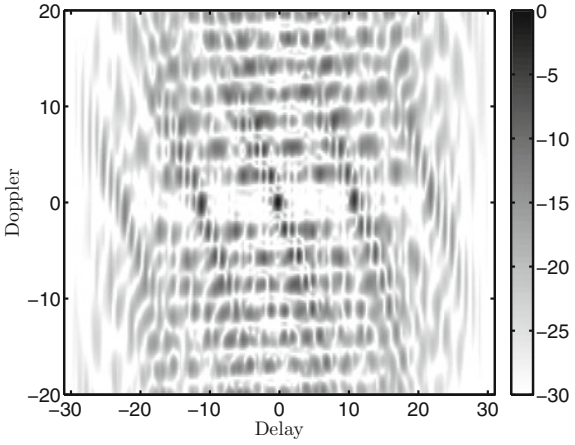


Fig. 8 Synthesized CAF of Oppermann sequences with $\gamma = 0.05$ and using constraint (35)

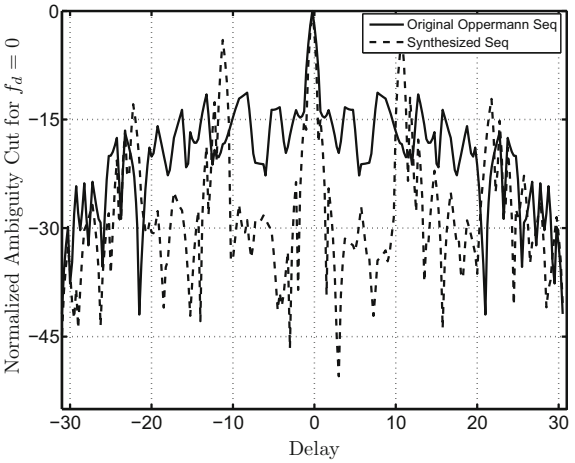


Fig. 9 Normalized zero-Doppler cuts through the CAFs shown in Figs. 7, 8

6 Conclusions

In this chapter, the problem of synthesizing CAFs using a metaheuristic approach based on the echolocation of bats has been addressed. The fundamental problem in this context is to minimize the integrated square error between a desired CAF and a synthesized CAF. In particular, the IBA has been combined with a cyclic approach to solve this problem. By using four examples, we have shown that the approach based on echolocation of bats can indeed synthesize CAFs that approximate CAF surfaces having a diagonal ridge and zero value elsewhere as well as CAF surfaces with a clear area around the origin. Our results indicate that the proposed approach is a promising technique for synthesizing CAFs. Further research will focus on more extensive studies of how to synthesize other complex functions and waveforms.

References

1. Van Tress, H.L.: Optimum signal design and processing for reverberation-limited environment. IEEE Trans. Military Electron. **9**(3), 212–229 (1965)
2. Stoica, P., Li, J., Xue, M.: Transmit codes and receive filters for radar. IEEE Signal Process. Mag. **25**(6), 94–109 (2008)
3. Blunt, S.D., Gerlach, K.: Adaptive pulse compression via MMSE estimation. IEEE Trans Aerosp Electron. Syst. **42**(2), 572–584 (2006)
4. Delong Jr, D.F., Hofstetter, E.M.: The design of clutter-resistant radar waveforms with limited dynamic range. IEEE Trans. Inf. Theor. **15**(3), 376–385 (1967)
5. Kay, S.: Optimal signal design for detection of Gaussian point targets in stationary Gaussian clutter/reverberation. IEEE J. Sel. Top. Sign. Process. **1**(1), 31–41 (2007)
6. Key, E.L.: A method of sidelobe reduction in coded pulse waveform. Tech. Report 209, M.I.T Lincoln Lab., Lexington, Mass (1959)

7. Rummler, W.D.: A technique for improving the clutter performance of coherent pulse train signals. IEEE Trans Aerosp. Electron. Syst. **3**(6), 898–906 (1967)
8. Stutt, C., Spafford, L.J.: A best mismatched filter response for radar clutter discrimination. IEEE Trans. Inf. Theor. **14**(2), 280–287 (1968)
9. Urkowitz, H.: Some high-velocity clutter effects in matched and mismatched receivers. IEEE Trans Aerosp. Electron. Syst. **4**(3), 481–485 (1968)
10. Spafford, L.J.: Optimum radar signal processing in clutter. IEEE Trans. Inf. Theor. **14**(5), 734–743 (1968)
11. Woodward, P.M.: Probability and information theory with applications to radar. Pergamon Press (1953). Reprint: Artech House, London (1980)
12. Stoica, P., He, H., Li, J.: New algorithms for designing unimodular sequences with good correlation properties. IEEE Trans. Signal Process. **57**(4), 1415–1425 (2009)
13. Wilcox, C.H.: The synthesis problem for radar ambiguity functions. MRC Tech. Summary Report 157, US Army, University of Wisconsin, Madison, Wisconsin, USA (1960). Reprint: Radar and Sonar, Part 1, The IMA volumes in mathematics and its applications. Springer (1991)
14. Gladkova, I., Chebanov, D.: On the synthesis problem for a waveform having a nearly ideal ambiguity function. In: International Conference on Radar Systems. Toulouse, France (2004)
15. Stein, S.: Algorithms for ambiguity function processing. IEEE Trans. Acoust. Speech Signal Process. **29**(3), 588–599 (1981)
16. Sharama, R.: Analysis of MIMO radar ambiguity function and implications on clear region. In: IEEE International Radar Conference. Washington DC, USA (2010)
17. Blau, W.: Synthesis of ambiguity functions for prescribed responses. IEEE Trans. Aerosp. Electron. Syst. **3**(4), 656–663 (1967)
18. Rihaczek, A.W., Mitchell, R.L.: Radar waveforms for suppression of extended clutter. IEEE Trans. Aerosp. Electron. Syst. **3**(3), 510–517 (1967)
19. Siebert, W.: A Radar Detection Philosophy. IRE Trans. Inf. Theor. **2**(3), 204–221 (1956)
20. Sussman, S.: Least-square synthesis of the radar ambiguity function. IEEE Trans. Inf. Theor. **8** (3), 246–254 (1962)
21. Wolf, J.D., Lee, G.M., Suyo, C.E.: Radar waveform synthesis by mean-square optimization techniques. IEEE Trans Aerosp. Electron. Syst. **5**(4), 611–619 (1969)
22. He, H., Stoica, P., Li, J.: On synthesizing cross ambiguity function. In: IEEE International Conference on Acoustics, Speech and Signal Processing. Prague, Czech Republic (2011)
23. Jamil, M., Zepernick, H.-J., Yang, X-S.: Lévy flight based cuckoo search algorithm for synthesizing cross-ambiguity functions. In: IEEE Military Communications Conference, pp. 823–828, San Diego, USA (2013)
24. Jamil, M., Yang, X-S., Zepernick, H.-J.: Improved Bat-inspired metaheuristic algorithm with Lévy flights for global optimization problems. J. Appl. Softw. Comput.—Under Revision
25. Yang, X-S., Deb, S.: Cuckoo search via Lévy flights. Congress on Nature and Biological Inspired Computing, pp. 210–214, Coimbatore, India (2009)
26. Yang, X-S.: Firefly algorithm, Lévy flights and global optimization. In: Bramer, M., Ellis, R., Petridis, M. (eds.) Research and Development in Intelligent Systems XXVI, pp. 209–218, Springer, Berlin (2010)
27. Yang, X-S.: A new metaheuristic bat-inspired algorithm. In: Gonzalez et. al. J.R. (eds.) Nature Inspired Cooperative Strategies for Optimization, Studies in Computational Intelligence, pp. 65–74, Springer, Berlin (2010)
28. Gutowski, M.: Lévy flights as an underlying mechanism for global optimization algorithms. In: Proceedings of National Conference on Evolutionary Computation and Global Optimization. Jastrzębia Góra, Poland (2001)
29. Austin, D., Bowen, W.D., McMillan, J.I.: Intraspecific variation in movement patterns: modelling individual behaviour in a large marine predator. Oikos **105**(1), 15–30 (2004)
30. Bartumeus, F., Peters, F., Pueyo, S., Marrase, C., Catalan, J.: Helical Lévy walks: adjusting searching statistics to resource availability in microzooplankton. Proc. Nat. Acad. Sci. USA **100**(22), 12771–12775 (2003)

31. Humphries, N.E., Querioz, N., Dyer, J.R.M., Pade, N.G., Musyl, M.K., Schaefer, K.M., Fuller, D.W., Brunnschweiler, J.M., Doyle, T.K., Houghton, J.D.R., Hays, G.C., Jones, C.S., Noble, L. R., Wearmouth, V.J., Southall, E.J., Sims, D.W.: Environmental context explains Lévy and Brownian movement patterns of marine predators. Nature **451**(7301), 1066–1069 (2010)
32. Mårell, A.J., Ball, P., Hofgraad, A.: A foraging and movement paths of female reindeer: insights from fractal analysis, correlated random walks and Lévy flights. Can. J. Zool. **80**(5), 854–865 (2002)
33. Viswanathan, G.M., Afanasyev, V., Buldyrev, S.V., Murphy, E.J., Prince, P.A., Stanley, H.E.: Lévy flight search patterns of wandering albatrosses. Nature **381**(6581), 413–415 (1996)
34. Viswanathan, G.M.: Fish in Lévy-flight foraging. Nature **465**(7301), 1018–1019 (2010)
35. Glover, F.: Tabu search—Part I. ORSA J. Comput. **1**(3), 190–206 (1989)
36. Glover, F.: Tabu search—Part II. ORSA J. Comput. **2**(1), 4–32 (1990)
37. Yang, X.-S., He, X.: Bat algorithm review and applications. Int. J. Bio-Inspired Comput. **5**(4), 141–149 (2013)
38. Fienup, J.R.: Phase retrieval algorithms: a comparison. Appl. Opt. **21**(15), 2758–2769 (1982)
39. Overfield, J., Biskaduros, Z., Buehrer, R.M.: Geolocation of MIMO signals using the cross ambiguity function and TDOA/FDOA. In: IEEE International Conference on Communications, pp. 3648–3653, Ottawa, Canada (2012)
40. Levanon, N., Mozeson, E.: Radar Signals. Wiley, New York (2004)
41. Oppermann, I., Vucetic, B.S.: Complex spreading sequences with a wide range of correlation properties. IEEE Trans. Commun. **45**(3), 365–375 (1997)

Sustainable Building Design: A Review on Recent Metaheuristic Methods

Somayeh Asadi and Zong Woo Geem

Abstract A notable portion of the total primary energy is consumed by today's buildings in developed countries. In recent years, decision makers and planners are facing increased pressure to respond more effectively to a number of energy-related issues and conflicts. Therefore, this article strives to make a technical review of all relevant research applying simulation-based optimization methods to sustainable building design problems. A summary of the application of common heuristic and meta-heuristic methods to different fields of sustainable building design is given.

Keywords Sustainable · Residential and commercial building · Design · Optimization · Heuristic methods

1 Introduction

1.1 World Energy Consumption

The rapidly increasing world energy consumption has raised concerns about supply complications, exhaustion of energy resources, and severe environmental impacts. The International Energy Agency (IEA) has provided informative data on energy consumption trends. Over the last years, primary energy has increased by 83 % from 1980 to 2011 and CO_2 emissions by 85 %, with an average annual increase of 2.67 and 2.74 % respectively (refer to Fig. 1) [1]. According to IEA, world energy consumption will increase from 524 quadrillion Btu in 2010 to 630 quadrillion Btu

S. Asadi
Department of Architectural Engineering, Pennsylvania State University, 213 Engineering Unit A, University Park, PA 16802, USA

Z.W. Geem (✉)
Department of Energy IT, Gachon University, 1342 Seongnam Daero, Seongnam 461-701, South Korea
e-mail: geem@gachon.ac.kr

X.-S. Yang (ed.), *Recent Advances in Swarm Intelligence and Evolutionary Computation*, Studies in Computational Intelligence 585, DOI 10.1007/978-3-319-13826-8_11

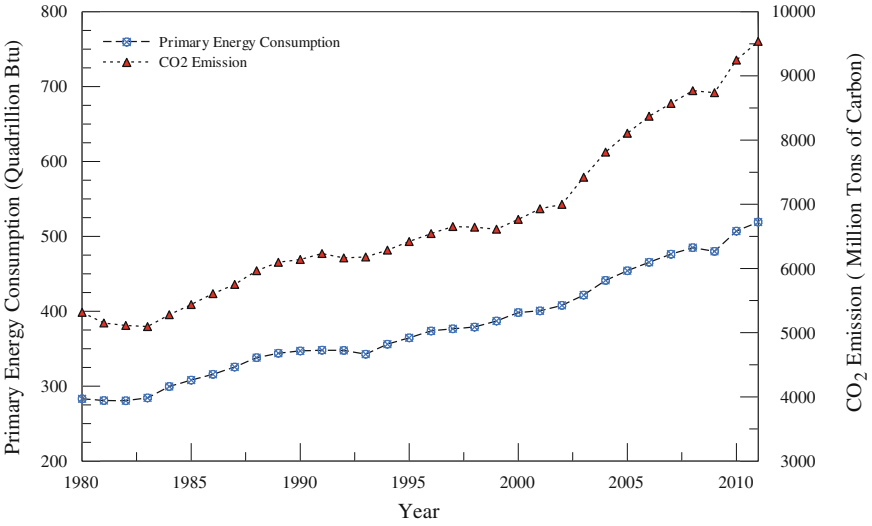


Fig. 1 World energy consumption and CO_2 emission

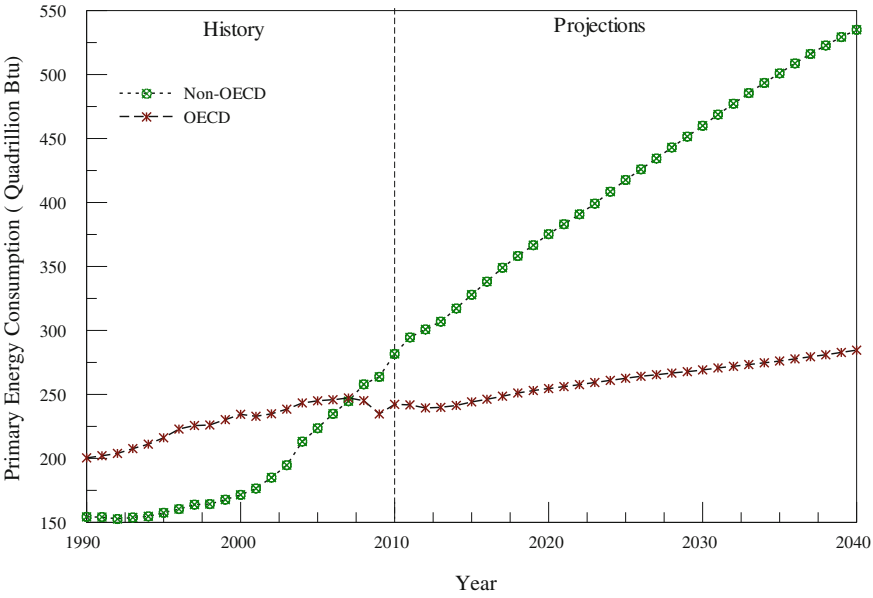


Fig. 2 World energy consumption prediction

in 2020 and 820 quadrillion Btu in 2040, a 30-year increase of 56 % (Fig. 2). More than 85 % of the increase in global energy demand from 2010 to 2040 takes place among the developing nations outside the Organization for Economic Cooperation

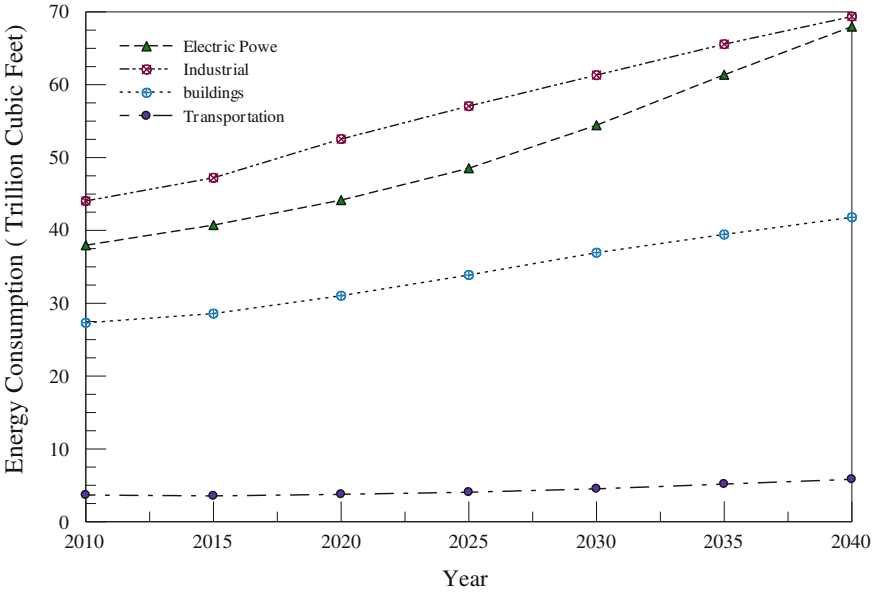


Fig. 3 World energy consumption in different sectors

and Development (non-OECD), driven by strong economic growth and expanding populations. This discrepancy is due to the different pattern of consumption in OECD and non-OECD countries in which the former have more mature consumption pattern and well established electricity markets than the latter [1].

On the other hand, the building sector plays a significant role regarding energy consumption. As it can be seen in Fig. 3, building sector is one of the largest energy consumers in the world. According to the World Business Council for Sustainable Development, buildings consume 40 % of the total energy [2]. Aside from energy consumption, buildings produce Greenhouse Gas emission (GHG) which results in global warming. It was also anticipated that in 2035, the carbon emission of buildings across the world will get to 42.4 billion tons which will increase by 43 % compared to 2007 [3]. Furthermore, the renovation, refurbishment, and retrofitting of building consume natural resources and energy and produce GHG emission as well as other pollutants [4].

Recently, several studies have been carried out to enhance energy efficiency and decrease energy consumption and GHG emission. It is obvious that energy-efficient design methods provide considerable benefits to the end user. A building which is designed based on energy-saving measures decreases building life cycle costs due to lower energy consumption and CO_2 emission. Lower CO_2 emissions into the atmosphere during the building's life cycle benefits society as well.

1.2 Sustainable Building Design

Sustainable buildings [5] are defined as buildings with minimum negative environmental impacts. Sustainable buildings may be defined as building practices which struggle for essential quality (including economic, social, and environmental performance). Therefore, the reasonable use of natural resources and suitable management of the building stock will save rarer resources, decrease energy use, and improve environmental quality.

To design a sustainable building, the whole life cycle of buildings from raw material extraction to building end of life should be considered. Therefore, sustainable building design is the contemplative incorporation of architecture with electrical, mechanical, and structural engineering resources. As sustainability progressively influences upon the lives of corporations, individuals and wider society, the chances for accountable and holistic thinking is also growing [6]. The design of sustainable buildings is not simple since all buildings are distinct, there are no specific models, and they must attain high levels of performance with low cost. In addition, there are many physical procedures that result in conflicting objectives in design of sustainable buildings. Therefore, these challenges have made them valuable to utilize computational methods of design optimization [7, 8].

In recent years, several solution approaches such as gradient-based optimization methods and meta-heuristic optimization algorithms have been suggested to design sustainable buildings. The gradient-based optimization methods require derivative information of the mathematical equations and a good starting point for decision variables; therefore, their applications may be both problematic and impracticable. This makes meta-heuristic optimization algorithms as one of the proper methods to design sustainable buildings and optimize their performance. Figure 4 shows the growing trend of international optimization studies (indexed by SciVerse Scopus of Elsevier) in the field of building science. It can be seen that the number of optimization papers has increased sharply since the year 2005. This indicates a great interest on optimization techniques among building research communities [9]. Therefore, this study seeks to critically review the green building related studies in order to emphasize the state of art and future needs in this field. It also provides a systematic review of existing body of knowledge. Such systematic review plays a significant role to not only classify the common research streams but also determine the future research trends.

2 Overview of Existing Meta-Heuristic Algorithms

2.1 Classifications

Heuristic or metaheuristic is considered as one of the main approaches of problem-solving by trial and error. 'Heuristic' means to 'find' or 'search' by trials and errors. The meta-heuristics algorithms contain two main components including intensification and

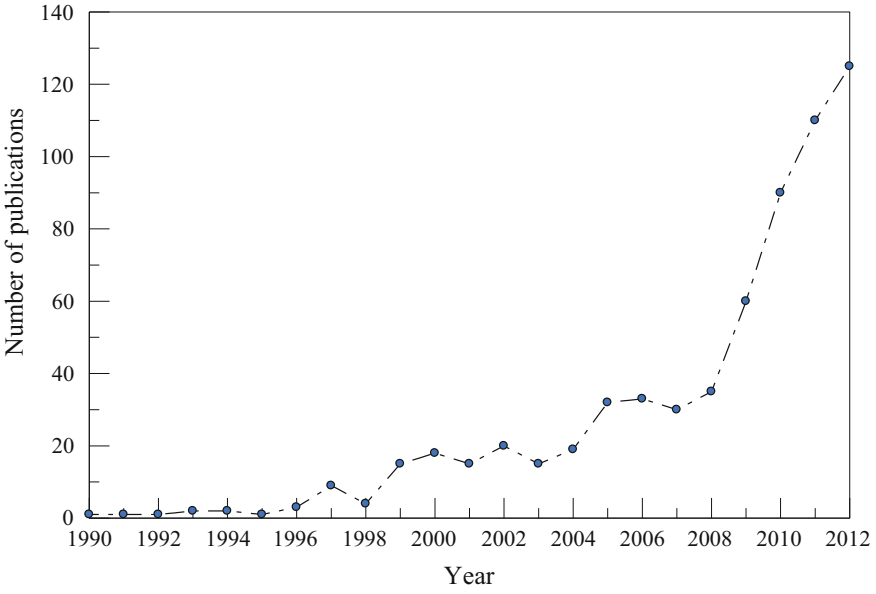


Fig. 4 Trend of number of optimization studies in building science (after [9])

diversification which was defined by Tabu [10, 11]. In order to develop an efficient and effective algorithm, it must be able to make a wide range of solutions while it strengthens its search around the neighborhood of an optimal or nearly optimal solution. Therefore, every section of the search area must be available. Diversification creates diverse solutions which is able to examine the search area on the global scale. However, intensification only concentrates on the search in a local region by exploiting the information that a current good solution is found in this region [10–15]. In a case that the intensification is too robust, only a portion of solution area might be considered which may result in a risk of being stuck in a local optimum. In the meantime, if the diversification is too robust, the algorithms converge very gradually because solutions jump around some potentially optimal solutions. Usually, the solutions start with random solutions and slowly decrease their diversification while raising their intensification simultaneously. Therefore, in order to develop an effective meta-heuristic algorithm, a proper balance between intensification and diversification is required [12]. Meta-heuristic models are usually based on the assumption that a methodology is chosen and formed according to the minimization of some objective function. Therefore, randomization provides a good way to move away from local search to the search on the global scale. Almost all meta-heuristic algorithms intend to be suitable for global optimization [16].

There are various ways to classify the meta-heuristics algorithms including population-based and trajectory-based. For example, genetic algorithms are population-based since they use a set of strings, so is the particle swarm optimization (PSO) which uses multiple agents [17]. Instead, simulated annealing uses a single

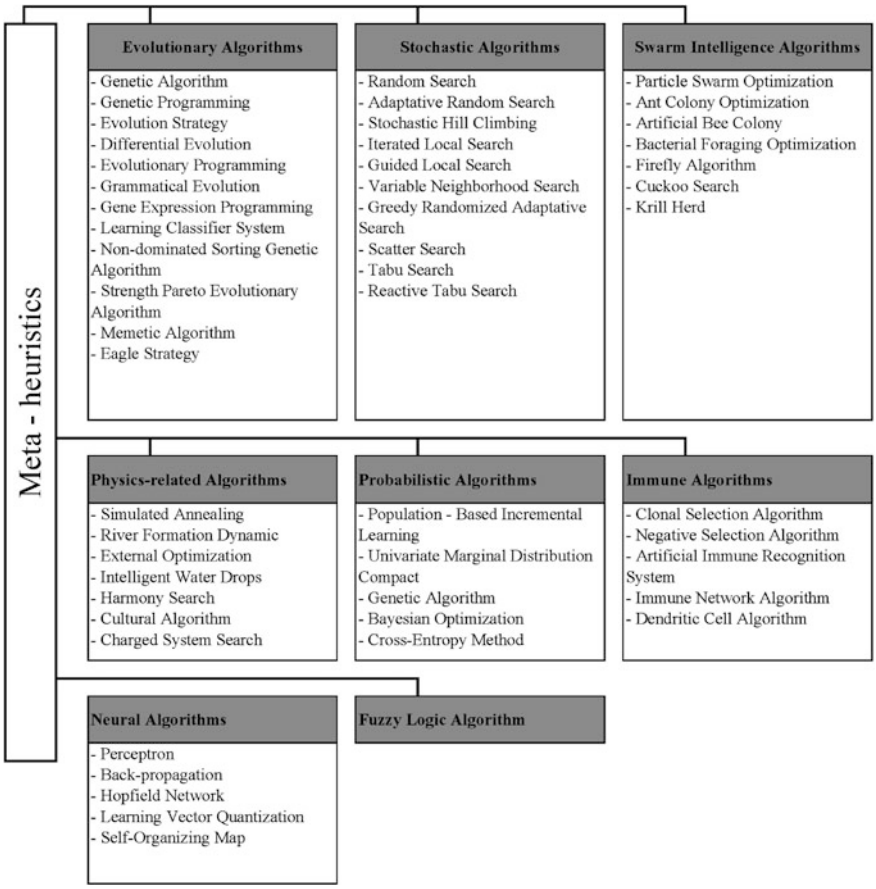


Fig. 5 Classification of meta-heuristic algorithms (after [19])

agent or solution which moves through the search area in a piecewise style [18]. Figure 5 shows the classification of meta-heuristic algorithms. However, it is difficult to decide which type of method is more effective under appropriate conditions.

2.2 Application of Some Selected Algorithms to Design Sustainable Buildings

Providing a general rule for the algorithm selection to design sustainable buildings is generally infeasible due to complexity and diversity of their design. The meta-heuristic algorithms such as harmony search, genetic algorithm, particle swarm

optimization, ant colony optimization, etc. were the most frequently used methods in building performance optimization. Such stochastic algorithms cannot promise that the best solution will be reached after a finite number of iterations, but they are utilized to find the best solutions in a rational amount of time [9].

2.2.1 Harmony Search

Recently, the meta-heuristic Harmony Search (HS) optimization algorithm which is conceived using the musical process of searching for the perfect state of harmony was developed by Geem et al. [20]. HS is based on the musical process of searching for a perfect state of harmony, such as jazz improvisation. Jazz improvisation searches for musically pleasing harmony as identified by an aesthetic standard. The notes and the pitches of each musical instrument regulate the aesthetic quality, just as the objective function value is determined by the values assigned to design variables. Practice improves the quality of harmony, similar to the solution quality that improves with iteration [20].

HS algorithm is simple in concept, involves only a few parameters, and can be easily implemented. It has been successful in a wide variety of real-world and benchmark optimization problems [20–23], presenting several advantages with respect to traditional optimization techniques such as the following [22]: (a) it imposes fewer mathematical requirements and does not require initial value settings of the design variables, (b) it uses stochastic random search so calculus-based derivative information is also unnecessary, and (c) it generates a new solution vector after considering all of the existing vectors. These features increase the flexibility of the HS algorithm and result in better solutions. The HS algorithm performs well for global searching; however, since it does not use gradient information, it may take a relatively long time to converge to a local optimum.

In Table 1, the HS algorithm is briefly described in a pseudo-code form. In the HS optimization methodology, first, the initial values of design variables are randomly assigned within the defined range of variables. Then a simulation is done to evaluate the objective function. Next, the HS algorithm, based on the obtained results, sets new values for design variables and another simulation is performed to evaluate the objectives of the new design. The new values of the design variables can be chosen either by random or using the best obtained values which are already stored in harmony memory of the algorithm. In a case that the new solution is better than the worst solution available in the harmony memory, the worst solution is replaced by the new solution. As optimization process proceeds, little by little, the solutions stored in harmony memory become better and approach the optimum solution. The process is continued until a pre-specified maximum number of iterations for the HS algorithm is reached [24].

The HS algorithm has been recently used to design energy efficient and low emission building and several engineering optimization problems. Originally, applications where HS was first evaluated as an effective meta-heuristic method, focused mainly on the design of water distribution networks [20], benchmark

Table 1 The pseudo-code of the HS algorithm [24]

```
procedure HS

 Initiate_parameters()
 Initialize_HM()

 do{
   for (I = 1 to number of decision variables N)
    if (rand() < P_HMCR)    /* (memory consideration) */
      X[I] will be randomly chosen from the HM
      If (rand()< P_PAR)  /* (pitch adjustment) */
        X[I] = X[I] ± Δ
      end if
    else  /* (random selection) */
      X[I] = X_LB[I] + rand() × (X_UB[I] - X_LB[I])
    end if
   end for

 /* evaluate the fitness of each vector */
  fitness_X = evaluate_fitness(X)
  update_memory(X,fitness_X)  /* update HM if applicable */

 }while(not_termination)

 print_final_solution()

end procedure
```

optimization [25], structural design [26], and vehicle routing problems [27, 28]. Since then, the activity around this algorithm has increased sharply, spanning its applicability to a very heterogeneous portfolio of application scenarios such as traveling salesman problem [29], optimization of the river flood model [30], optimum design of water distribution network [20], optimum design of truss structures [26, 31, 32], design of energy efficient buildings [24], and the simultaneous determination of aquifer parameters and zone structures by an inverse solution algorithm [33]. In order to get a clear overview of the classification and analysis presented in different sections, Fig. 6 depicts the current approximate distribution of areas of HS application as measured by the number of publications of each particular topic.

Fesanghary et al. [24] applied HS algorithm to investigate the trade-off between energy use and life-cycle cost (LCC) by varying building constructions. A multi-objective optimization model based on HS algorithm was presented in this study. The objective function of this study was to minimize the LCC and carbon dioxide

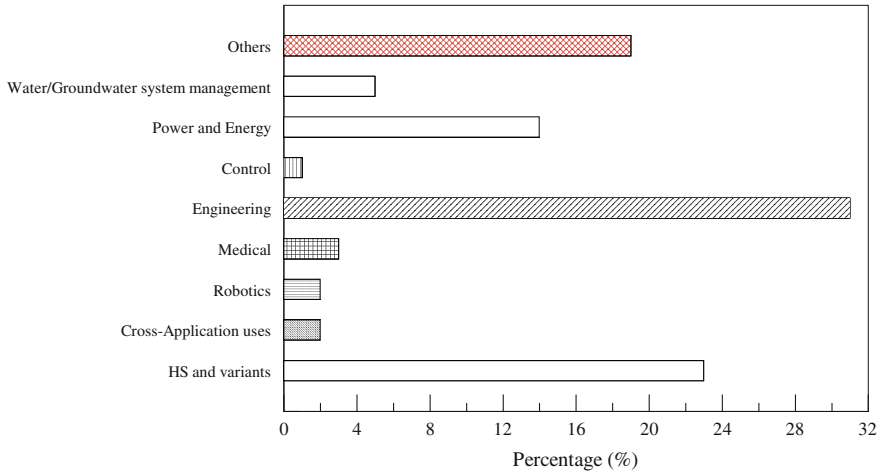


Fig. 6 Application of HS algorithm in different discipline areas

equivalent (CO_2-eq) emissions of a typical residential building. Several building envelope parameters including wall, roof, ceiling, and floor construction materials as well as glazing type are taken as the design variables. It is common to define these parameters as continuous variables because in numerical optimization methods it is difficult to deal with discrete variables. However, the optimum values obtained in this case are not necessarily available in the market, which causes mismatch between optimization suggestions for materials based on numerical results and components commonly used in design practice [34]. In order to solve the problem of the obvious mismatch, Fesanghary et al. [24] treated all variables as discrete variables and those materials which are available in the market were considered in the optimization process. To demonstrate the efficiency of the proposed approach, the performance of the model was tested on a typical single-family house. A series of Pareto optimal solutions was identified which can help designers to get a better understanding of the trade-off relation between the economical and environmental performances.

In another study conducted by Asadi [35], a multi-objective optimization model coupled with an energy simulation program with life cycle assessment (LCA) and LCC allowing the design space to be explored in search of optimal or near optimal solutions. To demonstrate the effectiveness and efficiency of the proposed method, a typical single-family house in different climate regions in the United States is selected. In this study, a series of Pareto optimal solutions was identified which can help designers to get a better understanding of the trade-off relation between the LCC and LCA at the early stages of the design. Results of this study showed the significant effect of climate regions on the global warming potential gases and life

cycle cost. It was found that changing the climate zone from zone 1 to 5 increases the global warming potential gases by 24 % and LCC by 21 %. This was mainly due to energy consumption during the use phase of building's life cycle.

As it can be found from literature review, HS has successfully been applied to a wide variety of practical optimization problem, however, there has been a lack of studies to design sustainable building and only few studies focus on this area.

2.2.2 Genetic Algorithm

Genetic Algorithm (GA) method which is an effective method for solving optimization problems was first proposed by Holland in 1975 [36]. Since then, it slowly advanced into a computer model of solving optimization problems by simulating natural evolution and was widely utilized in many domains [37]. GA is defined as heuristic search method having extensive applicability, ease of use, and the capacity to search many variables from a global perspective. Since the search of design alternatives within the design process is not linear, stochastic approaches such as GAs are normally considered suitable.

GA based solution methods have been used in several studies to predict building energy consumption and design sustainable buildings [38–65]. Among which the Vector evaluated GA (VEGA) [66], Multi-objective Genetic Algorithm (MOGA) [67], Niched Pareto Genetic Algorithm (NPGA) [68], Weight-based Genetic Algorithm (WBGA) [69], Non-dominated Sorting Genetic Algorithm (NSGA) [70], Fast Non-dominated Sorting Genetic Algorithm (NSGA-II) [71], and Multi-objective Evolutionary Algorithm (MEA) [72] are frequently used in building research.

Hauglustaine and Azar [73] used GA to optimize the building envelope. Different criteria associated with code compliance, energy use, and cost were considered in this study. In a study conducted by Wang et al. [34], a multi-objective optimization model was developed using GA that could assist designers to design green buildings. Variables considered in this study were those parameters that are usually determined at the early stages of design and that have significant impact on building performance. LCA methodology was used to assess design alternatives for both economical and environmental criteria. Life cycle environmental impacts were assessed in terms of expanded cumulative exergy consumption. They also identified a series of Pareto optimal solutions which consisted of discrete regions with different optimal solutions. GA was used by Tuhus-Dubrow and Krarti [61] to optimize nine construction and two shape parameters of a residential building to minimize LCC. In another study, Palonen et al. [54] used GA to optimize building energy consumption and cost. The design variables considered in this study were construction features and heat recovery efficiency.

In another study conducted by Hamdy et al. [74], a modified multi objective optimization approach based on GA was developed and combined with building performance simulation program. The objective function of this study was to minimize the CO_2-eq emissions and the investment cost for a two-story house and

its HVAC system. Several parameters including heating/cooling energy source, heat recovery type, and six building envelope parameters were considered as design variables. It was found that in comparison with preliminary design, the CO_2-eq emissions and investment cost were reduced by 32 and 26 %, respectively. They also found that the type of heating energy source has a noticeable effect on the optimal solutions. In a similar study, Wright et al. [75] optimized the thermal design and control of buildings employing multi-criteria GAs. The objectives of this study were to reduce the functioning cost and improve thermal comfort. The design variables included on/off status for 15 h/day, supply air flow rate, and temperature for each hour, coil width, coil height, number of rows, and water circuits in each of the two coils, water flow rate, fan diameter, and heat recovery device size which resulted in 200 design parameters.

Verbeeck and Hens [76] carried out a comprehensive study to optimize the life cycle energy, cost, and environmental impact of buildings. In this study, GA was combined with the Pareto front concept to solve a multi variable problem with multiple objectives. They determined a hierarchy of methods which can reduce building energy consumption over its life cycle. This approach consisted of improving insulation in the building, applying an efficient heating system, and finally renewable energy driven heating. The primary focus of this study was on operational energy. In another study, Kubota et al. [77] applied GA to properly extract and select measured data, and then developed fuzzy neural networks model to predict building energy load prediction.

Two different problems of optimization of building parameters regarding heating and cooling loads were addressed separately by Znouda et al. [82] to minimize the power consumption and the cost. The building parameters considered in this study were geometry of the building, composition of walls and floors, and solar protection. Later, a comprehensive optimization work using GA was carried out by Wright, Zhang and others to investigate the [78–80] the configuration of HVAC mechanisms. Various parameters such as component selections, network topologies, flow rates, and thermal capacities were considered in this optimization study. In a similar study conducted by Stanescu et al. [81], GA algorithm was used to optimize the configuration of an HVAC network by allocating different zones to different systems. DOE-2 energy simulation program was used to quantify the energy consumption of the HVAC system. Genetic algorithms are also applied to electric utility planning and building energy management problems [82].

2.2.3 Particle Swarm Optimization

Particle Swarm Optimization (PSO) which is first recommended by Kennedy and Eberhart [83] is a heuristic search method that mimics the movements of a flock of birds seeking to find food. Each bird which is called particle, in the population, named swarm, is supposed to ‘fly’ over the search space to search for proper solutions [84]. Unlike the genetic GA, PSO does not use filtering operators such as crossover and mutation in searching for solutions. As an alternative, it uses a

population of particles that "fly" through a multi-dimensional search space with given rates. The simplicity of PSO method and the fact that is a population-based technique have made it an accepted candidate to be used for multi-objective optimization [85].

In PSO, a number of simple entities—the particles—are located in the search area of some problem or function, and each assesses the objective function at its current location. Each particle then identifies its movement through the search area by uniting some aspect of the history of its own current and best locations with those of one or more members of the swarm, with some random perturbations. The next iteration occurs after all particles have been moved. Eventually the swarm as a whole, like a flock of birds collectively searching for food, is likely to move close to an optimum of the fitness function [86].

Particle swarm optimization model has several advantages. The search mechanism is robust and efficient in maintaining diversity, and is able to reach at the global optimization solution with a high probability. The algorithm is easy to realize with only a few adjustable parameters, is fast converging, and can be conducted with parallel computation. Canonical particle swarm [87] and fully-informed particle swarm [88] are defined as variants of PSO. Figure 7 shows a dramatic annual increase in the literature published in the area of PSO over the last 17 years. These studies include data clustering, sensor network clustering, building, construction, image segmentation and clustering, gene clustering, document and text clustering, hybrid clustering, and other discipline areas.

Even though PSO is a generally popular method and has been applied to tackle a wide variety of problems, but rarely has been used in studies focusing on the optimization of building design. Some studies applied PSO algorithm to design

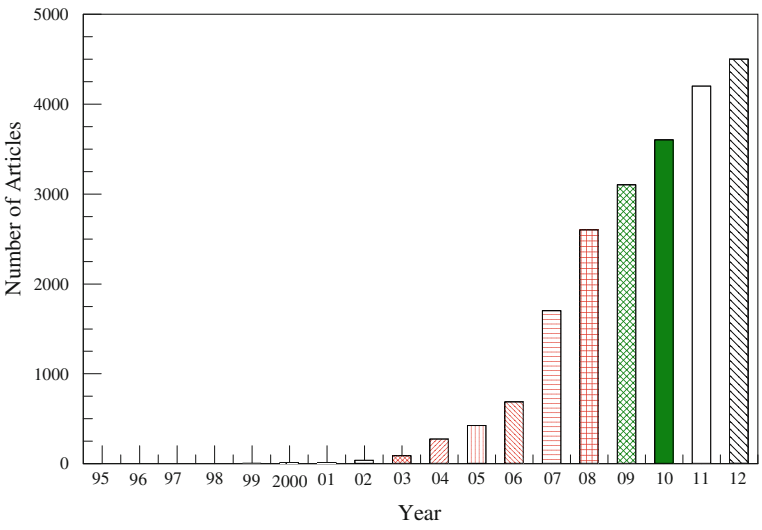


Fig. 7 Trend of PSO studies

efficient HVAC systems [89–91]. Wang et al. [92] used PSO algorithm to analyze the energy flow of the conventional separation production (SP) system and the redundant BCHP system. Four decision variables including the capacity of power generation unit (PGU), the capacity of heat storage tank, the on–off coefficient of PGU, and the ratio of electric cooling were considered in this study as the design variables. The objective function of this study was to simultaneously measure the energetic, economical, and environmental benefits attained by BCHP system in comparison to SP system.

Multi-objective Particle Swarm Optimization (MOPSO) was used by Yang et al. [93] to optimize thermal comfort and building energy consumption. This study investigates the relation between energy consumption and occupants' comfort. They suggested a multi-agent based control framework for energy and comfort management in smart building. Two objectives including energy consumption and the occupant thermal comfort were considered as the objective functions of this study. In a similar study, Kusiak et al. [94] used a data-driven approach to optimize HVAC system in an office building. A neural network (NN) algorithm was used to develop a predictive model. The NN-derived predictive model was then optimized with a strength multi-objective particle-swarm optimization (S-MOPSO) algorithm. It was found that the developed model was highly accurate and the optimization approach provides acceptable solutions for users with different preferences.

Rapone and Saro [95] used PSO to optimize curtain wall façades of office buildings and determined the configuration of selected parameters that minimizes the total carbon emissions associated with building operation. In this study, PSO algorithm was coupled to a dynamic energy simulation engine in order to conduct a completely automated search intended at determining the optimal values of the envelope features. This study investigated the effect of different types of external shading devices and different climate regions. It was found that PSO algorithm is considerably better method to quantify energy performance within a short amount of time and by simulating only a small percentage of all potential alternatives. The optimized designs prove the tendency for a reduction of cooling demands at the expense of heating energy demands and at the same time allowing for enough daylighting. In addition, results of the sensitivity analysis demonstrated the dependency of the optimized configuration on climate region and external shading devices.

Hasan et al. [96] used PSO algorithms to optimize LCC of a single detached house in Finland. They coupled the IDA ICE 3.0 building performance simulation program with the GenOpt 2.0 generic optimization program to identify optimized values of five selected design variables in the building construction and HVAC system. Insulation thickness of the external wall, roof, and floor as well as U-value of the windows and type of heat recovery was considered as design variables. It was found that coupling the combining simulation and optimization was very advantageous. Results suggested decreasing the U-value of external wall, roof, floor and the window from their initial values. Reduction of 23–49 % in the space heating energy for the optimized house is achieved in comparison with the reference case. In another study, Stephan et al. [97] used a hybrid PSO-Hooks Jeeves algorithm for

the optimization of envelope openings in order to ensure desired natural ventilation. The method presented in this study has the benefit of considerably reducing the number of independent variables.

2.2.4 Ant Colony Optimization

Ant Colony Optimization (ACO) is a Swarm Intelligence technique which stimulated from the foraging behavior of real ant colonies and was proposed by Colorni et al. [98]. The ants deposit pheromone on the ground in order to mark the route for identification of their routes from the nest to food that should be followed by other members of the colony. The basic way of working of an ACO algorithm is graphically shown in Fig. 8.

Hadjiski [98] proposed an intelligent system for HVAC control by the integration of a multi-agent system, dynamic ontology, and ant colony optimization. The combination of data-driven and knowledge-driven methods results in a significant improvement of all behavioral indexes of HVAC control systems such as speed, stability, internal communication rate, robustness and disturbances. Ghasemi and Farshchin [99] applied ACO method to investigate the ordinary moment-resisting frames for multi-objective design under seismic loading. The objective function of this study was to minimize the total weight as well as the corresponding seismic base shear. To analyze the ordinary moment-resisting frames, the modal spectral method was used.

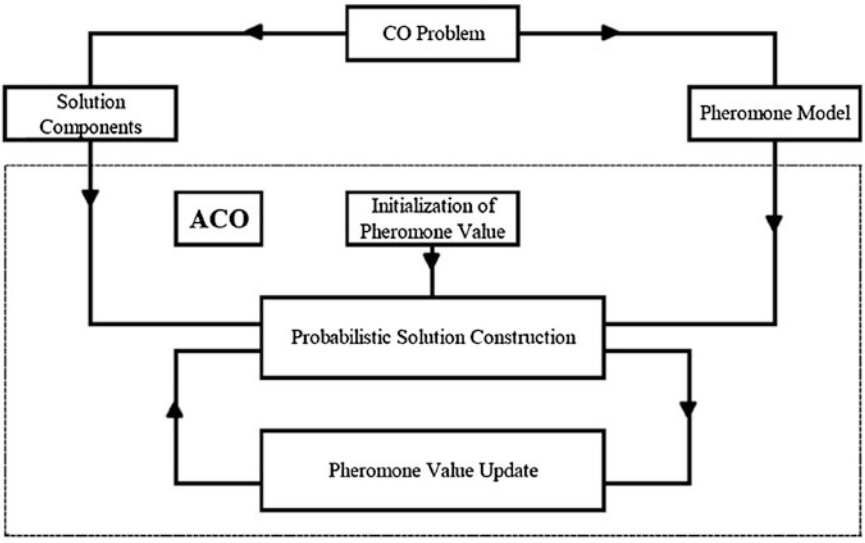


Fig. 8 The working of the ACO meta heuristic

In a study conducted by Lixing et al. [100], a novel building cooling load forecasting approach combining support vector regression (SVR) and ACO was proposed. ACO was developed to optimize three parameters of SVR, including penalty parameter, insensitive loss function, and kernel function. Normalized Mean square error (NMSE) of fitting result was used as the target of the model. ACO identified the best parameters which were corresponded to the NMSE. The results showed that the proposed approach in comparison with back-propagation neural network model was an efficient way to model building cooling load with good predictive accuracy. ACO and SVR provided a useful tool for maximizing the combustion efficiency of cooling load.

A multi criteria ACO method using Pareto filtering was used by Shea et al. [101]. The Radiance software program was used to calculate lighting performance for the media center in Paris. Results showed that the method is capable of generating Pareto optimal design and it archived up to 11 independent performance criteria. In addition, a preliminary GUI for visualizing the Pareto design archives and selecting designs was developed in this study. The results indicated that for desired values of lighting performance in different internal spaces, there is often a range of possible panel configurations and costs. In a similar study, a novel algorithm for optimization of building life cycle energy consumption was developed by improving the multi-objective ACO. In this algorithm, the estimation mechanism of Pareto optimal solution and the update rule of pheromone were derived. An effective optimization solution for building life cycle energy consumption and an innovative application of multi-objective ACO algorithm in the building energy efficiency area were presented [102].

Although in general, ACO algorithms attain very good results, there are cases where an hybridization with other heuristics or meta-heuristics is necessary. Therefore, in the past few years, authors have developed hybrid algorithms between ACO and Local Search [103], Simulated Annealing [104], Post Processing Procedures [105], and even with GA [106]. This allowed ACO algorithms to achieve even better results in complex problems solved by a single heuristic method.

2.2.5 Evolutionary Programming

Evolutionary Programming (EP) is one of a class of paradigm for simulating evolution which utilizes the concepts of Darwinian evolution to iteratively generate increasingly appropriate solutions in light of a static or dynamically changing environment. In a most general framework, EP may be considered an optimization technique wherein the algorithm iteratively optimizes behaviors and parameters.

A meta-heuristic simulation–EP coupling approach was developed by Fong et al. [107]. The simulation–EP coupling is able to solve the discrete, non-linear, and highly constrained optimization problems such as HVAC systems. The effectiveness of this simulation was shown through the establishment of a monthly optimum reset scheme for both the chilled water and supply air temperatures of the HVAC installations of a local project. Results showed that the component-based HVAC

model and the evolutionary programming technique functioned well together in achieving the optimum combination of the chilled water and supply air temperatures for effective energy management throughout a year. Later, Fong et al. [108] developed the robust evolutionary algorithm to optimize HVAC energy management system. The robust evolutionary algorithm is an effective method to manage with the complex simulation models, as well as those represented by explicit mathematical expressions of HVAC engineering optimization problems. This can improve the optimization-simulation for a variety of HVAC scenarios, even extended to other kinds of engineering problems.

Yang et al. [109] proposed an evolutionary approach to identify building thermal model parameters using SaNSDE+. The HAMBase building simulation model was used to formulate optimization problems with six and ten objective functions. Results showed that the adopted evolutionary algorithm (i.e. SaNSDE+) was very effective for parameter identification problems. It was found that SaNSDE+ is significantly better than the other evolutionary algorithms such as SGA and FEP. In another study conducted by Kampf [110], a hybrid CMA-ES/HDE algorithm was used to optimize building forms for the utilization of solar irradiation either passive or active. The backwards ray tracing program RADIANCE in conjunction with a cumulative sky model was used to predict the solar irradiation. It was found that a hybrid CMA-ES/HDE algorithm consistently converged to a good solution while taking constraints into account. Results indicated that the forms of these solutions tend to be highly non-intuitive.

Fang et al. [111] utilized an evolutionary algorithm to develop an optimal design of solar water heating system. The objective function of this study was to maximize the year-round energy saving using the solar heating instead of conventional domestic electric heating. The TRNSYS plant simulation model was developed and coupled with the optimization algorithm. The optimization results showed that the solar collectors can be installed onto the external shading devices as an integrated architectural feature, since the optimal tilt angle is 21° and relatively flat. Both the optimal values of calorifier storage capacity and pump flow rate indicated that the calculations from normal design practice may not achieve an optimal performance. Therefore, an effective methodology of optimization and simulation is required to create an optimal design.

3 Conclusions

This papers provides an overview of the latest research developments concerning to the application of meta-heuristic algorithms for design and control problems in the field of sustainable building. The review of over 100 papers from the major referenced journals in the fields of sustainable energy offers interesting conclusions that can be useful for building design researchers. The first conclusion of this review is that the number of research papers that use meta-heuristic algorithms to design sustainable buildings has increased dramatically in recent years, especially

for HVAC systems. This can be attributed to the ever-increasing computational power available to solve the problems that were previously unfeasible. In addition, we found that GA optimization technique is widely used in designing sustainable buildings in comparison with other techniques (e.g. HS, PSO, and ACO). For ACO, PSO, and even HS, we discovered there is not as much of researches in sustainable building design. The third conclusion of this review is that the number of research papers that use these methods to optimize building design, is still small compared to the number of papers regarding the optimization of building control. A possible explanation is that the whole building simulation study became popular only in the last decade. Recently, optimization tools coupled to a whole building simulation program exist and they are ready to be used. It was also found that the environmental impact, the initial cost investment, the operational cost, and comfort criteria are usually the targets of these optimization studies while construction materials, building geometry and orientation, HVAC system design, and shading options are the typical design variables.

References

1. IEA.: Key world energy statistics (2013)
2. WBCSD.: Energy efficiency in buildings, business realities and opportunities 2007. The World Business Council for Sustainable Development (2007)
3. USEIA.: International Energy Outlook 2010. Department of Energy, Washington, DC (2010)
4. WBCSD.: Vision 2050: the new agenda for business. World Business Council for Sustainable Development (2010)
5. OECD.: Design of sustainable building policies. Available at http://www.uea.ac.uk/env/ (2002)
6. BIONIS.: Available at http://www.extra.rdg.ac.uk/eng/BIONIS/ (2004)
7. Evins, R.: A review of computational optimisation methods applied to sustainable building design. Renew. Sustain. Energy Rev. **22**, 230–245 (2013)
8. Evins, R.: A review of computational optimisation methods applied to sustainable building design. Renew. Sustain. Energy Rev. **22**, 230–245 (2012)
9. Nguyen, A.-T., Reiter, S., Rigo, P.: A review on simulation-based optimization methods applied to building performance analysis. Appl. Energy **113**, 1043–1058 (2014)
10. Glover, F., Laguna, M.: Tabu Search. Kluwer Academic Publishers, Dordrecht (1997)
11. Blum, C., Roli, A.: Metaheuristics in combinatorial optimisation: overview and conceptual comparison. ACM Comput. Surv. **35**, 268–308 (2003)
12. Yang, X.S.: Nature-Inspired Metaheuristic Algorithms. Luniver Press, Bristol (2008)
13. Yang, X.S.: Harmony search as a metaheuristic algorithm. In: Geem, Z.W. (ed.) Music-Inspired Harmony Search Algorithm. Springer, Berlin (2008)
14. Geraili, A., Sharma, P., Romagnoli, J.A.: A modeling framework for design of nonlinear renewable energy systems through integrated simulation modeling and metaheuristic optimization: applications to biorefineries. Comput. Chem. Eng. **61**, 102–117 (2014)
15. Geraili, A., Sharma, P., Romagnoli, J.A.: Technology analysis of integrated biorefineries through process simulation and hybrid optimization. Energy, 1–15 (2014)
16. Yang, X.S.: Review of metaheuristics and generalized evolutionary walk algorithm. Int. J. Bio-Inspired Comput. **3**(2), 77–84 (2011)
17. Kennedy, J., Eberhart, R.: Particle swarm optimisation. In: Proceedings of the IEEE International Conference on Neural Networks. Piscataway, NJ (1995)

18. Kirkpatrick, S., Gellat, C.D., Vecchi, M.P.: Optimisation by simulated annealing. Science **220**, 671–680 (1983)
19. Manjarres, D., Linda-Torres, I., Gil-Lopez, S., Del Ser, J., Bilbao, M.N., Salcedo-Sanz, S., Geem, Z.W.: A survey on applications of the harmony search algorithm. Eng. Appl. Artif. Intell. **26**, 1818–1831 (2013)
20. Geem, Z.W.: Optimal cost design of water distribution networks using harmony search. Eng. Optim. **38**(3), 259–280 (2006)
21. Vasebi, A., Fesanghary, M., Bathaee, S.M.T.: Combined heat and power economic dispatch by harmony search algorithm. Int. J. Electr. Power **29**, 713–719 (2007)
22. Doodman, A., Fesanghary, M., Hosseini, R.: A robust stochastic approach for design optimization of air cooled heat exchanger. Appl. Energy **86**, 1240–1245 (2009)
23. Cheng, Y.M., Li, L., Lansivaara, T., Chi, S.C., Sun, Y.J.: An improved harmony search minimization algorithm using different slip surface generation methods for slope stability analysis. Eng. Optim. **40**, 95–115 (2008)
24. Fesanghary, M., Asadi, S., Geem, Z.W.: Design of low-emission and energy-efficient residential buildings using a multi-objective optimization algorithm. Build. Environ. **49**, 245–250 (2012)
25. Li, H.Q., Li, L.: A novel hybrid particle swarm optimization algorithm combined with harmony search for high dimensional optimization problems. In: International Conference on Intelligent Pervasive Computing, Korea (2007)
26. Lee, K.S., Geem, Z.W.: A new structural optimization method based on the harmony search algorithm. Comput. Struct. **82**(9–10), 781–798 (2004)
27. Geem, Z.W.: School bus routing using harmony search. In: Genetic and Evolutionary Computation Conference. Washington, DC (2005)
28. Geem, Z.W., Tseng, C.-L., Park, Y.: Harmony search for generalized orienteering problem: best touring in China. In: Lecture Notes in Computational Science, pp. 741–750 (2005)
29. Geem, Z.W., Kim, J.H., Loganathan, G.V.: A new heuristic optimization algorithm: harmony search. Simulation **76**(2), 60–68 (2001)
30. Kim, J.H., Geem, Z.W., Kim, E.S.: Parameter estimation of the nonlinear Muskingum model using harmony search. J. Am. Water Resour. Assoc. **37**(5), 1131–1138 (2001)
31. Kaveh, A., Talatahari, S.: Particle swarm optimizer, ant colony strategy and harmony search scheme hybridized for optimization of truss structures. Comput. Struct. **87**, 267–283 (2009)
32. Kaveh, A., Abadi, A.S.M.: Cost optimization of a composite floor system using an improved harmony search algorithm. J. Constr. Steel Res. **66**(5), 664–669 (2010)
33. Ayvaz, T.: Simultaneous determination of aquifer parameters and zone structures with fuzzy c-means clustering and meta-heuristic harmony search algorithm. Adv. Water Resour. **30** (11), 2326–2338 (2007)
34. Wang, W.M., Zmeureanu, R., Rivard, H.: Applying multi-objective genetic algorithms in green building design optimization. Build. Environ. **40**, 1512–1525 (2005)
35. Asadi, S.: A multiobjective harmony-search algorithm for building life-cycle energy optimization. In: Construction Research Congress, Atlanta (2014)
36. Holland, J.H.: Adaptation in Natural and Artificial Systems. MIT Press, Cambridge (1975)
37. Bian, J., Bai, Q., Shi, Q.: A review of genetic algorithms applied to optimizing maintenance strategy of bridges. Adv. Mater. Res. **374–377**, 2437–2441 (2012)
38. Caldas, L.G., Norford, L.K.: A design optimization tool based on a genetic algorithm. Autom. Constr. **11**(2), 173–184 (2002)
39. Caldas, L.: Generation of energy-efficient architecture solutions applying GENE_ARCH: an evolution-based generative design system. Adv. Eng. Inform. **22**(1), 59–70 (2008)
40. Coley, D., Schukat, S.: Low-energy design: combining computer-based optimisation and human judgement. Build. Environ. **37**, 1241–1247 (2002)
41. Congradac, V., Kulic, F.: Recognition of the importance of using artificial neural networks and genetic algorithms to optimize chiller operation. Energy Build. **47**, 651–658 (2012)
42. Chow, T.T., Zhang, G.Q., Lin, Z., Song, C.L.: Global optimization of absorption chiller system by genetic algorithm and neural network. Energy Build. **34**(1), 103–109 (2002)

43. Evins, R., Pointer, P., Vaidyanathan, R.: Multi-objective optimisation of the configuration and control of a double-skinfacade. In: Proceedings of the Building Simulation (2011)
44. Evins, R., Pointer, P., Burgess, S.C.: Multi-objective optimisation of a modular building for different climate types. In: Proceedings of the Building Simulation and Optimisation (2012)
45. Evins, R., Pointer, P., Vaidyanathan, R., Burgess, S.: A case study exploring regulated energy use in domestic buildings use in design-of-experiments and multi-objective optimisation. Build. Environ. **54**, 126–136 (2012)
46. Gagne, J., Andersen, M.: A generative facade design method based on daylighting performance goals. J. Build. Perform. Simul. **5**(3), 141–154 (2011)
47. Hamdy, M.H.A., Siren, K.: Applying a multi-objective optimization approach for design of low-emission cost-effective dwellings. Build. Environ. **46**(1), 109–123 (2011)
48. Hamdy, M., Hasan, A., Siren, K.: Amulti-stage optimization method forcost- optimal and nearly-zero-energy building solutions in line with the EPBD- Recast 2010. Energy Build. **56**, 189–203 (2013)
49. Jin, Q., Overend, M.: Facade renovation for a public building based on a whole-life value approach. In: Proceedings of The Building Simulation and Optimization Conference (2012)
50. Huang, H., Kato, S., Hu, R.: Optimum design for indoor humidity by coupling genetic algorithm with transient simulation based on contribution ratio of indoor humidity and climate analysis. Energy Build. **47**, 208–216 (2012)
51. Kayo, G., Ooka R.: Application multi-objective genetic algorithm for optimal design method of distributed energy system. In: Proceedings of the Building Simulation (2009)
52. Lu, L., Cai, W., Xie, L., Li, S., Soh, Y.C.: HVAC system optimization—in-building section. Energy Build. **37**(1), 11–22 (2005)
53. Li, H., Nalim, R., Haldi, P.-A.: Thermal-economic optimization of a distributed multi-generation energy system a case study of Beijing. Appl. Therm. Eng. **26** (2006)
54. Palonen, M., Hasan, A., Siren, K.: A genetic algorithm for optimization of building envelope and HVAC system parameters. In: Proceedings of the Building Simulation (2009)
55. Ooka, R., Komamura, K.: Optimal design method for building energy systems using genetic algorithms. Build. Environ. **44**(7), 1538–1544 (2009)
56. Pernodet, F., Lahmidi, H., Michel, P.: Use of genetic algorithms for multicriteria optimization of building refurbishment. In: Proceedings of the Building Simulation (2009)
57. Pountney, C.: Better carbon saving: using a genetic algorithm to optimise building carbon reduction. In: Proceedings of the Building Simulation and Optimization Conference (2012)
58. Salminen, M., Palonen, M., Siren, K.: Combined energy simulation and multi-criteria optimisation of a LEED-certified building. In: Proceedings of the Building Simulation and Optimization Conference (2012)
59. Sahu, M., Bhattacharjee, B., Kaushik, S.: Thermal design of air-conditioned building for tropical climate using admittance method and genetic algorithm. Energy Build. **53**, 1–6 (2012)
60. Romero, D., Rincon, J., Almao, N.: Optimization of the thermal behavior of tropical buildings. In: Proceedings of the Building Simulation (2001)
61. Tuhus-Dubrow, D., Krarti, M.: Genetic-algorithm based approach to optimize building envelope design for residential buildings. Build. Environ. **45**(7), 1574–1581 (2010)
62. Zheng, D.X.M., Ng, S.T., Kumaraswamy, M.M.: Applying a genetic algorithm-based multiobjective approach for time-cost optimization. J. Constr. Eng. Manage. **130**(2), 168–176 (2004)
63. Krarti, M.: An overview of artificial intelligence-based methods for building energy systems. J. Sol. Energy Eng. **125**, 331 (2003)
64. Huang, W., Lam, H.N.: Using genetic algorithms to optimize controller parameters for HVAC systems. Energy Build. **26**, 277–282 (1997)
65. Guillemin, A., Morel, N.: An innovative lighting controller integrated in self-adaptive building control system. Energy Build. **33**, 477–487 (2001)
66. Schaffer, J.: Multi objective optimization with vector evaluated genetic algorithms. In: International Conference on Genetic Algorithm and Their Applications (1987)

67. Fonseca, C.M., Fleming, P.J.: Multi-objective genetic algorithms. In: IEE Colloquium on Genetic Algorithms for Control Systems Engineering, London (1993)
68. Horn, J., Nafpliotis, N., Goldberg, D.E.: A Niched Pareto Genetic Algorithm for Multiobjective Optimization. In: Proceedings of the First IEEE Conference on Evolutionary Computation (1994)
69. Hajela, P., Lin, C.-Y.: Genetic search strategies in multicriterion optimal design. Struct. optim. **4**(2), 99–107 (1992)
70. Srinivas, N., Deb, K.: Muiltiobjective optimization using nondominated sorting in genetic algorithms. Evol. Comput. **2**(3), 221–248 (2007)
71. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. In: IEEE Transactions on Evolutionary Computation (2002)
72. Sarker, R., Liang, K., Newton, C.: A new multiobjective evolutionary algorithm. Eur. J. Oper. Res. **140**(1), 12–23 (2002)
73. Hauglustaine, J.-M., Azar, S.: Interactive tool aiding to optimise the building envelope during the sketch design. In: Lamberts, R., Negarao, C.O.R., Hensen, J. (eds.) Proceedings of the 7th International IBPSA Conference (2001)
74. Hamdy, M., Hasan, A., Siren, K.: Applying a multi-objective optimization approach for Design of low-emission cost-effective dwellings. Build. Environ. **46**, 109–123 (2011)
75. Wright, J.A., Loosemore, H.A., Farmani, R.: Optimization of building thermal design and control by multi-criterion genetic algorithm. Energy Build. **34**(9), 959–972 (2002)
76. Verbeeck, G., Hens, H.: Life cycle optimisation of extremely low energy dwellings. J. Build. Phys. **31**(2), 143–177 (2007)
77. Kubota, N., Hashimoto, S., Kojima, F., Taniguchi K.: GP-preprocessed fuzzy inference for the energy load prediction. In: Proceedings of the 2000 Congress on Evolutionary Computation, pp. 1–6 (2000)
78. Zhang, Y., Hanby, V., Wright, J.A.: Energy aspects of HVAC system configurations—problem definition and test cases. HVAC&R Res. **12**(3c), 871–888 (2006)
79. Wright, J.A., Zhang, Y.: Evolutionary synthesis of HVAC system configurations: experimental results. HVAC&R Res. **14**(1), 57–72 (2008)
80. Wright, J.A., Zhang, Y., Angelov, P., Hanby, V.I., Buswell, R.A.: Evolutionary synthesis of HVAC system configurations: algorithm development. HVAC&R Res. **14**(1), 33–55 (2008)
81. Stanescu, M., Kajl, S., Lamarche, L.: Evolutionary algorithm with three different permutation options used for preliminary HVAC system design. In: Proceedings of the Buildings Simulation and Optimization Conference (2012)
82. Wright, J.A., Loosemore, H., Fermani, R.: Optimization of building thermal design and control by multi-criterion genetic algorithm. Energy Build. **34**, 959–972 (2002)
83. Kennedy, J., Eberhart, R.C.: Particle swarm optimization. In: IEEE International Conference on Neural Networks Man, and Cybernetics, Piscataway, pp. 1942–1948 (1995)
84. Shi, Y., Eberhart, R.: A modified particle swarm optimizer. In: Proceedings of IEEE International Joint Conference on Evolutionary Computation, pp. 69–73 (1998)
85. del Valle, Y., Venayagamoorthy, G.K., Mohaghenghi, S., Hernandez, J.C., Harley, R.G.: Particle swarm optimization: basic concepts, variants and applications in power systems. IEEE Trans. Evol. Comput. **12**, 171–195 (2008)
86. Poli, R., Kennedy, J., Blackwell, T.: Particle swarm optimization. Overv. Swarm Intell. **1**, 33–57 (2007)
87. van den Bergh, F., Engelbrecht, A.: A new locally convergent particle swarm optimiser. In: IEEE International Conference on Systems, Man and Cybernetics, vol. 3 (2002)
88. Zhang, J., Huang, D.S., Liu, K.H.: Multi-sub-swarm particle swarm optimization algorithm for multimodal function optimization. In: IEEE Congress on Evolutionary Computation (CEC 2007), pp. 3215–3220 (2007)
89. Bravo, R.H., Flocker, F.W.: Designing HVAC systems using particle swarm optimization. HVAC&R Res. **18**(5), 845–857 (2011)
90. Patel, V.K., Rao, R.V.: Design optimization of shelland—tube heat exchanger using particle swarm optimization technique. Appl. Therm. Eng. **30**, 1417–1425 (2010)

91. Khooban, M.H., Soltanpour, M.R., Abadi, D.N.M., Esfahani, Z.: Optimal intelligent control for HVAC systems. J. Power Technol. **92**(3), 192–200 (2012)
92. Wang, J., Zhai, Z., Jing, Y., Zhang, C.: Particle swarm optimization for redundant building cooling heating and power system. Appl. Energy **87**, 3668–3679 (2010)
93. Yang, R., Wang, L., Wang, Z.: Multi-objective particle swarm optimization for decision-making in building automation. In: IEEE Power and Energy Society General Meeting (2011)
94. Kusiak, A., Xu, G., Tang, F.: Optimization of an HVAC system with a strength multi-objective particle-swarm algorithm. Energy **36**, 5935–5943 (2011)
95. Rapone, G., Saro, O.: Optimisation of curtain wall facades for office buildings by means of PSO algorithm. Energy Build. **45**, 189–196 (2012)
96. Hasan, A., Vuolle, M., Siren, K.: Minimisation of life cycle cost of a detached house using combined simulation and optimisation. Build. Environ. **43**, 2022–2034 (2008)
97. Stephan, L., Bastide, A., Wurtz, E., Souyri, B.: Ensuring desired natural ventilation rate by means of optimized openings. In: Building Simulation. Glasgow (2009)
98. Hadjiski, V.B.: Dynamic Ontology-based approach for HVAC Control via Ant Colony Optimization in DECOM 2007. Izmir (2007)
99. Ghasemi, M.R., Farshchin, M.: Ant colony optimisation-based multiobjective frame design under seismic conditions. In: Proceedings of the ICE—Structures and Buildings (2011)
100. Ding, L., Lv, J., Li, X., Li, L.: Support vector regression and ant colony optimization for HVAC cooling load prediction. In: International Symposium on Computer, Communication, Control and Automation, pp. 537–541 (2010)
101. Shea, K., Sedgwick, A., Antonuntto, G.: Multicriteria optimization of paneled building envelopes using ant colony optimization. In: Intelligent Computing in Engineering and Architecture, pp. 627–636. Springer, Berlin (2006)
102. Yuan, Y., Yuan, J., Du, H., Li, L.: An improved multi-objective ant colony algorithm for building life cycle energy consumption optimisation. Int. J. Comput. Appl. Technol. **43**(1), 60–66 (2012)
103. Pour, H.D., Nosraty, M.: Solving the facility layout and location problem by ant-colony optimization-meta heuristic. Int. J. Prod. Res. **44**, 5187–5196 (2006)
104. Bouhafs, L., Hajjam, A., Koukam, A.: A combination of simulated annealing and ant colony system for the capacitated location-routing problem. In: KES, pp. 409–416 (2006)
105. Crawford, B., Castro, C.: Integrating lookahead and post processing procedures with aco for solving set partitioning and covering problems. In: *ICAISC*, pp. 1082–1090 (2006)
106. Altiparmak, F., Karaoglan, I.: A genetic ant colony optimization approach for concave cost transportation problems. In: Evolutionary Computation, pp. 1685–1692 (2007)
107. Fong, K.F., Hanby, V.I., Chow, T.T.: HVAC system optimization for energy management by evolutionary programming. Energy Build. **38**, 220–231 (2006)
108. Fong, K.F., Hanby, V.I., Chow, T.T.: System optimization for HVAC energy management using the robust evolutionary algorithm. Appl. Therm. Eng. **29**, 2327–2334 (2009)
109. Yang, Z., Li, X., Bowers, C.P., Schnier, T., Tang, K., Yao, X.: An efficient evolutionary approach to parameter identification in a building thermal model. IEEE Trans. Syst. Man Cybern. Part C Appl. Rev. **42**(6), 957–968 (2012)
110. Kampf, J.H., Robinson, D.: Optimisation of building form for solar energy utilisation using constrained evolutionary algorithms. Energy Build. **42**, 807–814 (2010)
111. Fong, K.F., Chow, T.T., Hanby, V.I.: Development of optimal design of solar water heating system by using evolutionary algorithm. In: ASME 2005 International Solar Energy Conference, pp. 333–341. ASME, Orlando (2005)

Firefly Algorithm for Flow Shop Optimization

M.K. Marichelvam, T. Prabaharan and M. Geetha

Abstract In this chapter, a recently developed bio-inspired meta-heuristic algorithm namely firefly algorithm (FA) is addressed to solve the flow shop scheduling problems with sequence dependent setup times which have been proved to be strongly NP-hard type of combinatorial optimization problems. Four different performance measures namely minimization of makespan, mean flow time, mean tardiness and number of tardy jobs are considered. Extensive computational experiments were carried out to compare the performance of the proposed FA on different random problem instances. The results indicate that the proposed FA is more effective than many other algorithms reported earlier in the literature.

Keywords Flow shop · Scheduling · NP-hard · Firefly algorithm (FA) · Makespan · Flow time · Tardiness · Tardy jobs

1 Introduction

Scheduling is defined as a process of allocating resources over time to perform a collection of tasks [3]. It is a decision-making process and plays a vital role for the development industries. Scheduling problems are non-deterministic polynomial time hard (NP-hard) type combinatorial optimization problems. Hence it is difficult to solve the problems. Researchers addressed different type of scheduling problems [44]. Among them flow shop scheduling problems have attracted the researchers for the past several decades. Many industries such as metal, plastic, chemical and food

M.K. Marichelvam (✉) · T. Prabaharan
Department of Mechanical Engineering, Mepco Schlenk Engineering College, Sivakasi 626005, Tamilnadu, India
e-mail: mkmarichelvamme@gmail.com

M. Geetha
Department of Mathematics, Kamaraj College of Engineering and Technology, Virudhunagar 626001, Tamilnadu, India

X.-S. Yang (ed.), *Recent Advances in Swarm Intelligence and Evolutionary Computation*, Studies in Computational Intelligence 585, DOI 10.1007/978-3-319-13826-8_12

industries resemble the flow shop environment. Most of the research papers addressed the flow shop scheduling problems without considering the setup times. However, setup time plays significant role in many industries such as ceramic tile and paper cutting industries [53]. Hence, in this chapter we consider the flow shop scheduling problems with sequence dependent setup times. Moreover, researchers have proposed many bio-inspired metaheuristic algorithms recently. Firefly algorithm is one of the algorithms. We present the firefly algorithm to solve the flow shop scheduling problems to minimize the makespan, mean flow time, mean tardiness and number of tardy jobs. The remaining of the chapter is organised as follows. A brief review of literature is presented in Sect. 2. The problem definition is presented in Sect. 3. The proposed firefly algorithm is explained in detail in Sect. 4. Section 5 illustrates the computational results. Finally, the conclusions and future research opportunities are discussed in Sect. 6.

2 Literature Review

This section will briefly present the backgrounds necessary for the current study. It includes the flow shop scheduling problems with different objective functions and firefly algorithm.

2.1 Flow Shop Scheduling

The flow shop model was first proposed by Johnson [27] and many researchers addressed different types of scheduling problems for the past few decades. However, it has been reported that most of the researchers considered makespan minimization as the objective function [23] and very few of them only considered the setup times. Corwin and Esogbue [11] first proposed a two machine flow shop scheduling problems with sequence dependent setup times. They developed a dynamic programming to minimize the makespan. They compared the dynamic programming approach with the branch-and-bound algorithm. Gupta and Darrow [20] developed four efficient approximate algorithms to solve the two-machine flow shop scheduling problems sequence dependent setup times to minimize the makespan. Srikar and Ghosh [58] developed a mixed integer linear programming model for the flow shop scheduling problems with sequence dependent setup times. Osman and Potts [41] proposed a simulated annealing (SA) algorithm for solving the permutation flow-shop scheduling problems to minimize the maximum completion time. Rios-Mercado and Bard [51] addressed two new heuristics for solving the flow shop scheduling problems with sequence-dependent setup time to minimize the makespan. T'kindt et al. [60] applied the ant colony optimization (ACO) algorithm to solve a 2-machine bicriteria flow shop scheduling problem with the objective of minimizing both the total completion time and the makespan criteria.

The proposed algorithm heuristic also used the features of SA and local search algorithms. Computational experiments showed the effectiveness of the proposed algorithm.

Rajendran and Ziegler [49] proposed two ACO algorithms for permutation flow shop scheduling problems to minimize the makespan and total flow time. They tested the performance of the proposed algorithms with the benchmark problems addressed in the literature. França et al. [16] proposed two evolutionary algorithms namely genetic algorithm (GA) and a memetic algorithm (MA) with local search for solving flowshop scheduling problems with sequence dependent family setup times to obtain optimal schedule. Ravindran et al. [50] proposed three different heuristics to minimize the makespan and total flow time in a flow shop scheduling environment. The effectiveness of the heuristics was tested with the benchmark problems addressed in the literature. Ruiz et al. [53] addressed the different variants of GA to solve the permutation flowshop scheduling problems with sequence dependent setup times to minimize the makespan. They calibrated the parameters and operators of the GA by means of Design of Experiments. They tested the proposed algorithms with the benchmark problems addressed in the literature and proved that the proposed algorithms were superior. Liao et al. [32] addressed a discrete particle swarm optimization (DPSO) algorithm for solving the flow shop scheduling problems. A local search was also incorporated into the proposed algorithm. Computational results showed that the proposed PSO algorithm was very competitive. Marichelvam [34] addressed an improved hybrid cuckoo search algorithm to minimise the makespan in FSSPs. Chowdhury et al. [9] proposed a novel GA to solve the blocking flow shop problems to minimize the makespan.

Ignall and Schrage [25] applied the branch-and-bound algorithm to solve the flow shop scheduling problems to minimize the flow time. Rajendran [46] developed a heuristic algorithm for solving flow shop scheduling problems to minimize the total flowtime. Vempati et al. [67] developed an effective heuristic to solve the flow shop scheduling problems to minimise the total flow time. Neppalli et al. [40] developed two GA based approaches for solving the two-stage bicriteria flow shop scheduling problems. The objective was minimization of total flow time and makespan. Computational experiments showed that the proposed algorithms were effective. Rajendran and Ziegler [47] developed a new heuristic to minimize the sum of weighted flow time in a flow shop environment with sequence-dependent setup times of jobs. Extensive computational experiments showed that the proposed heuristic was faster and more effective than other heuristics. Gupta et al. [21] proposed a tabu search algorithm to minimise the total flow time in a flow shop environment. Gupta et al. [22] proposed several polynomial heuristic solution algorithms to solve the two-machine flow shop scheduling problems to minimize the total flow time and makespan. Tang and Liu [59] developed modified version of GA to solve the flow shop scheduling problems to minimise the mean flow time. Varadharajan and Rajendran [66] developed a multi-objective SA (MOSA) algorithm for solving the flow shop scheduling problems to minimise the makespan and total flow time. Nagano and Moccellin [38] proposed a constructive heuristics to minimise the mean flow time in a flow shop environment. Tasgetiren et al. [61]

applied the PSO algorithm to solve the flow shop scheduling problems for minimizing the makespan and total flow time. Yagmahan and Yenisey [71] addressed an ACO algorithm for solving the multi-objective flow shop scheduling problems. They considered minimization of makespan, total flow time and total machine idle time as the objective function. They compared the performance of the proposed algorithm with other multi-objective heuristics. Computational results showed that proposed algorithm was more effective and better than other methods compared. Dong et al. [14] suggested an iterated local search algorithm to minimise the total flow time in the flow shops. A distribution algorithm was proposed by Jarboui et al. [26] for minimizing the total flow time.

Zhang et al. [72] presented a hybrid GA for solving the flow shop scheduling problems with total flow time objective. Chakraborty and Turvey [7] addressed a differential evolution (DE) algorithm and Czapiński [12] addressed a parallel SA algorithm with genetic enhancement for solving the flow shop problems with flow time objective. A mathematical programming model was developed by Salmasi et al. [54] for minimizing total flow time of the flow shop with sequence dependent group scheduling. They proposed a tabu search algorithm and a hybrid ACO algorithm to solve the problem. Tseng and Lin [64] proposed a genetic local search algorithm for minimizing the total flow time. A discrete harmony search algorithm was developed for solving the flow shop scheduling problems to minimise the total flow time by Gao et al. [18]. Tasgetiren et al. [62] addressed a discrete artificial bee colony (ABC) algorithm to minimise the total flow time in permutation flow shops. Two constructive heuristics were developed by Gao et al. [19] to solve the no-wait flow shop scheduling problems to minimize the total flow time.

Simons [57] proposed several decomposition methods to solve the reentrant flow shop scheduling problems with sequence dependent setup times to minimize maximum lateness. Kim [31] also developed a branch-and-bound algorithm to minimize the total tardiness in a permutation flow shop environment. Murata et al. [37] developed a multi-objective GA (MOGA) to solve the flow shop scheduling problems to minimize the makespan, total tardiness and total flow time. Parthasarathy and Rajendran [43] also proposed the SA algorithm to solve the flow shop scheduling problems with sequence-dependent setup times to minimize mean weighted tardiness. They considered a drill-bit manufacturing industry. Computational results revealed that the proposed heuristic was better than many other algorithms. Armentano and Ronconi [1] proposed a tabu search based heuristic for solving the flow shop scheduling problems to minimize total tardiness. Rajendran and Ziegler [48] proposed heuristics to solve the flow shop scheduling problems with sequence-dependent setup times to minimize the sum of weighted flowtime and weighted tardiness of jobs.

Arroyo and Armentano [2] developed a genetic local search for solving the multi-objective flow shop scheduling problems. The algorithm was applied to the flowshop scheduling problem for the following two pairs of objectives: (i) makespan and maximum tardiness; (ii) makespan and total tardiness. The performance of the proposed algorithm was compared with two multi-objective genetic local search algorithms proposed in the literature. Computational results showed that the

proposed algorithm was better than other algorithms. Rahimi-Vahed and Mirghorbani [45] developed a multi-objective PSO (MOPSO) algorithm for solving the flow shop scheduling problems to minimize the weighted mean completion time and weighted mean tardiness. The computational results showed that the proposed algorithm was better than the GA. Ruiz and Stützle [52] developed two new iterated greedy heuristics to solve the flowshop scheduling problems with sequence dependent setup times. Minimization of makespan and weighted tardiness were the objectives considered by them. Tavakkoli-Moghaddam et al. [63] addressed a hybrid multi-objective algorithm based on the features of a biological immune system (IS) and bacterial optimization (BO) to find Pareto optimal solutions for solving the multi-objective no-wait flow shop scheduling problems to minimize the weighted mean completion time and weighted mean tardiness. They compared the performance of the proposed algorithm against five different multi-objective evolutionary algorithms addressed in the literature and proved that the proposed algorithm was efficient. Naderi et al. [39] presented an electromagnetism-like mechanism and SA algorithms for flow shop scheduling problems for minimizing the total weighted tardiness and makespan. Pan et al. [42] presented a discrete ABC algorithm to solve the lot-streaming flow shop scheduling problems with the criterion of total weighted earliness and tardiness. They used the dispatching rules to construct the initial population. A simple and effective local search approach was also used by them. Khalili and Tavakkoli-Moghaddam [30] presented a new multi-objective electromagnetism algorithm (MOEM) to solve the bi-objective flowshop scheduling problems. The objective was to minimize the makespan and total weighted tardiness. They considered the transportation times between machines. They also applied the SA algorithm to solve the given problem. They conducted the computational experiments and proved that the proposed MOEM provided better results. Boxma and Forst [6] proposed heuristics to minimize the expected weighted number of tardy jobs in a stochastic flow shops.

2.2 Firefly Algorithm

Firefly algorithm (FA) is one of the recently developed meta-heuristic algorithms by Yang [70]. Sayadia et al. [55] presented a new discrete firefly algorithm (DFA) meta-heuristic to minimize the makespan for the permutation flow shop scheduling problems. Computational results indicated that the proposed DFA was better than the ACO algorithm for the well known benchmark problems reported in the literature. Banati and Bajaj [4] presented a new feature selection approach by combining the rough set theory with the FA. The FA was applied by Gandomi et al. [17] to solve the mixed continuous/discrete structural optimisation problems. Kazemzadeh and Kazemzadeh [28] proposed an improved FA to solve the structural optimisation problems. Liu and Ye [33] solved the permutation flow shop scheduling problem by

a FA to minimize the makespan. Computational experiments proved the efficiency of the proposed FA.

The FA has been applied to solve the clustering problems by Senthilnath et al. [56]. Chandrasekaran and Simon [8] proposed a binary real coded FA to solve the unit commitment problem. Coelho and Mariani [10] proposed the FA to solve the multivariable PID controller tuning problems. Dekhici et al. [13] applied the FA for economic power dispatching problems with pollutants emission. The FA was applied for vector quantization in image compression problems by Horng [24]. The job shop scheduling problems have been solved using the FA by Khadwilard et al. [29]. They also investigated the different parameters for the proposed algorithm and compared the performance with different parameters. The FA was applied for solving the economic load dispatching problems by Yang et al. [69].

An efficient FA was presented by Miguel et al. [36] to simultaneously optimize the size, shape and topology of the truss structures. They proved the effectiveness of the proposed FA by solving the benchmark problems reported in the literature. Fister et al. [15] presented a comprehensive review of FAs. Yang [68] proposed a multi-objective FA for continuous optimisation problems. Vahedi Nouri et al. [65] proposed a hybrid algorithm based on firefly and SA algorithms for solving the flow shop scheduling problems. The objective was to minimize the sum of tardiness costs and maintenance costs. A mixed integer linear programming model was proposed to formulate the problem. Marichelvam et al. [35] addressed a DFA for solving the hybrid flow shop scheduling problems to minimise the makespan and mean flow time.

3 Problem Definition

Flow shop scheduling environment consists of bank of m machines in series and n jobs are to be scheduled. Each job should be processed on the machines in a particular sequence. A job is first processed on machine 1, then on machine 2 and finally completed on machine m. The processing time of the jobs are known in advance, fixed and nonnegative. It is expected that the jobs are available at time zero. Each machine can processes only one job at a time and each job can be processed on only one machine at a time. The processing of each job cannot be interrupted, that is, preemption is not allowed. It is also assumed that the machines are available for the entire scheduling period (no machine breakdown). Minimization of makespan, mean flow time, mean tardiness and number of tardy jobs are the objective functions considered. The flow shop scheduling environment is given in Fig. 1 [34].

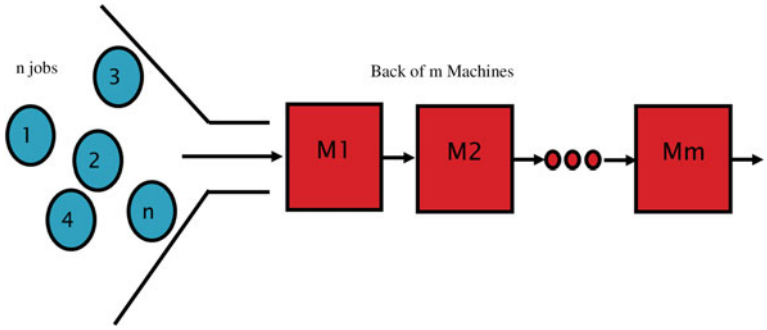


Fig. 1 Layout of a flow shop environment

3.1 Makespan (C_{max})

Makespan is the completion time of the last job to leave the system production system. Minimizing makespan would lead to the maximization of the resource utilisation and the throughput of a production system.

3.2 Mean Flow Time ($\overline{f}$)

Mean flow time is defined as the average time spent by the jobs in the production system. It is one of the most important performance measures. Mean flow time can help to effective utilization of resources, rapid turn-around of jobs, and minimization of work-in-process inventory costs.

3.3 Mean Tardiness ($\overline{T}$)

The tardiness is defined as the lateness of a job if it fails to meet its due date, or zero otherwise. Tardiness is associated with the service quality and customer satisfaction.

3.4 Number of Tardy Jobs (N_T)

This performance measure represents how many jobs are delayed in satisfying the due date. The detailed mathematical model for makespan, mean flow time, mean tardiness and number of tardy jobs can be found in [3].

3.5 Objective Function

The objective of this chapter is to minimize the weighted sum of makespan, mean flow time, mean tardiness and number of tardy jobs.

$$Z = w_1 C_{\max} + w_2 \overline{f} + w_3 \overline{T} + w_4 N_T \quad (1)$$

where w_1, w_2, w_3 and w_4 are the weight values of the objective functions and $w_1 \geq 0,\ w_2 \geq 0,\ w_3 \geq 0$ and $w_4 \geq 0$.

$$w_1 + w_2 + w_3 + w_4 = 1 \quad (2)$$

4 Firefly Algorithm

Firefly algorithm (FA) is a nature- inspired meta-heuristic algorithm. The FA is inspired by the social behavior of fireflies. Fireflies may also be called as lighting bugs. There are about two thousand firefly species in the world. Most of the firefly species produce short and rhythmic flashes. The pattern of flashes is unique for a particular species. A firefly's flash is mainly act as a signal system to attract mating partners and to attract potential prey. Flashes also serve as a protective warning mechanism. The following three idealized rules are considered for describing the FA [70].

1. All fireflies are unisex so that one firefly will be attracted to other fireflies regardless of their sex
2. Attractiveness is proportional to their brightness, thus for any two flashing fireflies, the less bright one will move toward the brighter one. The attractiveness is proportional to the brightness and they both decrease as their distance increases. If there is no brighter one than a particular firefly, it will move randomly
3. The brightness of a firefly is affected or determined by the landscape of the objective function. For a maximization problem, the brightness may be proportional to the objective function value. For the minimization problem the brightness may be the reciprocal of the objective function value.

The Pseudo code of the FA is given in Fig. 2.

4.1 Attractiveness of a Firefly

The attractiveness of a firefly is determined by its light intensity. The attractiveness may be calculated by using the Eq. (3).

Fig. 2 Pseudo code of the firefly algorithm

```
Objective function f(x), x = (x_1, ..., x_d)^T
Generate initial population of fireflies x_i (i = 1, 2,..., n)
Light intensity I_i at x_i is determined by f(x_i)
Define light absorption coefficient γ
While (t <MaxGeneration)
for i = 1 : n all n fireflies
for j = 1 : i all n fireflies
if (I_j > I_i ), Move firefly i towards j in d-dimension; end if
Attractiveness varies with distance r via exp [–γr^2]
Evaluate new solutions and update light intensity
end for j
end for i
Rank the fireflies and find the current best
end while
Postprocess results and visualization
```

$$\beta(\rho) = \beta_0 e^{-\gamma r^2} \tag{3}$$

where β is the attractiveness of a firefly and γ is the light absorption coefficient.

4.2 Distance Between Two Fireflies

The distance between any two fireflies k and l at X_k and X_l is the Cartesian distance using the Eq. (4).

$$r_{kl} = \|X_k - X_l\| = \sqrt{\sum_{k=1}^{d} (X_{k,o} - X_{l,o})^2} \tag{4}$$

4.3 Movement of a Firefly

The movement of a firefly k that is attracted to another more attractive firefly l is determined by the Eq. (5).

$$X_k = X_k + \beta_0 e^{-\gamma r^2 kl}(X_l - X_k) + \alpha\left(rand - \frac{1}{2}\right) \tag{5}$$

where α is the randomization parameter and rand is a number uniformly drawn from the interval [0, 1].

4.4 Discrete Firefly Algorithm (DFA)

The FA has been originally developed for solving the continuous optimization problems. The FA cannot be applied directly to solve the discrete optimization problems. In this chapter, the smallest position value (SPV) rule described by Bean [5] is used to enable the continuous FA to be applied to solve the discrete HFS scheduling problems. For this, a discrete firefly algorithm (DFA) is proposed. More detail about the DFA can be found in [35].

4.5 Implementation of the DFA for Flow Shop Scheduling Problems

This section illustrates how the DFA is applied for solving the flow shop scheduling problems.

4.5.1 Solution Representation

Solution representation is one of the most important issues in designing a DFA. The solution search space consists of n dimensions as n number of jobs is considered in this book chapter. Each dimension represents a job. The vector $X_i^t = (X_{i1}^t, X_{i2}^t, \ldots, X_{in}^t)$ represents the continuous position values of fireflies in the search space. The SPV rule is used to convert the continuous position values of the fireflies to the discrete job permutation. The solution representation of a firefly with 6 jobs is illustrated in Table 1

The smallest position value is $x_{i4}^t = 0.07$and the dimension $j = 4$ is assigned to be the first job in the permutation according to the SPV rule. The second smallest position value is $x_{i3}^t = 0.22$ and the dimension $j = 3$ is assigned to be the second job in the permutation. Similarly, all the jobs are assigned in the permutation.

4.5.2 Population Initialization

In most of the meta-heuristics, the initial population is generated at random. In the DFA the initial population is also generated at random. The continuous values of positions are generated randomly using a uniform random number between 0 and 1.

Table 1 Solution representation of a FA

	Dimension j						
	1	2	3	4	5	6	7
x_{ij}	0.83	0.94	0.22	0.07	0.74	0.61	0.96
jobs	5	6	2	1	4	3	7

4.5.3 Solution Updation

By using the permutation, each firefly is evaluated to determine the objective function value. The objective function value of each firefly is associated with the light intensity of the corresponding firefly. A firefly with less brightness is attracted and moved to a firefly with more brightness. The attractiveness of the firefly is determined using the Eq. (3). The distance between each two fireflies is determined by the Eq. (4). The SPV rule is applied to obtain the job permutation. The attractiveness is calculated for each firefly. Then, the movement of the firefly is determined by the Eq. (5) depending on the attractiveness of the firefly. The above steps are repeated until the termination criterion is met.

5 Computational Results

The proposed algorithm was coded in C++ and run on a PC with an Intel Core Duo 2.4 GHz CPU, 2 GB RAM, running Windows XP. Simulation experiments are performed with different parameter settings to evaluate the performance of the FA. The factor levels for the design of experiments are given in Table 2.

Hence we conduct $3 \times 3 \times 1 \times 1 \times 1 \times 3 \times 3 \times 3 = 243$ experiments to evaluate the performance of the proposed algorithm. Each problem is tested with 20 replications. We compare the performance of the proposed discrete firefly algorithm with the genetic algorithm (GA), ant colony optimization (ACO) algorithm, cuckoo search (CS), particle swarm optimization (PSO) and the simulated annealing (SA)

Table 2 Factor levels for the design of experiments

Sl. No.	Factors	Levels
1	Number of jobs	20, 50 and 100
2	Number of machines	2, 5 and 10
3	Processing time distribution	Uniform (1–100)
4	Setup times	Uniform (0–10)
5	Due date	0.5–1.2 times processing time
6	Attractiveness of a firefly β_0	0.0 (low)
		0.5 (medium)
		1.0 (high)
7	Light absorption coefficient γ	0.5 (low)
		0.75 (medium)
		1.0 (high)
8	Randomization parameter α	0.0 (low)
		0.5 (medium)
		1.0 (high)

Table 3 Result comparison of different algorithms

Sl. No.	Number of jobs	Number of machines	β_0	γ	α	MRDI					
						SA	ACO	GA	CS	PSO	DFA
1	20	2	0	0.50	0	8.00	7.85	6.62	2.21	4.23	0.00
2	20	2	0	0.75	0	8.35	7.25	6.69	2.77	4.41	0.00
3	20	2	0	1.00	0	8.95	7.46	6.54	2.07	4.62	0.00
4	20	2	0	0.50	0.50	8.15	7.70	6.51	2.39	4.89	0.00
5	20	2	0	0.75	0.50	8.26	7.48	6.74	2.19	4.60	0.00
6	20	2	0	1.00	0.50	8.55	7.05	6.85	2.91	4.51	0.00
7	20	2	0	0.50	1.00	8.42	7.76	6.83	2.92	4.97	0.00
8	20	2	0	0.75	1.00	8.14	7.80	6.43	2.36	4.38	0.00
9	20	2	0	1.00	1.00	8.62	7.31	6.53	2.18	4.92	0.00
10	20	5	0.50	0.50	0	8.94	7.33	6.88	2.13	4.06	0.00
11	20	5	0.50	0.75	0	8.45	7.56	6.68	2.98	4.81	0.00
12	20	5	0.50	1.00	0	8.32	7.91	6.34	2.34	4.63	0.00
13	20	5	0.50	0.50	0.50	8.28	7.24	6.29	2.56	4.76	0.00
14	20	5	0.50	0.75	0.50	8.56	7.36	6.82	2.82	4.12	0.00
15	20	5	0.50	1.00	0.50	8.72	7.42	6.18	2.64	4.14	0.00
16	20	5	0.50	0.50	1.00	8.16	7.31	6.54	2.57	4.48	0.00
17	20	5	0.50	0.75	1.00	8.43	7.52	6.62	2.73	4.51	0.00
18	20	5	0.50	1.00	1.00	8.32	7.75	6.41	2.32	4.78	0.00
19	20	10	1.00	0.50	0	8.46	7.33	6.50	2.56	4.51	0.00
20	20	10	1.00	0.75	0	8.45	7.50	6.39	2.72	4.52	0.00
21	20	10	1.00	1.00	0	8.24	7.70	6.24	2.82	4.55	0.00
22	20	10	1.00	0.50	0.50	8.56	7.63	6.63	2.68	4.63	0.00
23	20	10	1.00	0.75	0.50	8.46	7.29	6.61	2.64	4.53	0.00

(continued)

Table 3 (continued)

Sl. No.	Number of jobs	Number of machines	β_0	γ	α	MRDI					
						SA	ACO	GA	CS	PSO	DFA
24	20	10	1.00	1.00	0.50	8.48	7.58	6.41	2.34	4.62	0.00
25	20	10	1.00	0.50	1.00	8.53	7.91	6.59	2.48	4.75	0.00
26	20	10	1.00	0.75	1.00	8.63	7.57	6.64	2.68	4.60	0.00
27	20	10	1.00	1.00	1.00	8.64	7.33	6.67	2.70	4.62	0.00
28	50	2	0	0.50	0	8.56	7.29	6.38	2.57	4.48	0.00
29	50	2	0	0.75	0	8.43	7.69	6.55	2.46	4.61	0.00
30	50	2	0	1.00	0	8.36	7.45	6.72	2.92	4.68	0.00
31	50	2	0	0.50	0.50	8.56	7.66	6.59	2.68	4.45	0.00
32	50	2	0	0.75	0.50	8.08	7.94	6.51	2.67	4.59	0.00
33	50	2	0	1.00	0.50	8.12	7.35	6.40	2.48	4.59	0.00
34	50	2	0	0.50	1.00	8.32	7.74	6.38	2.45	4.57	0.00
35	50	2	0	0.75	1.00	8.46	7.88	6.41	2.37	4.59	0.00
36	50	2	0	1.00	1.00	8.36	7.33	6.59	2.94	4.55	0.00
37	50	5	0.50	0.50	0	8.56	7.28	6.34	2.69	4.66	0.00
38	50	5	0.50	0.75	0	8.00	7.71	6.45	2.54	4.56	0.00
39	50	5	0.50	1.00	0	8.35	7.60	6.52	2.73	4.62	0.00
40	50	5	0.50	0.50	0.50	8.95	7.43	6.55	2.68	4.70	0.00
41	50	5	0.50	0.75	0.50	8.15	7.21	6.54	2.54	4.67	0.00
42	50	5	0.50	1.00	0.50	8.26	7.83	6.67	2.63	4.44	0.00
43	50	5	0.50	0.50	1.00	8.46	7.78	6.62	2.58	4.56	0.00
44	50	5	0.50	0.75	1.00	8.45	7.13	6.72	2.63	4.81	0.00
45	50	5	0.50	1.00	1.00	8.24	7.89	6.54	2.56	4.42	0.00
46	50	10	1.00	0.50	0	8.00	7.58	6.54	2.83	4.58	0.00

(continued)

Table 3 (continued)

Sl. No.	Number of jobs	Number of machines	β_0	γ	α	MRDI					
						SA	ACO	GA	CS	PSO	DFA
47	50	10	1.00	0.75	0	8.35	7.52	6.50	2.67	4.59	0.00
48	50	10	1.00	1.00	0	8.95	7.64	6.40	2.48	4.60	0.00
49	50	10	1.00	0.50	0.50	8.15	7.20	6.73	2.78	4.54	0.00
50	50	10	1.00	0.75	0.50	8.26	7.84	6.50	2.48	4.57	0.00
51	50	10	1.00	1.00	0.50	8.43	7.53	6.45	2.69	4.48	0.00
52	100	10	1.00	0.50	1.00	8.32	7.82	6.40	2.78	4.37	0.00
53	100	10	1.00	0.75	1.00	8.46	7.18	6.63	2.82	4.65	0.00
54	100	10	1.00	1.00	1.00	8.45	7.40	6.54	2.49	4.65	0.00
55	100	2	0	0.50	0	8.24	7.34	6.53	2.68	4.49	0.00
56	100	2	0	0.75	0	8.00	7.23	6.67	2.58	4.65	0.00
57	100	2	0	1.00	0	8.53	7.64	6.56	2.76	4.66	0.00
58	100	2	0	0.50	0.50	8.59	7.54	6.65	2.97	4.64	0.00
59	100	2	0	0.75	0.50	8.21	7.42	6.36	2.67	4.46	0.00
60	100	2	0	1.00	0.50	8.62	7.46	6.53	2.91	4.63	0.00
61	100	2	0	0.50	1.00	8.55	7.64	6.62	2.68	4.77	0.00
62	100	2	0	0.75	1.00	8.42	7.84	6.34	2.59	4.62	0.00
63	100	2	0	1.00	1.00	8.14	7.43	6.54	2.67	4.58	0.00
64	100	5	0.50	0.50	0	8.62	7.36	6.59	2.46	4.49	0.00
65	100	5	0.50	0.75	0	8.94	7.46	6.47	2.73	4.43	0.00
66	100	5	0.50	1.00	0	8.46	7.62	6.54	2.91	4.51	0.00
67	100	5	0.50	0.50	0.50	8.45	7.43	6.54	2.67	4.71	0.00
68	100	5	0.50	0.75	0.50	8.24	7.36	6.64	2.68	4.45	0.00
69	100	5	0.50	1.00	0.50	8.00	7.56	6.46	2.74	4.51	0.00

(continued)

Table 3 (continued)

Sl. No.	Number of jobs	Number of machines	β_0	γ	α	MRDI					
						SA	ACO	GA	CS	PSO	DFA
70	100	5	0.50	0.50	1.00	8.35	7.28	6.58	2.68	4.61	0.00
71	100	5	0.50	0.75	1.00	8.95	7.12	6.74	2.56	4.53	0.00
72	100	5	0.50	1.00	1.00	8.15	7.32	6.64	2.74	4.44	0.00
73	100	10	1.00	0.50	0	8.26	7.46	6.55	2.82	4.59	0.00
74	100	10	1.00	0.75	0	8.55	7.36	6.50	2.72	4.67	0.00
75	100	10	1.00	1.00	0	8.42	7.56	6.61	2.74	4.53	0.00
76	100	10	1.00	0.50	0.50	8.14	7.82	6.48	2.68	4.59	0.00
77	100	10	1.00	0.75	0.50	8.62	7.64	6.65	2.73	4.46	0.00
78	100	10	1.00	1.00	0.50	8.94	7.20	6.57	2.77	4.57	0.00
79	100	10	1.00	0.50	1.00	8.54	7.84	6.34	2.74	4.77	0.00
80	100	10	1.00	0.75	1.00	8.42	7.53	6.58	2.68	4.43	0.00
81	100	10	1.00	1.00	1.00	8.12	7.82	6.70	2.67	4.58	0.00

algorithms addressed in the literature. Mean Relative Deviation Index (MRDI) is used as a performance measure to compare the performance of different algorithms. MRDI is calculated as given below:

$$MRDI = \sum_{l=1}^{R} \frac{|Z^* - Z_{META}|}{Z^*} \times 100/R \tag{6}$$

Where,

Z^*	best objective function value
Z_{META}	objective function value obtained by different metaheuristic algorithms
R	number of runs (20)

Lower value of MRDI value indicates the better performance of the algorithm. The comparison results of some sample problems are presented in Table 3.

From the result table, it can be easily concluded that the proposed DFA is better than many other algorithms addressed in the literature.

6 Conclusions

A discrete firefly algorithm is presented in this chapter to minimize the weighted sum of makespan, mean flow time, mean tardiness and number of tardy jobs for flow shop scheduling problems. The proposed DFA has been tested over a set of random problem instances with different parameter settings and the results have been compared with other metaheuristics addressed in the literature. It is concluded that the DFA provides better results than many other metaheuristics. This work may be extended in many directions. The algorithm can also be applied to solve real industrial scheduling problems. The research may also be conducted for other types of scheduling problems. It would be interesting to conduct the computational experiments with several other parameters values to determine the optimal parameters of the firefly algorithm.

References

1. Armentano, V.A., Ronconi, D.P.: Tabu search for total tardiness minimization in flowshop scheduling problems. Comp. Oper. Res. **26**(3), 219–235 (1999)
2. Arroyo, J.E.C., Armentano, V.A.: Genetic local search for multi-objective flowshop scheduling problems. Eur. J. Oper. Res. **167**, 717–738 (2005)
3. Baker, K.R.: Introduction to sequencing and scheduling. Wiley, New York (1974)
4. Banati, H., Bajaj, M.: Firefly based feature selection approach. Int. J. Comp. Sci. Issues **8**(4), 473–480 (2011)

5. Bean, J.C.: Genetic algorithms and random keys for sequencing and optimization. ORSA J. comput. **6**(2), 154–160 (1994)
6. Boxma, O.J., Forst, F.G.: Minimizing the expected weighted number of tardy jobs in stochastic flow shops. Oper. Res. Lett. **5**(3), 119–126 (1986)
7. Chakraborty, U.K., Turvey, K.P.: Floating-point to integer mapping schemes in differential evolution for permutation flow shop scheduling. Int. J. Bio-Inspired Comput. **2**(3), 183–204 (2010)
8. Chandrasekaran, K., Simon, S.P.: Network and reliability constrained unit commitment problem using binary real coded firefly algorithm. Int. J. Electr. Power Energy Syst. **43**(1), 921–932 (2012)
9. Chowdhury, A., Ghosh, A., Sinha, S., Das, S., Ghosh, A.: A novel genetic algorithm to solve travelling salesman problem and blocking flow shop scheduling problem. Int. J. Bio-Inspired Comput. **5**(5), 303–314 (2013)
10. Coelho, L.D.S., Mariani, V.C.: Firefly algorithm approach based on chaotic Tinkerbell map applied to multivariable PID controller tuning. Comput. Math. Appl. **64**(8), 2371–2382 (2012)
11. Corwin, B.D., Esogbue, A.O.: Two machine flow shop scheduling problems with sequence dependent setup times: a dynamic programming approach. Naval Res. Logistics Q. **21**(3), 515–524 (1974)
12. Czapiński, M.: Parallel simulated annealing with genetic enhancement for flowshop problem with C_{sum}. Comput. Ind. Eng. **59**(4), 778–785 (2010)
13. Dekhici, L., Borne, P., Khaled, B.: Firefly algorithm for economic power dispatching with pollutants emission. Informatica Economică **16**(2), 45–57 (2012)
14. Dong, X., Huang, H., Chen, P.: An iterated local search algorithm for the permutation flowshop problem with total flowtime criterion. Comput. Oper. Res. **36**(5), 1664–1669 (2009)
15. Fister, I., Fister Jr, I., Yang, X.S., Brest, J.: A comprehensive review of firefly algorithms. Swarm Evol. Comput. **13**, 34–46 (2013)
16. França, P.M., Gupta, J.N., Mendes, A.S., Moscato, P., Veltink, K.J.: Evolutionary algorithms for scheduling a flowshop manufacturing cell with sequence dependent family setups. Comput. Ind. Eng. **48**(3), 491–506 (2005)
17. Gandomi, A.H., Yang, X.S., Alavi, A.H.: Mixed variable structural optimization using firefly algorithm. Comput. Struct. **89**(23), 2325–2336 (2011)
18. Gao, K.Z., Pan, Q.K., Li, J.Q.: Discrete harmony search algorithm for the no-wait flow shop scheduling problem with total flow time criterion. Int. J. Adv. Manuf. Technol. **56**(5–8), 683–692 (2011)
19. Gao, K., Pan, Q., Suganthan, P.N., Li, J.: Effective heuristics for the no-wait flow shop scheduling problem with total flow time minimization. Int. J. Adv. Manuf. Technol. **66**(9–12), 1563–1572 (2013)
20. Gupta, J.N., Darrow, W.P.: The two-machine sequence dependent flowshop scheduling problem. Eur. J. Oper. Res. **24**(3), 439–446 (1986)
21. Gupta, J.N.D., Chen, C.L., Yap, L.Y., Deshmukh, H.: Designing a tabu search algorithm to minimize total flow time in a flow shop. Arab. J. Sci. Eng. **25**(1), 79–94 (2000)
22. Gupta, J.N., Neppalli, V.R., Werner, F.: Minimizing total flow time in a two-machine flowshop problem with minimum makespan. Int. J. Prod. Econ. **69**(3), 323–338 (2001)
23. Gupta, J.N., Stafford Jr, E.F.: Flowshop scheduling research after five decades. Eur. J. Oper. Res. **169**(3), 699–711 (2006)
24. Horng, M.H.: Vector quantization using the firefly algorithm for image compression. Expert Syst. Appl. **39**(1), 1078–1091 (2012)
25. Ignall, E., Schrage, L.: Application of the branch and bound technique to some flow-shop scheduling problems. Oper. Res. **13**(3), 400–412 (1965)
26. Jarboui, B., Eddaly, M., Siarry, P.: An estimation of distribution algorithm for minimizing the total flowtime in permutation flowshop scheduling problems. Comput. Oper. Res. **36**(9), 2638–2646 (2009)
27. Johnson, S.M.: Optimal two and three stage production schedules with setup times included. Naval Res. Logistics Q. **1**(1), 61–68 (1954)

28. Kazemzadeh, A.S., Kazemzadeh, A.S.: Optimum design of structures using an improved firefly algorithm. Int. J. Optim. Civ. Eng. **2**, 327–340 (2011)
29. Khadwilard, A., Chansombat, S., Thepphakorn, T., Thapatsuwan, P., Chainat, W., Pongcharoen, P.: Application of firefly algorithm and its parameter setting for job shop scheduling. J. Ind. Technol. **8**, 49–58 (2012)
30. Khalili, M., Tavakkoli-Moghaddam, R.: A multi-objective electromagnetism algorithm for a bi-objective flowshop scheduling problem. J. Manuf. Syst. **31**(2), 232–239 (2012)
31. Kim, Y.D.: Minimizing total tardiness in permutation flowshops. Eur. J. Oper. Res. **85**(3), 541–555 (1995)
32. Liao, C.J., Tseng, C.T., Luarn, P.: A discrete version of particle swarm optimization for flowshop scheduling problems. Comput. Oper. Res. **34**(10), 3099–3111 (2007)
33. Liu, C.P., Ye, C.M.: Solving Permutation Flow Shop Scheduling Problem by firefly Algorithm. Ind. Eng. Manage. **17**(3), 56–59 (2012)
34. Marichelvam, M.K.: An improved hybrid Cuckoo Search (IHCS) metaheuristics algorithm for permutation flow shop scheduling problems. Int. J. Bio-Inspired Comput. **4**(4), 200–205 (2012)
35. Marichelvam, M.K., Prabaharan, T., Yang, X.S.: A discrete firefly algorithm for the multi-objective hybrid flowshop scheduling problems. IEEE Trans. Evol. Comput. **18**(2), 301–305 (2014)
36. Miguel, L.F.F., Lopez, R.H., Miguel, L.F.F.: Multimodal size, shape, and topology optimisation of truss structures using the firefly algorithm. Adv. Eng. Softw. **56**, 23–37 (2013)
37. Murata, T., Ishibuchi, H., Tanaka, H.: Multi-objective genetic algorithm and its applications to flowshop scheduling. Comput. Ind. Eng. **30**(4), 957–968 (1996)
38. Nagano, M.S., Moccellin, J.V.: Reducing mean flow time in permutation flow shop. J. Oper. Res. Soc. **59**(7), 939–945 (2008)
39. Naderi, B., Tavakkoli-Moghaddam, R., Khalili, M.: Electromagnetism-like mechanism and simulated annealing algorithms for flowshop scheduling problems minimizing the total weighted tardiness and makespan. Knowl.-Based Syst. **23**(2), 77–85 (2010)
40. Neppalli, V.R., Chen, C.L., Gupta, J.N.: Genetic algorithms for the two-stage bicriteria flowshop problem. Eur. J. Oper. Res. **95**(2), 356–373 (1996)
41. Osman, I.H., Potts, C.N.: Simulated annealing for permutation flow-shop scheduling. Omega **17**(6), 551–557 (1989)
42. Pan, Q.K., Fatih Tasgetiren, M., Suganthan, P.N., Chua, T.J.: A discrete artificial bee colony algorithm for the lot-streaming flow shop scheduling problem. Inf. Sci. **181**(12), 2455–2468 (2011)
43. Parthasarathy, S., Rajendran, C.: An experimental evaluation of heuristics for scheduling in a real-life flowshop with sequence-dependent setup times of jobs. Int. J. Prod. Econ. **49**(3), 255–263 (1997)
44. Pinedo, M.: Scheduling: theory, algorithms, and systems. Prentice-Hall, New Jersey (2002)
45. Rahimi-Vahed, A.R., Mirghorbani, S.M.: A multi-objective particle swarm for a flow shop scheduling problem. J. Comb. Optim. **13**(1), 79–102 (2007)
46. Rajendran, C.: Heuristic algorithm for scheduling in a flowshop to minimize total flowtime. Int. J. Prod. Econ. **29**(1), 65–73 (1993)
47. Rajendran, C., Ziegler, H.: An efficient heuristic for scheduling in a flowshop to minimize total weighted flowtime of jobs. Eur. J. Oper. Res. **103**(1), 129–138 (1997)
48. Rajendran, C., Ziegler, H.: Scheduling to minimize the sum of weighted flowtime and weighted tardiness of jobs in a flowshop with sequence-dependent setup times. Eur. J. Oper. Res. **149**(3), 513–522 (2003)
49. Rajendran, C., Ziegler, H.: Ant-colony algorithms for permutation flowshop scheduling to minimize makespan/total flowtime of jobs. Eur. J. Oper. Res. **155**(2), 426–438 (2004)
50. Ravindran, D., Selvakumar, S.J., Sivaraman, R., Haq, A.N.: Flow shop scheduling with multiple objective of minimizing makespan and total flow time. Int. J. Adv. Manuf. Technol. **25**(9–10), 1007–1012 (2005)

51. Rios-Mercado, R.Z., Bard, J.F.: Heuristics for the flow line problem with setup costs. Eur. J. Oper. Res. **110**(1), 76–98 (1998)
52. Ruiz, R., Stützle, T.: A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. Eur. J. Oper. Res. **177**(3), 2033–2049 (2007)
53. Ruiz, R., Maroto, C., Alcaraz, J.: Solving the flowshop scheduling problem with sequence dependent setup times using advanced metaheuristics. Eur. J. Oper. Res. **165**(1), 34–54 (2005)
54. Salmasi, N., Logendran, R., Skandari, M.R.: Total flow time minimization in a flowshop sequence-dependent group scheduling problem. Comput. Oper. Res. **37**(1), 199–212 (2010)
55. Sayadi, M., Ramezanian, R., Ghaffari-Nasab, N.: A discrete firefly meta-heuristic with local search for makespan minimization in permutation flow shop scheduling problems. Int. J. Ind. Eng. Comput. **1**(1), 1–10 (2010)
56. Senthilnath, J., Omkar, S.N., Mani, V.: Clustering using firefly algorithm: performance study. Swarm Evol. Comput. **1**(3), 164–171 (2011)
57. Simons Jr, J.V.: Heuristics in flow shop scheduling with sequence dependent setup times. Omega **20**(2), 215–225 (1992)
58. Srikar, B.N., Ghosh, S.: A MILP model for the n-job, m-stage flowshop with sequence dependent set-up times. Int. J. Prod. Res. **24**(6), 1459–1474 (1986)
59. Tang, L., Liu, J.: A modified genetic algorithm for the flow shop sequencing problem to minimize mean flow time. J. Intell. Manuf. **13**(1), 61–67 (2002)
60. TT'kindt, V., Monmarché, N., Tercinet, F., Laügt, D.: An ant colony optimization algorithm to solve a 2-machine bicriteria flowshop scheduling problem. Eur. J. Oper. Res. **142**(2), 250–257 (2002)
61. Tasgetiren, M.F., Liang, Y.C., Sevkli, M., Gencyilmaz, G.: A particle swarm optimization algorithm for makespan and total flowtime minimization in the permutation flowshop sequencing problem. Eur. J. Oper. Res. **177**(3), 1930–1947 (2007)
62. Tasgetiren, M.F., Pan, Q.K., Suganthan, P.N., Chen, A.H.: A discrete artificial bee colony algorithm for the total flowtime minimization in permutation flow shops. Inf. Sci. **181**(16), 3459–3475 (2011)
63. Tavakkoli-Moghaddam, R., Rahimi-Vahed, A., Mirzaei, A.H.: A hybrid multi-objective immune algorithm for a flow shop scheduling problem with bi-objectives: weighted mean completion time and weighted mean tardiness. Inf. Sci. **177**(22), 5072–5090 (2007)
64. Tseng, L.Y., Lin, Y.T.: A genetic local search algorithm for minimizing total flowtime in the permutation flowshop scheduling problem. Int. J. Prod. Econ. **127**(1), 121–128 (2010)
65. Vahedi Nouri, B., Fattahi, P., Ramezanian, R.: Hybrid firefly-simulated annealing algorithm for the flow shop problem with learning effects and flexible maintenance activities. Int. J. Prod. Res. **51**(12), 3501–3515 (2013)
66. Varadharajan, T.K., Rajendran, C.: A multi-objective simulated-annealing algorithm for scheduling in flowshops to minimize the makespan and total flowtime of jobs. Eur. J. Oper. Res. **167**(3), 772–795 (2005)
67. Vempati, V.S., Chen, C.L., Bullington, S.F.: An effective heuristic for flow shop problems with total flow time as criterion. Comput. Ind. Eng. **25**(1), 219–222 (1993)
68. Yang, X.S.: Multiobjective firefly algorithm for continuous optimization. Eng. Comput. **29**(2), 175–184 (2013)
69. Yang, X.S., Sadat Hosseini, S.S., Gandomi, A.H.: Firefly algorithm for solving non-convex economic dispatch problems with valve loading effect. Appl. Softw. Comput. **12**(3), 1180–1186 (2012)
70. Yang, X.S.: Nature-Inspired metaheuristic algorithms. Luniver press, UK (2008)
71. Yagmahan, B., Yenisey, M.M.: Ant colony optimization for multi-objective flow shop scheduling problem. Comput. Ind. Eng. **54**(3), 411–420 (2008)
72. Zhang, Y., Li, X., Wang, Q.: Hybrid genetic algorithm for permutation flowshop scheduling problems with total flowtime minimization. Eur. J. Oper. Res. **196**(3), 869–876 (2009)

Evaluation of Harmony Search and Differential Evolution Optimization Algorithms on Solving the Booster Station Optimization Problems in Water Distribution Networks

Şerife Gökçe and M. Tamer Ayvaz

Abstract Disinfection of water in distribution networks is usually achieved by chlorine injection at outlet of the treatment plants. However, such disinfection applications cause non-uniform chlorine residuals since chlorine decays during its propagation in the network. To maintain chlorine residuals within allowable limits, additional chlorine injection locations called as the booster stations are installed at some strategic locations of distribution networks. Therefore, estimation of the numbers, locations, and injection rates of the booster stations becomes an important problem. For this purpose, this chapter evaluates the performance of Harmony Search (HS) and Differential Evolution (DE) optimization algorithms for solving the booster station optimization problems.

Keywords Booster stations · Differential evolution algorithm · Harmony search algorithm · Water distribution networks

1 Introduction

Disinfection of drinking water distribution networks is crucial task for human health. This task is usually performed by using the chlorine as a disinfector due to its large effects on microorganisms and its low price. Generally, chlorine is injected to the network from outlet of the treatment plants. However, such disinfection applications cause excessive residuals around the injection locations, which may

Ş. Gökçe
Department of Civil Engineering, Afyon Kocatepe University, Afyonkarahisar, Turkey
e-mail: sgokce@aku.edu.tr

M.T. Ayvaz (✉)
Department of Civil Engineering, Pamukkale University, Denizli, Turkey
e-mail: tayvaz@pau.edu.tr

X.-S. Yang (ed.), *Recent Advances in Swarm Intelligence and Evolutionary Computation*,
Studies in Computational Intelligence 585, DOI 10.1007/978-3-319-13826-8_13

lead to some taste and odor problems as well as formation of some carcinogenic disinfection by-products (DBPs). In addition, these injection applications may not maintain the required residual limits throughout the network since chlorine decays by reacting with some materials in both bulk water and pipe wall. Therefore, maintaining the chlorine residuals within the allowable limits (usually between 0.20 and 4.0 mg/l) becomes an important engineering problem. With this purpose, additional chlorine injection locations, called as the booster chlorination stations, are usually installed in the vicinity of the locations with low levels of chlorine residuals. Therefore, identification of the numbers, locations as well as the chlorine injection schedules of the booster stations becomes an important optimization problem to be solved by the decision makers.

In the literature, the solution of the booster station optimization problems was performed by using both deterministic and heuristic optimization approaches [7]. Among them, the pioneering work was performed by Bocelli et al. [3] where the problem of booster station optimization was formulated as a linear programming (LP) problem. The objective of their problem was to minimize the chlorine injection rates of the fixed booster locations by maintaining the chlorine residual limits throughout the network. Their results indicated that chlorine concentrations at consumer points are the linear functions of the chlorine injection rates in case of the first-order bulk and wall reaction kinetics [6]. By utilizing this relationship, chlorine residuals for a given consumer point and time can be calculated based on a response matrix (RM) approach. Using this feature, Tryby et al. [19] also defined the booster locations as the decision variables of the optimization model and solved by using a mixed integer linear programming (MILP). Propato and Uber [14] formulated the booster station optimization problem as linear least-square (LLS) problem and solved with quadratic programming (QP) approaches. As an extension, they also combined their LLS formulation with the mixed-integer quadratic programming (MIQP) approach to determine the locations of the booster stations together with the corresponding chlorine injection rates [15]. Sert [17] also solved the booster station optimization problem via LP by considering the non-zero initial concentrations at consumer points within the network.

It should be noted that deterministic solution approaches were used in all of the studies given above. Global optimum solutions were obtained by utilizing the RM approach by assuming the first-order bulk and wall reaction kinetics throughout the system. However, if this assumption is satisfied in the network, chlorine residuals cannot be classified as the linear functions of the chlorine injection rates from the booster stations. For such cases, the booster station optimization problem cannot be solved via LP based RM approach, and thus, use of the nonlinear optimization approaches is required. Although nonlinear optimization approaches are effective in finding the global optimum solutions in reasonable times, their efficiency usually depends on the initial values of the decision variables especially for the non-convex solution spaces [1]. Furthermore, since almost all of these approaches require of taking the partial derivatives of the objective and constraint functions, they cannot be used for the cases where taking the partial derivatives is not possible. Therefore, use of the heuristic optimization approaches is usually preferred for solving the

booster station optimization problems. There are several heuristic optimization approaches in the literature, which mimic some natural phenomena. These phenomena include natural selection and evolution in genetic algorithm (GA) [8] and differential evolution (DE) [18], physical annealing process in simulated annealing (SA) [10], social behaviors of birds or fishes in particle swarm optimization (PSO) [9], finding the shortest paths between nest and a food source in ant colony optimization (ACO) [4], and musical improvisation process in harmony search (HS) [5]. Using these approaches, the current literature includes several applications for solving the booster station optimization problems. Munavalli and Mohan Kumar [11], Özdemir and Uçaner [13], Ostfeld and Salamons [12] solved the booster station optimization problems by using different genetic algorithm (GA) based solution approaches. The same problem was also solved by using the ant colony optimization (ACO) by Wang and Guo [21], and particle swarm optimization method by Wang et al. [20].

The main objective of this chapter is to evaluate the performances of two different heuristic optimization approaches, HS and DE, on solving the booster station optimization problems. With this purpose, recent applications of HS and DE are evaluated [6, 7] and the identification results are compared with the other solutions which were obtained by using several deterministic and heuristic optimization approaches. The performance of these two approaches is evaluated on an existing water distribution network by comparing the trade-off between chlorine injection rates and water quality improvements. Identified results indicated that both HS and DE based optimization models not only determined the optimum chlorination plan, but also provided better results than those obtained by different solution approaches in the literature.

2 Harmony Search (HS) Algorithm

Like other heuristic optimization algorithms, HS is also inspired from a heuristic event. However, its main difference from the others is that HS does not get its main philosophy from a natural process, instead, gets from the musical improvisation which occurs when a group of musicians searches for a better state of harmony [2]. This philosophy was first adapted to the optimization problems by Geem et al. [5]. In this adaptation, each musician mimics to a decision variable and the notes in the musicians' memories correspond to the values of the decision variables. When the musicians find the fantastic harmony from their memories, it means, a global optimum solution is obtained using the corresponding decision variables. Note that an aesthetic harmony can be obtained based on the following three musical operations [2]: (i) Playing a note from the harmony memory, (ii) Playing a note randomly from the possible note range, (iii) Playing a note which is close to another one stored in memory. Adaptation of these musical operations into the engineering optimization problems is as follows: (i) New variable values are selected from the harmony memory, (ii) New variable values are randomly selected from the possible

random range, (iii) New variable values are further replaced with other values which are close to the current values. Combination of these three operations allows exploring the search space for finding a global optimum solution. The following computational scheme describes the required solution steps for solving an optimization problem using HS [7]:

Step 1: Generate random solution vectors $(\mathbf{x}^1, \ldots, \mathbf{x}^{\mathrm{HMS}})$ as many as harmony memory size (HMS), then, store them in harmony memory (HM)

Step 2: Generate a new solution vector $(\mathbf{x}')$. For each element $\left(x_i'\right)$:

- With probability of HMCR (harmony memory considering rate, HMCR $\in [0, 1]$), pick the sorted value from HM such that $x_i' \leftarrow x_i^{\mathrm{int}(\mathrm{r}(0,1)\cdot \mathrm{HMS})+1}$ where $r(0, 1)$ is the uniform random number between the specified lower and upper bounds
- With probability of 1-HMCR, pick a random value within the allowed range.

Step 3: Perform additional adjustment if the value in Step 2 came from HM:

- With probability of PAR (pitch adjusting rate, PAR $\in [0, 1]$), change the value x_i' by a small amount such that $x_i' \leftarrow x_i + fw \cdot r(-1, 1)$ where fw is the fret width which can be defined as the amount of the maximum change in pitch adjusting process
- With probability of 1-PAR, do nothing

Step 4: If the value of $\mathbf{x}'$ is better than the worst vector $\mathbf{x}^{\mathbf{worst}}$ in HM, replace $\mathbf{x}^{\mathbf{worst}}$ with $\mathbf{x}'$

Step 5: Repeat from Step 2 to Step 4 until termination.

3 Differential Evolution (DE) Algorithm

DE, proposed by Storn and Price [18], is a population-based heuristic optimization algorithm. It can solve the optimization problems with non-differentiable, non-continuous or noisy solution spaces. Also, it can handle continuous, discrete and integer variables and/or constraint equations. Basically, DE has similar characteristics with the GA in terms of operation and calculation schemes. As in GA, a new individual solution in DE is created by sequentially applying the mutation, crossover and selection operators. Although DE and GA have similar calculation schemes, they have some differences. While DE can solve an optimization problem using the real coded decision variables only, GA can both use the real and binary coded decision variables. Unlike the GA, all the individual solutions in DE are subjected to evolution via genetic operators and the evolved solutions are directly transferred to next generations, if their corresponding objective function values are improved. The basic computational steps of DE can be described as follows [6]:

Step 1: Randomly initialize all agents **x** (e.g. candidate solutions) in the population (NP being the population number).

Step 2: Repeat the following until a termination criterion is met:

- For each agent **x** in the population do:
  - Randomly select three distinct solutions **a**, **b** and **c** from the population
  - Pick a random index $R \in \{1, 2, 3, \ldots, n\}$ (n being the dimension of the problem).
  - Compute the agent's potentially new position $y = [y_1, y_2, y_3, \ldots, y_n]$ as follows:

    For each i, pick a uniformly distributed random number $r_i = U(0, 1)$
    If $r_i < CR$ ($CR \in [0, 1]$ is the crossover rate) or $i = R$ then set $y_i = a_i + F(b_i - c_i)$ ($F \in [0, 2]$ is the differential weight) otherwise set $y_i = x_i$
    In essence, the new position is outcome of binary crossover of agent **x** with intermediate agent $\mathbf{z} = \mathbf{a} + F(\mathbf{b} - \mathbf{c})$

  - If $f(y) < f(x)$ replace the agent in the population with the improved candidate solution, that is replace **x** with **y** in the population

Step 3: Pick the agent from the population that has the highest fitness or lowest cost and return it as best found candidate solution.

4 Problem Formulation

The problem of booster station optimization in water distribution networks can be solved by developing an optimization model [6, 7]. The main objective of this model is to maintain the chlorine residual limits throughout the network by maximizing an objective function consisting two conflicting objectives. While the first objective aims to maximize the volume of "high quality water" which is defined as the volumetric water demand within the specified residual limits, the second objective aims to minimize the total chlorine injection rates from the defined booster locations. These two objectives can be combined in a single objective function as follows [6, 7]:

$$z = \max(\omega_1 f_1 - \omega_2 f_2) \tag{1}$$

$$f_1 = \frac{\sum_{m=t}^{t+n_h-1} \sum_{j=1}^{n_m} V_j^m}{V} \times 100 \tag{2}$$

$$V_j^m = \begin{cases} Q_j^m \Delta t_h & \text{if } c_j^{\min} \le c_j^m \le c_j^{\max} \\ 0 & \text{otherwise} \end{cases} \quad j = 1, 2, 3, \ldots, n_m; \quad m = t, \ldots, t + n_h - 1 \tag{3}$$

$$f_2 = \sum_{i=1}^{n_b} \sum_{k=1}^{n_k} u_i^k \tilde{Q}_i^k \tag{4}$$

where n_m is the number of consumer points where chlorine residuals are controlled, n_h is the number of monitoring time steps, t is the monitoring starting time, V_j^m is the volumetric water demand within the specified residual limits at node j in monitoring period m, V is the total volume of demand over a hydraulic cycle, Q_j^m is the demand at node j in monitoring period m, Δt_h is the length of the monitoring time step, c_j^m is the chlorine residual at monitoring node j and monitoring time m, $c_j^{\min}$ (0.2 mg/l) and $c_j^{\max}$ (4 mg/l) are the lower and upper limits of the chlorine residuals at monitoring node j, n_b is the number of booster stations, n_k is the number of chlorine injection time steps, u_i^k is the chlorine injection rate [ML^{-3}] added from booster station i at injection period k, $\tilde{Q}_i^k$ is the total outflow [L^3T^{-1}] at node i at injection period k, ω_1 and ω_2 are the weighting coefficients.

It can be seen that the main objective function (z) includes two different objective terms. While the first term (f_1) is related with the maximization of percentage of high quality water within chlorine residual limits, the second term (f_2) deals with the minimization of chlorine injection rates. Because of the nature of these two objective functions (f_1 and f_2) are different, weighting coefficients of ω_1 and ω_2 are used for adjusting the contribution of both objectives to the main objective function value (z) [6, 7]. After performing several trials, it is concluded that use of the values of $\omega_1 = 1$ and $\omega_2 = 0.01$ is sufficient to approximately equalize the contribution from both objectives [6, 7].

It can be seen from the mathematical formulation given above, calculation of Eq. (3) requires of knowing the nodal chlorine residuals of c_j^m ($j = 1, 2, 3, \ldots, n_m; \quad m = t, \ldots, t + n_h - 1$) for all the consumer points of the network. A water quality simulation model is required to establish the link between chlorine dosages applied at booster stations and chlorine residuals at the consumer points. As indicated previously, Bocelli et al. [3] showed that chlorine residuals at junctions vary linearly with the injected chlorine concentrations when first-order bulk and wall reaction kinetics are valid throughout the network. Thus, chlorine residuals at junction j and time period m can be calculated by using the linear superposition principle as follows:

$$c_j^m = \sum_{i=1}^{n_b} \sum_{k=1}^{n_k} \alpha_{ij}^{km} u_i^k \quad j = 1, 2, 3, \ldots, n_m; \quad m = t, \ldots, t + n_h - 1 \tag{5}$$

where booster stations and time periods for which chlorine injection is realized are represented by indices i and k, respectively; α_{ij}^{km} is the response coefficient whose

value is calculated from $\alpha_{ij}^{km} = \partial c_j^m \Big/ \partial u_i^k$. Note that the response coefficients for unit chlorine injections from the potential booster stations and time periods are calculated by using the EPANET model [16]. EPANET is a commonly used robust tool to perform hydraulic and water quality simulations in looped distribution networks. It also includes a developer's toolkit to reach EPANET's built-in functions from the different programming environments. By using this feature, developed EPANET model can be run for each optimization cycle to calculate the associated chlorine residuals in the consumer points. Although such integration is practically possible and is implemented in many of the previously conducted studies, it may have high computational cost especially for large networks and/or simulation times. Therefore, the RM approach proposed by Bocelli et al. [3] is considered for calculating the chlorine residuals in this chapter. Note that, RM approach depends on the linear superposition principle and this principle is only applicable when first-order bulk and wall decay reaction kinetics are valid throughout the network [3]. When higher order reaction kinetics is valid in the system, this principle cannot be used due to nonlinear behavior of the residual concentrations with respect to the injected chlorine dosages. In addition, since nodal chlorine concentrations cannot be calculated using the response coefficients, EPANET has to be integrated into the optimization model as indicated as the first option.

5 Numerical Applications

The performance of the developed solution model is evaluated on Cherry Hill-Brushy Plains water distribution network of the South Central Connecticut (USA) Regional Water Authority. This network is the second tutorial network of the EPANET 2.0 model and previously used in many studies regarding booster station optimization [3, 6, 7, 11, 14–17, 19]. The schematic view of the network is given in Fig. 1.

The network includes 1 pumping station (1st junction), 1 storage tank (26th junction), 34 consumer nodes (2nd–25th and 27th–36th junctions) which are connected with 47 links with a total pipe length of 11.26 km. There are also six additional hypothetical nodes (junctions A-F) in Fig. 1 which were used as the potential booster locations in some previous studies [3, 6, 7, 11, 14–17, 19]. For this network, baseline demand values and global demand multipliers of the 34 consumer nodes are shown in Figs. 2a, b. Pumping station at junction 1 pumps a flow rate of 43.81 l/s to the consumer nodes based on the pump multipliers given in Fig. 2c.

The EPANET model is executed with the network data to obtain the hydraulic behavior of the network (see Fig. 3). As can be seen in Fig. 3, the network hydraulic behavior is mainly controlled by the pumping station for the first and the third 6-hour time periods of the day. For the remaining time periods, the consumer nodes are supplied from the tank since the pumping station is switched-off based on Fig. 2c. Due to this change in the network hydraulics, flow directions in the pipes change in each 6-hour time period.

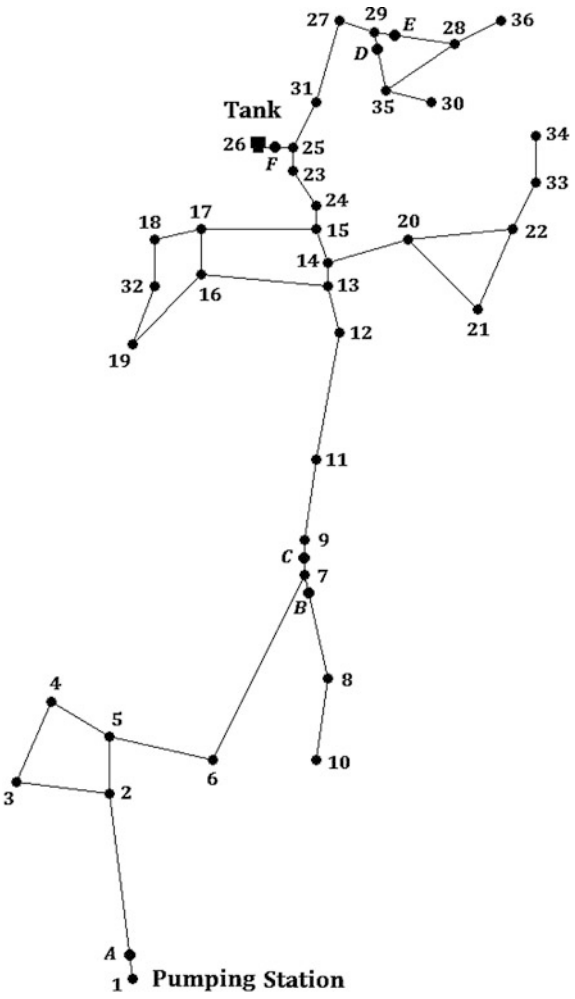


Fig. 1 Layout of the Cherry Hill-Brushy Plains network [3]

As indicated previously, a number of EPANET runs are conducted to determine the response coefficients, α_{ij}^{km} at the consumer nodes. The previous works of Bocelli et al. [3] and Tryby et al. [19] indicate that a periodic hydraulic behavior is required to compute chlorine residuals using the linear superposition principle. The periodic hydraulic behavior is usually obtained when two successive daily chlorine residuals are equal and this can be achieved as a result of a long simulation period [17]. In the current study, a total simulation time of 288 h is used since the exact periodic behavior is obtained after about 264 h. The response coefficients of α_{ij}^{km} are then calculated using concentration values obtained for the following 24 h after the periodic behavior is reached. Note that bulk and wall chlorine decay rate coefficients are assumed to be 0.50 d^{-1} and 0, respectively.

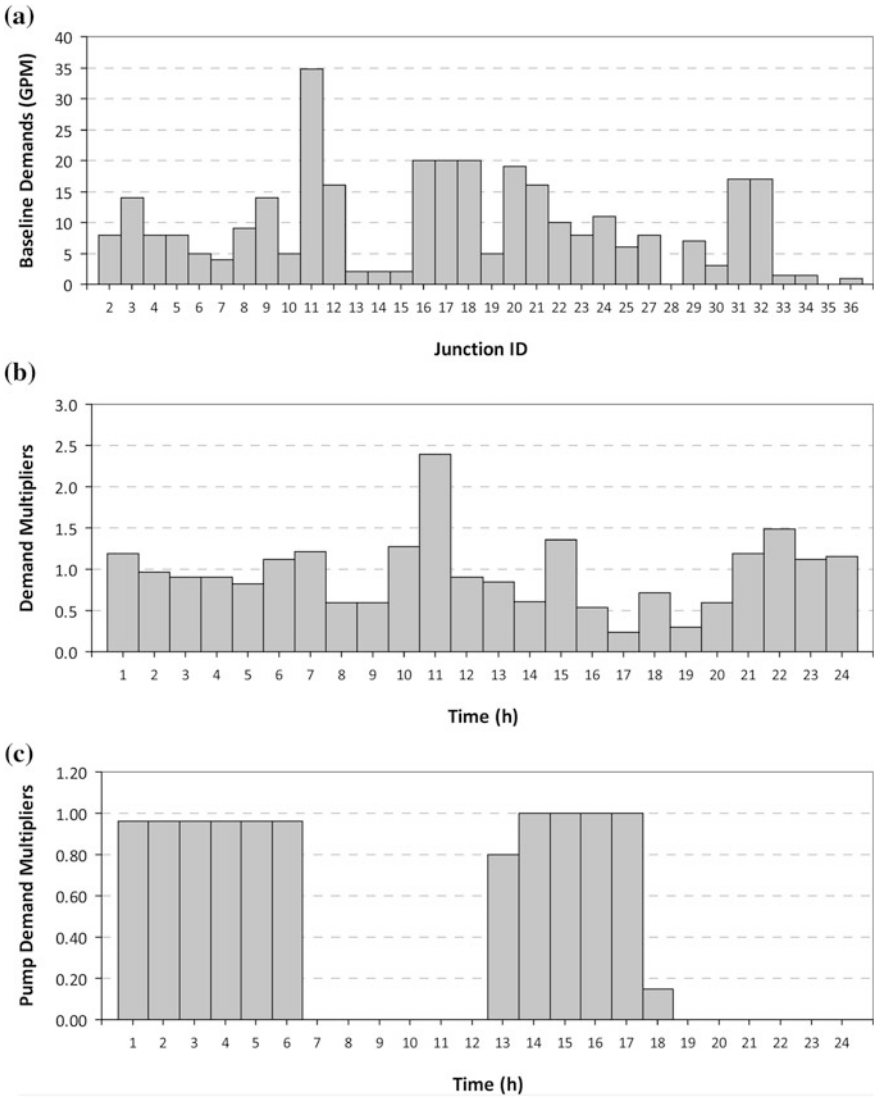


Fig. 2 **a** Baseline demand values of the consumer nodes; **b** global baseline demand multipliers; **c** pump demand multipliers

In the following sections, the identification results of two previously published studies are briefly evaluated [6, 7]. While the first study [7] aimed to determine the chlorine injection rates from the pre-defined booster locations, the second one [6] aimed to determine both locations and chlorine injection rates of the booster stations. Results of both studies are compared with the results of several studies those obtained by using different optimization formulations and approaches in literature.

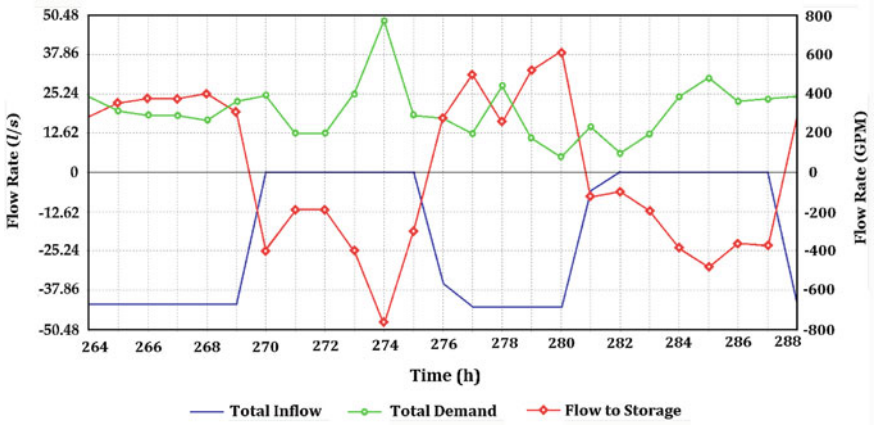


Fig. 3 Hydraulic behavior of the network

5.1 Application 1: Evaluation of HS Based Optimization Model

In this section application of the HS based optimization model by Gökçe and Ayvaz [7] on solving the booster station optimization problem is evaluated. As indicated previously, the main objective of the HS based optimization model is to determine the chlorine injection rates pre-defined booster locations by taking the mathematical formulation between Eqs. (1) and (5) into account. It is assumed that the chlorine is injected to the network from the booster locations of A–F by considering the mass booster (MB) and 4 chlorine injection steps in a day.

For this problem, the performance of the HS based optimization model is evaluated for 8 different HS parameter sets given in Table 1. This is an important step for evaluating the robustness of the optimization models since the quality of the identified solutions in heuristic optimization approaches depends on the selection of their solution parameters. Note that during the optimization process $c_j^{\min}$ and $c_j^{\max}$

Table 1 Different HS solution parameters [7]

Case number	HMS	HMCR	PAR
1	5	0.85	0.40
2	5	0.95	0.50
3	10	0.85	0.40
4	10	0.95	0.50
5	20	0.85	0.40
6	20	0.95	0.50
7	30	0.85	0.40
8	30	0.95	0.50

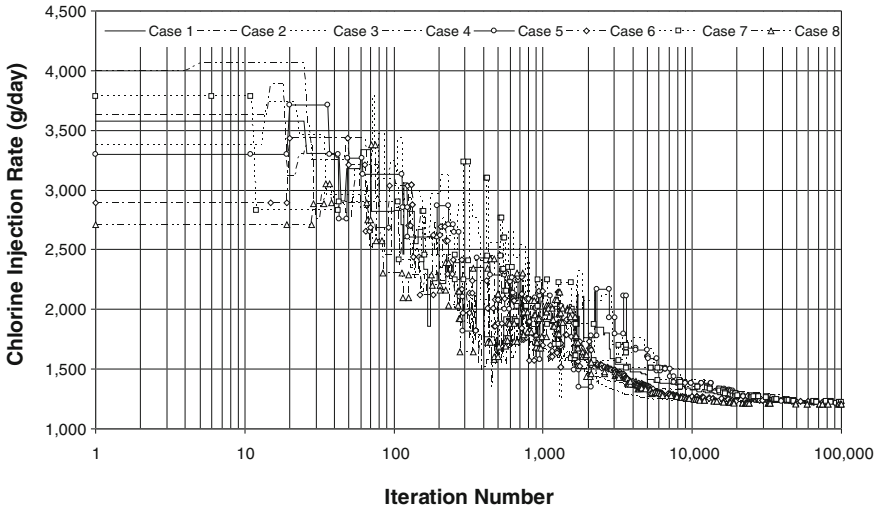


Fig. 4 Convergence plots of the 8 cases in terms of the chlorine injection rates (g/day); (a) Case 1; (b) Case 2; (c) Case 3; (d) Case 4; (e) Case 5; (f) Case 6; (g) Case 7; (h) Case 8 [7]

$(j = 1, 2, 3, \ldots, 34)$ are chosen as 0.20 mg/l and 4.00 mg/l respectively to satisfy both the Safe Drinking Water Act (SDWA) regulations and obtain comparable results with literature.

Figure 4 shows the convergence charts for each HS solution in terms of the final chlorine injection rates (g/day). As can be seen, all the solutions start the search process from different initial solutions and converge to the approximately same solution wherever the optimization begins. Results of the best solution in Fig. 4 are compared in Table 2 with the results obtained by QP and GA based solution approaches in literature. As can be seen from the given results, the final chlorine injection rates have similar magnitudes for all the booster stations and injection periods. The main contribution is supplied from the booster stations A, C and F, and the remaining stations are only used for maintaining the minimum residual condition especially for the far points of the network. For the booster stations A and C, large chlorine injection rates are obtained for the first and the third injection periods. This result is an expected behavior since pumping station is active only for these time periods. On the contrary, large chlorine injections are obtained for booster station F in the second and the fourth injection periods since the network feeds mainly from the storage tank for those time periods. Comparison of the total chlorine injection rates showed that HS based optimization model finds the same injection rate with those obtained by the GA based optimization model. But these two results are higher than the one obtained by the QP based optimization approach. The reason of this situation may be associated with the differences between the used objective functions and/or the nature of the heuristic algorithms. Average chlorine residuals at all the consumer nodes are given in Fig. 5. It can be seen that all the

Table 2 Comparison of the identified results

	Injection period	Booster stations/Chlorine injection rates (mg/min)						Total injection rate (g/day)
		A	B	C	D	E	F	
QP [14]	1	589	7.9	419	0.1	0.1	0	1,176
	2	0	0	0	0.2	0.6	727	
	3	636	4.9	454	0	0.1	8	
	4	0	0.4	0	0.8	1.4	409	
GA [11]	1	599.3	9.8	473.6	0.7	0.4	0	1,205
	2	0	0.7	0	0.3	0.4	713.5	
	3	680.1	4.3	413	0	0.3	47.1	
	4	0	0	0.3	0.7	1	400.7	
HS [7]	1	600.7	9	415.1	0	2.8	2.7	1,205
	2	0	5.7	0.2	0	1.6	732.5	
	3	913.6	4.4	211.8	0	3.9	9.2	
	4	0	0	0	0	9.2	424	

LP Linear Programming; *QP* Quadratic Programming; *GA* Genetic Algorithms) [7]

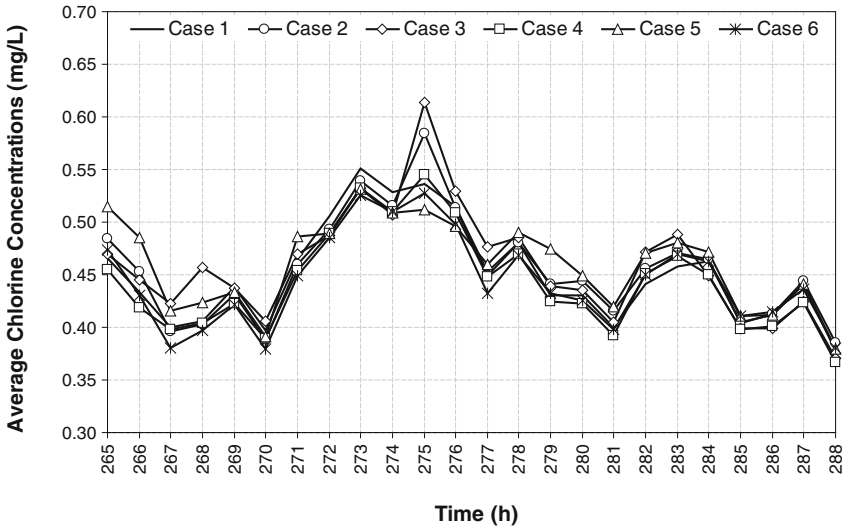


Fig. 5 Averaged chlorine concentrations at all consumer nodes [7]

chlorine residual values are obtained between 0.20 and 4.00 mg/l. This result shows strong constraint satisfaction ability of the HS based solution model during the solution of the booster station optimization problem.

5.2 Application 2: Evaluation of DE Based Optimization Model

In this section application of the DE based optimization model by Gökçe and Ayvaz [6] for solving the booster station optimization problems is evaluated. Unlike the first HS application, the objective of the DE based optimization model is to determine the locations as well as the associated chlorine injection rates of the booster stations. This task is performed by examining the trade-off between the booster station numbers and water quality improvements. For this application, it is assumed that all the nodes in Fig. 1 are considered as the potential flow paced booster (FPB) locations. All the solutions are performed by assuming 1 chlorine injection time steps in a day for comparison purposes. After calculating the composite response coefficients for all the potential booster locations, the DE based optimization model is executed for different booster station numbers in order to evaluate the trade-off between the booster station numbers and water quality improvements.

Figure 6 shows the convergence charts in terms of the chlorine injection rates for the solutions with 1 to 6 booster stations. It is clearly seen that, the final chlorine injection rates decrease when the number of booster stations increases as an expected behavior.

Table 3 compares the DE based optimization model with those obtained by using the MIQP based optimization model by Propato and Uber [15]. Note that Propato and Uber [15] also treated the booster locations as the integer decision variables and solved the related optimization problem through a branch-and-bound solution approach. When the identified results given in Table 3 are compared, it can be seen that, for $n_b = 1$, MIQP based solution model is located the booster station to junction A whereas the DE based model is located to the 2nd junction. Since the 2nd junction

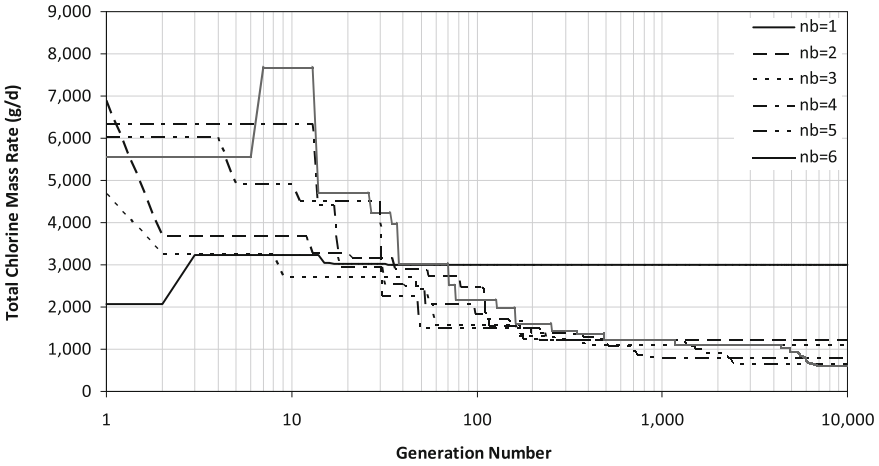


Fig. 6 The final convergence plots in terms of the chlorine injection rates (g/day) [6]

Table 3 Comparison of the locations and injected chlorine concentrations of the solutions

n_b	Injected chlorine dosages (mg/l) ⟨locations of the identified booster stations⟩											
	MIQP [15]						DE [6]					
1	1.835	–	–	–	–	–	1.780	–	–	–	–	–
	⟨A⟩						⟨2⟩					
2	0.531	0.358	–	–	–	–	0.517	0.349	–	–	–	–
	⟨1⟩	⟨26⟩					⟨2⟩	⟨26⟩				
3	0.489	0.727	0.408	–	–	–	0.433	0.372	0.054	–	–	–
	⟨1⟩	⟨26⟩	⟨29⟩				⟨2⟩	⟨26⟩	⟨29⟩			
4	0.360	0.703	0.430	0.436	–	–	0.351	0.209	0.143	0.077	–	–
	⟨1⟩	⟨26⟩	⟨29⟩	⟨33⟩			⟨2⟩	⟨26⟩	⟨29⟩	⟨33⟩		
5	0.360	0.709	0.434	0.432	0.682	–	0.284	0.052	0.220	0.160	0.197	–
	⟨1⟩	⟨26⟩	⟨33⟩	⟨35⟩	⟨*E*⟩		⟨2⟩	⟨8⟩	⟨22⟩	⟨26⟩	⟨29⟩	
6	0.299	0.066	0.191	0.119	0.118	0.187	0.256	0.066	0.661	0.163	0.207	0.019
	⟨1⟩	⟨8⟩	⟨26⟩	⟨33⟩	⟨35⟩	⟨*E*⟩	⟨2⟩	⟨8⟩	⟨22⟩	⟨26⟩	⟨29⟩	⟨32⟩

MIQP Mixed Integer Quadratic Programming; *DE* Differential Evolution Algorithm [6]

Table 4 Comparison of the calculated chlorine injection rates and the water quality responses [6]

n_b	MIQP [15]				DE [6]			
	Chlorine residuals (mg/l)			Chlorine injection rates (g/day)	Chlorine residuals (mg/l)			Chlorine injection rates (g/day)
	Mean	Minimum	Maximum		Mean	Minimum	Maximum	
1	1.06	0.20	3.29	3,116	1.07	0.20	3.52	3,010
2	0.45	0.20	0.55	1,260	0.45	0.20	1.02	1,213
3	0.42	0.20	0.63	1,155	0.41	0.20	0.86	1,094
4	0.31	0.20	0.46	835	0.31	0.20	0.69	799
5	0.31	0.20	0.38	830	0.27	0.20	0.56	645
6	0.27	0.20	0.32	703	0.29	0.20	0.89	614

MIQP Mixed Integer Quadratic Programming; *DE* Differential Evolution Algorithm

is at just downstream of the junction A, this result does not produce a significant change in the chlorine distributions. For $n_b = 2$, both DE and MIQP models determined the same locations (e.g. 26th junction) for the second booster station. For the other solutions, the DE and MIQP based solution found the same or close booster stations with approximately the similar injected chlorine concentrations.

For different booster station numbers, Table 4 compares the chlorine residuals and final injection rates which were calculated using the DE and MIQP based optimization models. As can be seen from Table 4, the mean chlorine residuals have almost the same values for both DE and MIQP. Minimum chlorine residuals are all obtained as 0.20 mg/l, which is previously defined as the minimum residual limit. Although maximum chlorine residual values of DE are all greater than the MIQP, it can be seen that all the results are in the range of the permissible residual limits. When chlorine injection rates are compared, it can be seen that DE model is resulted with the lower chlorine injection rates than MIQP for all the solutions.

6 Conclusions

In this chapter, the performance of HS and DE based optimization models is evaluated on solving the booster station optimization problems. The main objective of HS and DE based optimization model is to maintain the chlorine residual limits throughout the network by determining the optimum chlorination plans. Results of the both solution models are compared with those obtained by using QP, GA, and MIQP based optimization models. Identified results indicated that both HS and DE based optimization models determined identical or better solutions than different approaches in terms of the chlorine injection dosages and water quality improvements.

Acknowledgments The work given in this chapter is supported by The Turkish Academy of Sciences (TÜBA)—The Young Scientists Award Programme (GEBIP). The second author thanks TÜBA for their support of this study.

References

1. Arora, J.S.: Introduction to optimum design. Elsevier Academic Press, San Diego (2004)
2. Ayvaz, M.T.: Solution of groundwater management problems using harmony search algorithm, recent advances in harmony search algorithm. In: Zong Woo Geem (ed) Studies in computational intelligence series. Springer, Berlin (2010b)
3. Bocelli, D.L., Tryby, M.E., Uber, J.G., Rossman, L.A., Zierolf, M.L., Polycarpou, M.M.: Optimal scheduling of booster disinfection in water distribution systems. J. Water Res. Plan. Manage. **124**(2), 99–111 (1998)
4. Dorigo, M., Maniezzo, V., Colorni, A.: The ant system: optimisation by a colony of cooperating agents. IEEE Trans. Syst. Man Cybern. Part B Cybern. **26**(1), 29–41 (1996)
5. Geem, Z.W., Kim, J.H., Loganathan, G.V.: A new heuristic optimization algorithm: harmony search. Simulation **76**(2), 60–68 (2001)

6. Gokce, S., Ayvaz, M.T.: A simulation-optimization model for optimal estimation of the numbers, locations and chlorine injection rates of the booster stations in water distribution networks. In: 11th international conference on hydroinformatics (HIC2014), 17–21 August 2014
7. Gokce, S., Ayvaz, M.T.: Application of harmony search algorithm for solving the booster station optimization problems in water distribution networks. In: International civil engineering & architecture symposium for academicians (ICESA2014), 17–20 May 2014
8. Goldberg, D.E.: Genetic algorithms in search, optimization, and machine learning. Addison-Wesley Pub., Boston (1989)
9. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of the IEEE international conference on neural networks, Piscataway, pp. 1942–1948 (1995)
10. Kirkpatrick, S., Gelatt, C., Vecchi, M.: Optimization by simulated annealing. Science **220** (4598), 671–680 (1983)
11. Munavalli, G.R.: Mohan Kumar M. S.: Optimal scheduling of multiple chlorine sources in water distribution systems. J. Water Res. Plan. Manage. **129**(6), 493–504 (2003)
12. Ostfeld, A., Salomons, E.: Conjunctive optimal scheduling of pumping and booster chlorine injections in water distribution systems. Eng. Optim. **38**(3), 337–352 (2006)
13. Özdemir, O.N., Uçaner, M.E.: Success of booster chlorination for water supply networks with genetic algorithms. J. Hydraul. Res. **43**(3), 267–275 (2005)
14. Propato, M., Uber, J.G.: Linear least-squares formulation for operation of booster disinfection systems. J. Water Res. Plan. Manage. **130**(1), (2004)
15. Propato, M., Uber, J.G.: Booster system design using mixed-integer quadratic programming. J. Water Res. Plan. Manage. **130**(4), 348–352 (2004)
16. Rossman, L.A.: EPANET 2 users manual. EPA/600/R-00/057, U.S. Environmental Protection Agency, Cincinnati (2000)
17. Sert, Ç.: Booster disinfection in water distribution networks. M.Sc. thesis, Middle East Technical University, Ankara (2009)
18. Storn, R., Price, K.: Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. J. Global Optim. **11**(4), 341–359 (1997)
19. Tryby, M.E., Bocelli, D.L., Uber, J.G., Rossman, L.A.: Facility location model for booster disinfection of water supply networks. J. Water Res. Plan. Manage. **128**(5), 322–333 (2002)
20. Wang, H., Guo, W., Xu, J., Gu, H.: A hybrid PSO for optimizing locations of booster chlorination stations in water distribution systems. In: Proceedings of the 2010 International Conference on Intelligent Computation Technology and Automation (ICICTA), vol. 1, pp. 126–129 (2010)
21. Wang, H., Guo, W.:Ant colony optimization for booster chlorination stations of water distribution systems. In: Proceedings of the 2010 International Conference on Computer Application and System Modeling (ICCASM), vol. 1, pp. 166–170 (2010)

Web Document Clustering by Using PSO-Based Cuckoo Search Clustering Algorithm

Moe Moe Zaw and Ei Ei Mon

Abstract With the increasing amount of web information, web document clustering plays an important role in Information Retrieval. This paper presents a PSO-based Cuckoo Search Clustering Algorithm to combine the strengths of Cuckoo Search and Particle Swarm. The solutions of new cuckoos are based on the solutions of PSO. Among these solutions, the algorithm will replace some eggs on lack of fitness with successful solutions until an optimal solution emerges. The proposed hybrid algorithm is tested with a web document benchmark dataset and the results show that it performs well in web document clustering area.

Keywords Cuckoo search · Particle swarm optimization · Clustering algorithm · Web document clustering

1 Introduction

Clustering (or cluster analysis) is one of the main data analysis techniques and it deals with the organization of a set of objects in a multidimensional space into similar groups, called clusters. Objects contained in each cluster are very similar to each other and very dissimilar to objects in other clusters. An example of a clustering is depicted shown in figure. Figure 1 shows the input data and the existing clusters of input data are shown in Fig. 2. The same symbols illustrate that objects belong to the same cluster. Cluster analysis aims to discover objects that have some representative behavior in the collection. So, it is assumed that if a rule is valid for one object, it can also be assumed that that the rule can probably be applied to all the objects that are very similar to it. By using this technique, we can find dense and

M.M. Zaw (✉) · E.E. Mon
University of Technology (Yatanarpon Cyber City), Pyin Oo Lwin, Myanmar
e-mail: moemoezaw@gmail.com

E.E. Mon
e-mail: eieimon.ucsy@gmail.com

X.-S. Yang (ed.), *Recent Advances in Swarm Intelligence and Evolutionary Computation*,
Studies in Computational Intelligence 585, DOI 10.1007/978-3-319-13826-8_14

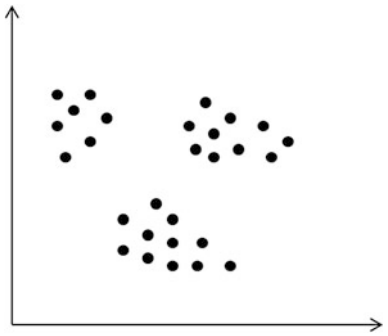


Fig. 1 Clustering example: input data

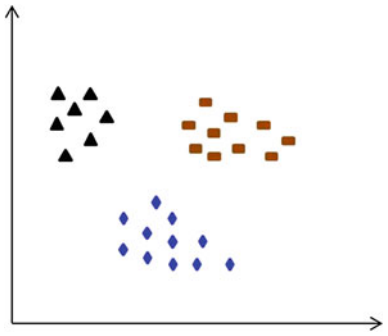


Fig. 2 Clustering example: existing clusters of input data

sparse regions in the data space. And then we can discover hidden similarities, relationships and concepts to group large datasets with regard to the common characteristics of their objects. Clustering is a type of *unsupervised classification*, in which the categories into which the data must be partitioned are not known. So the clustering process means the discovering of these categories [1].

Web document clustering is an important problem for two main reasons. First, we can more easily browse and use the web documents if we cluster a document collection into categories. Automatic categorization is especially important for the World Wide Web because of its huge number of dynamic (time varying) documents and diversity of topics; such features make it difficult to classify pages manually. Second, clustering can improve the performance of search and retrieval on a document collection [2].

Text document clustering groups similar documents in order to form a compact cluster, where documents that are different have separated across into different clusters. However, the definition of a pair of documents being similar or different is not always clear and can vary according to the own problem setting. For example, when research papers are clustered, two documents are assumed as similar if they share similar thematic topics. When clustering is employed on web sites, we are usually more interested in clustering the component pages according to the type of information described in the page. Therefore, by grouping similar types of information sources

together, this kind of clustering can benefit further analysis and utilize of the dataset such as information retrieval and information extraction [3].

The World Wide Web keeps on growing rapidly as a huge resource of information and services. The number of available web documents on the internet is growing rapidly day by day. So, a major challenge in Information Retrieval becomes to find the relevant information on the web. If the World Wide Web continues to grow rapidly, automatic categorization of web pages is essentially needed because of its increasing size and the dramatic content. One of the techniques that can play a special role towards the achievement of this objective is web document clustering.

The existence of an abundance of information, makes information retrieval a tedious process for the average user because of both the dynamic and heterogeneous nature of the Web. Search engines, meta-search engines and Web Directories have been developed so that they can help the users quickly and easily satisfy their information need. This causes the need of new techniques to assist users effectively navigate, trace and organize the available web documents. This is the ultimate goal of finding those best matching their needs. To get the achievement of this objective, one important role is *document clustering*. The increasing importance of document clustering and the variety of its applications cause the development of a wide range of algorithms with different quality [4].

Although there is already much research conducted on the field of web document clustering, it is clear there are still some open issues that call for more research. Therefore, this chapter aims to develop a clustering algorithm and apply in web document clustering area.

2 Literature Review

Clustering can be performed by dividing a set of objects into a specified number of clusters. The main purpose of clustering a set of data is to find inherent structure inside the data and to describe this structure as a set of groups. A wide variety of application areas, such as pattern recognition, marketing, biology, geology and web analysis are the most common used areas of clustering. To apply swarm intelligence in the clustering research area, it can be viewed that the clustering problem is an optimization problem that locates the optimal centroids of the clusters rather than it is an optimal partition finding problem. So, swarm intelligence algorithms are widely applied in clustering area.

The authors in [5] proposed the two new approaches to using PSO to cluster data. It is shown how PSO can he used to find the centroids of a user specified number of clusters. The algorithm is then extended to use K-means clustering to seed the initial swarm. In the second algorithm, PSO refines the clusters formed by K-means. The new PSO algorithms are evaluated on six data sets. And they are compared to the performance of K-means clustering. The experiments show that the

PSO approaches have better convergence to lower quantization errors, and larger inter-cluster distances and smaller intra-cluster distances.

The hybrid PSO and K-means algorithm with a novel alternative metric, called Alternative KPSO-clustering (AKPSO), is developed in [6]. It automatically detects the cluster centers of geometrical structure data sets. It is known that the alternative metric is to have more robust ability than the common-used Euclidean norm. In AKPSO algorithm, the authors consider the special alternative metric to improve the traditional K-means clustering algorithm to deal with various structure data sets. The proposed system is tested on several artificial and real data sets. Simulation results are compared with some well-known clustering methods. The results show that the novel AKPSO method is robust and efficient.

In [7], a Particle Swarm Optimization (PSO) document clustering algorithm is proposed. Contrary to the localized searching of the K-means algorithm, the PSO clustering algorithm can perform a global search in the entire solution space. For experiments, the PSO, K-means and hybrid PSO clustering algorithm are tested on four different text document datasets. The results illustrate that the hybrid PSO algorithm can generate more compact clustering results than the K-means algorithm.

The authors in [8] articulate the unique requirements of Web document clustering and reports on the first evaluation of clustering methods in this domain. A key requirement is that the clusters are based on the short snippets returned by Web search engines. They have found that clusters based on snippets are almost as good as clusters created using the full text of Web documents. To satisfy the stringent requirements of the Web domain, the authors introduce an incremental, linear time (in the document collection size) algorithm called Suffix Tree Clustering (STC). This algorithm creates clusters based on phrases shared between documents. Single Pass, K-means, Buckshot, Fractionation, GAHC and STC algorithms are compared. The results show that not only STC is faster than standard clustering methods in this domain, but also STC is both feasible and potentially beneficial in web document clustering.

Samiksha Goel, Arpita Sharma and Punam Bedi propose Cuckoo Search Clustering Algorithm (CSCA). According to the results, the proposed algorithm is validated on two real time remote sensing satellite- image datasets for extraction of the water body. The CSCA uses Davies-Bouldin index (DBI) as fitness function. Also a new method for generation of new cuckoos used in this algorithm is introduced. The resulting algorithm is conceptually simpler because it takes less parameter than other nature inspired algorithms and after some parameter tuning yields very good results [9].

In [10], the Cuckoo Search Clustering Algorithm based on lévy flight is proposed. In this algorithm, the new cuckoo solution is generated by lévy distribution. The algorithm is applied in web document clustering area and tested with benchmark dataset. The result shows that the algorithm is effective.

As Cuckoo Search Algorithm, other swarm intelligence algorithms are applied in clustering area. The Bees Algorithm is used in optimizing the document clustering problem because the algorithm is able to perform global and local search simultaneously. The experiments show that the Bees algorithm outperforms the genetic

algorithm and the K-means algorithm [11]. The authors also show the details of the ACO documents clustering method and detail results of experiments [12]. The experiments conclude that the ant algorithms can be successfully implemented in text documents processing.

3 Web Document Clustering

Web document clustering is widely applicable in areas such as search engines, web mining and information retrieval research areas. In most document clustering methods, several pre-processing steps such as stop words removal and stemming are performed on the given document set. Then, each document is represented by a vector of frequencies of remaining terms within each document. Some document clustering algorithms perform an extra pre-processing step such as term frequency normalization in which the actual term frequency is divided by the overall frequency of the term in the entire document set.

For clustering a document, the text content in a web document provides a lot of information. There are many document clustering approaches frequently used in document clustering field. They differ in many parts, depending on the types of attributes they use to describe the documents, the similarity measure used, the representation of the clusters etc. The different approaches can be categorized into (i) text based clustering approach, (ii) link based clustering approach and (iii) hybrid clustering approach. The text-based web document clustering approaches characterize each document according its content, i.e. by using the words (or sometimes phrases) contained in it. The basic idea is that if two documents contain many common words then it is likely that two documents are very similar.

3.1 Preprocessing of the Web Document

If the system is a text-based web document clustering system, the preprocessing is very similar to the preprocessing of the documents. Preprocessing consists of steps that take as input a plain text document and output a set of tokens (which can be single terms or n-grams) to be included in the vector model. Preprocessing consists of tokenizing, stop word removal and stemming stages.

3.1.1 Tokenizing

Tokenizing means breaking a stream of text up into words, phrases, symbols or other meaningful elements called tokens. Therefore, tokenizing forms words from a

sequence of characters. The list of tokens is input for text mining. An example of tokenization is:

Input Mozilla Firefox, Internet Explorer, and Google Chrome are famous web browsers.

Output Mozilla Firefox Internet Explorer and Google Chrome are famous web browsers

An instance of a sequence of characters in some particular document is defined as a token. They are grouped together as a useful semantic unit for processing. The class of all tokens containing the same character sequence is called a type. Then, a *term* is a type in the IR system's dictionary. The set of index terms could be entirely distinct from the tokens. They could be semantic identifiers in taxonomy. In modern IR systems, they are strongly related to the tokens in the document. However, they are usually derived from them by various normalization processes rather than being exactly the tokens that appear in the document [13].

3.1.2 Stop Words Removal

The extremely common words may appear in documents to be of little value in helping to select documents matching a user need. These words are defined as *stop words*. For example, all stop words are common words, such as a and the, are removed from the text collection. The common way to determine a stop list is to count the total number of times each term appears in the document collection. It is called stop list. The members of the stop list are discarded during indexing. By using the stop list, we can reduce the number of postings that a system has to store. The stop words removal can reduce the index space, improve the response time and improve effectiveness.

3.1.3 Stemming

Stemming refers to the mapping of word forms to stems or basic word forms. In other words, stemming is the use of linguistic analysis to get the root form of a word. Word forms can differ from stems due to morphological changes for grammatical reasons. Documents are going to contain different forms of a word, such as *play*, *played*, *and playing*. Besides, words such as *connect*, *connector*, and *connection* have similar meanings as they are families of derivationally related words. In many situations, it seems as if we search for one of these words, it should return documents that contain another word that has same meaning in the set. Both stemming and lemmatization are used to reduce inflectional forms. For instance:

Has, have, had	⇒	have
Bank, banks, bank's	⇒	bank
Save, saving, saves, saved	⇒	save

This mapping of text will make the result like:

My mother has opened a saving account at KBZ bank. => my mother have open a save account at KBZ bank.

However, the two words differ in their flavor. *Stemming* is a crude heuristic process chopping off the ends of words in the hope of achieving this goal correctly most of the time. It usually includes the removal of derivational affixes.

3.2 Vector Space Model Representation

Salton et al. [14] introduced, in the early seventies, a model for automatic indexing and the vector space model has become the standard document model for document clustering. In this model a document d can be defined as a set of terms $\{t_1, \ldots, t_n\}$. The importance or occurrence of each of these terms can be weighted by some metric.

The vector space model is the best well-known and most widely used IR model. In the vector space model, a document is represented as a weight vector. In a weight vector, each component weight is computed based on TF or TF-IDF scheme. The weight w_{ij} of term t_i in document d_j is no longer in {0, 1} as in the Boolean model. But it can also be any number.

For Term Frequency (TF), the weight of a term t_i in document d_j is the number of times that t_i appears in document d_j, defined by f_{ij}. Normalization can also be performed as shown in Eq. (1).

However, the TF does not consider the situation where a term can appear in many documents of the collection. This kind of term may not be discriminative.

So, TF-IDF is most widely used to weigh the term. In TFIDF, TF stands for the term frequency and IDF is the inverse document frequency.

Let N be the total number of documents in the collection and df_i be the number of documents in which term t_i appears at least once. Let f_{ij} be the raw frequency count of term t_i in document d_j. Then, the normalized term frequency (denoted by tf_{ij}) of t_i in d_j is given by

$$tf_{ij} = \frac{f_{ij}}{\max\{f_{1j}, f_{2j}, \ldots, f_{|v|j}\}} \tag{1}$$

where the maximum is computed over all terms that appear in document d_j. If term t_i does not appear in d_j then $tf_{ij} = 0$. $|V|$ is the vocabulary size of the collection.

The inverse document frequency, $idf_{i,}$ of term t_i is given by:

$$idf_i = \log \frac{N}{df_i}, \tag{2}$$

However, if a term appears in a large number of documents in the collection, it is probably not important or not discriminative. The final TF-IDF term weight is given by:

$$w_{ij} = tf_{ij} \times idf_i. \tag{3}$$

3.3 Calculating Distance Values

A similarity/distance measure between two data points must be determined before clustering. This measure defines the degree of closeness or separation of the target objects. So, it should correspond to the characteristics that are believed to distinguish the clusters embedded in the data. In many cases, there is no measure that is a universal best fit for all various kinds of clustering problems. These characteristics depend on the data or the problem context at hand.

Moreover, choosing an appropriate similarity measure is also important for cluster analysis, especially for a particular type of clustering algorithms. It is great important to understand the effectiveness of different measures in helping to choose the best one. In our algorithm, Cosine is chosen to measure the distance between the center values and the documents.

3.3.1 Cosine Similarity

Accurate clustering should have a precise definition of the closeness between a pair of objects, in terms of either their similarity or distance. A variety of similarity or distance measures have been proposed and widely applied, such as cosine similarity.

When documents are represented in terms of term vectors, the similarity of two documents can be defined as the correlation between the vectors. This is quantified as the cosine of the angle between vector and it is also called cosine similarity. Cosine similarity is one of the most popular similarity measure that has been applied to text documents, such as in many information retrieval applications [15] and clustering too [16].

Given two documents $\overrightarrow{t_a}$ and $\overrightarrow{t_b}$, their cosine similarity is

$$SIM_C\left(\underset{t_a}{\rightarrow}, \underset{t_b}{\rightarrow}\right) = \frac{\underset{t_a}{\rightarrow} \times \underset{t_b}{\rightarrow}}{\left|\underset{t_b}{\rightarrow}\right| \times \left|\underset{t_b}{\rightarrow}\right|} \tag{4}$$

where $\overrightarrow{t_a}$ and $\overrightarrow{t_b}$ are m-dimensional vectors over the term set T = {t_1, …, t_m}.

Each dimension represents a term with its weight in the document, which is non-negative. As a result, the cosine similarity is non-negative and bounded between [0, 1].

As in the K-means algorithm, the PSO-based Cuckoo Search Clustering Algorithm works with distance measures which basically aim to minimize the within-cluster distances. Therefore, similarity measures do not directly fit into the algorithm, because smaller values mean dissimilarity. We need to apply a simple transformation to convert the similarity measure to distance values. As cosine similarity is bounded in [0, 1] and monotonic, D = 1 − SIM is taken as the corresponding distance value [3].

3.4 Clustering

After preprocessing and calculating the distance values, the system goes to the clustering stage. In the clustering step, the clustering algorithms group a set of web documents into subsets or clusters. The clustering algorithms' goal is to create clusters that are similar internally, but clearly different from each other. In other words, documents within a cluster should be similar as much as possible and documents in one cluster should be different as much as possible from documents in other clusters.

The problem of clustering can be expressed as an optimization problem where one tries to search the optimal centers that can be expressed the centroids of the clusters that are quite from each other as much as possible. So, the optimization algorithms are applied in clustering areas as the clustering algorithms for searching optimal centroids.

4 Particle Swarm Optimization

Swarm Intelligence (SI) is an innovative distributed intelligent paradigm for solving optimization problems. Its inspiration is originally taken from the biological examples such as swarming, flocking and herding phenomena in vertebrates [17].

Particle Swarm Optimization is a population based stochastic optimization technique. It can be used to find an optimal, or near optimal, solution to a numerical and qualitative problem. In the PSO algorithm, birds in a flock are symbolically described as particles. These particles can be considered as simple agents that are "flying" through a problem space. A problem space in PSO can contain many dimensions to model the problem space. A particle's location in the multi-dimensional problem space can be defined as one solution for the problem. When a particle moves to a new location, it generates a different solution. This solution is then evaluated by a fitness function. The fitness value provides a quantitative value of the solution's utility.

At each generation of movement, the particle changes the velocity and direction when it moves along each dimension of the problem space. It is the particle's personal experience combined with its neighbors' experience in which each particle moves through a problem space. For every generation, the particle's new location is computed by adding the particle's current velocity V-vector to its location X-vector. The personal best position of ith particle is the best position visited by the particle and is denoted as *pbest*. Let f be the objective function and the personal best of particle at time step t is updated as

$$pbest_i(t+1) = f(x) = \begin{cases} pbest_i(t), & \text{if } f(x_i(t+1)) \geq f(pbest_i(t)), \\ x_i(t+1), & \text{if } f(x_i(t+1)) < f(pbest_i(t)). \end{cases} \quad (5)$$

Moreover, each particle knows the best *pbest* among all the particles and this is defined as *gbest.* The global best is updated based on best known position of the swarm using Eq. (6) [18].

$$gbest(t) \in \{pbest_0, pbest_1, \ldots, pbest_k\} = \min\{pbest_0(t), \ldots, pbest_k(t)\} \quad (6)$$

By considering *pbest, gbest* and the velocity of each particle the update rule for their position is as the following equations:

$$V_{t+1} = W_t * V_t + c1 * rand1 * (pbest - x_t) + c2 * rand2 * (gbest - x_t), \quad (7)$$

$$X_{t+1} = X_t + V_{t+1}. \quad (8)$$

- X_t is the particle current location.
- $c1$ and $c2$ are two positive acceleration constants.
- d is the number of dimensions of the problem space.
- *rand1*, *rand2* are random values in the range of (0, 1).
- w is called the constriction coefficient.

5 Cuckoo Search

Cuckoo Search is a meta-heuristic algorithm inspired by some species of a bird family called Cuckoo because they have their special lifestyle and aggressive reproduction strategy [18]. Cuckoos are fascinating birds, not only because they can make the beautiful sounds, but also because their reproduction strategy is aggressive. Cuckoo birds never build their own nests. These species lay their eggs in the nests of other host birds. They have the amazing abilities like selecting the recently spawned nests and removing existing eggs that increase hatching probability of their eggs. Cuckoo birds carefully mimic the color and pattern of the eggs of host

birds. So, the host takes care of the eggs presuming that the eggs are its own. However, when the host knows that the egg is not their own egg, they throw out the discovered alien eggs or build their new nests in new locations. The cuckoo breeding analogy is used to develop new design optimization algorithm. A set of host nests describes a generation. Each nest carries an egg (solution). The quality of the solutions is improved by generating a new solution from an existing solution and modifying certain characteristics.

1. Each cuckoo lays one egg at a time, and dumps it in randomly chosen nest.
2. The best nests with high quality of eggs will carry over to the next generations.
3. The number of available host nests is fixed, and the egg laid by a cuckoo is discovered by the host bird with a probability pa [0, 1]. In this case, the host bird can either throw the egg away or abandon the nest, and build a completely new nest.

The algorithm considers both local search and global search. This algorithm uses a balanced combination of a local random walk and the global explorative random walk. This combination is controlled by a switching parameter pa. The local random walk is as follows:

$$x_i^{t+1} = x_i^t + \alpha s \otimes H(p_a - \epsilon) \otimes (x_j^t - x_k^t) \tag{9}$$

- x_j^t and x_k^t are two different solutions selected randomly by random permutation.
- $H(u)$ is a Heaviside function.
- ϵ is a random number drawn from a uniform distribution and s is the step size.

On the other hand, the global random walk is carried out by using Lévy flights.

When generating new solutions $x^{(t+1)}$ for, say cuckoo i, a Lévy flight is performed using the following equation:

$$X_i^{(t+1)} = x_i^t + \alpha L(s, \lambda), \tag{10}$$

where

$$L(s, \lambda) = \frac{\lambda \Gamma(\lambda) \sin(\pi\lambda/2)}{\pi} \frac{1}{s^{1+\lambda}}, \ (s \gg s_0 > 0), \tag{11}$$

Here, α is the step size scaling factor and $\alpha > 0$. It should be related to the scales of the problem of interest. In most cases, α = O(L/10) is used where L is the characteristic scale of the problem of interest. In some cases, α = O(L/100) can be more effective and avoid flying too far. The above equation is essentially the stochastic equation for a random walk. In general, a random walk is a Markov chain in which next status/location is only dependent on the current location (the first term in the above equation) and the transition probability (the second term). However, a

substantial fraction of the new solutions should be generated by far field randomization and their locations should be far enough from the current best solution; this will be sure that the system will not be trapped in a local optimum [19, 20].

6 PSO-Based Cuckoo Search Clustering Algorithm

In this section, we explore the details of proposed algorithm. As mentioned in Sect. 5, the nature of cuckoo birds is that they do not raise their own eggs and never build their own nests, instead they lay their eggs in the nests of other host birds. So, the main strength of cuckoo search is to replace not so good solutions and to carry successful solutions to the next generation. As cuckoo eggs replace worse solutions, the next generation carries the good solutions and cuckoo solutions are important. Cuckoo birds need to look for a better place for their eggs.

In the PSO algorithm, the particles can be considered as simple agents "flying" through a problem space. The studies show that the PSO has more chance to "fly" into the better solution areas more quickly, so it can discover reasonable quality solution much faster than other evolutionary algorithms.

In the PSO based Cuckoo Search Clustering Algorithm, the cuckoo lays their eggs in a place determined by PSO solution. The goal is to replace not so good solution, the main idea of Cuckoo Search, with PSO solutions in which particles can fly into better solutions.

In the context of clustering, a single host nest represents a cluster centroid vectors. That is, each host nest X_i, is constructed as follows:

$$X_i = (m_{i1}, \ldots, m_{ij}, \ldots m_{ik}), \tag{12}$$

- m_{ij} refers to the j-th cluster centroid vector of the i-th host nest.
- k denotes the number of cluster centroids (as provided by the user), i.e. the number of clusters to be formed.

An initial solution of each host nest is randomly generated. Figure 3 is an example of the encoding of the single host nest in initial population for n dimensional space. Let $n = 2$, $k = 3$, the string of this particle represents three cluster centers [(0.7,2.7), (1.8,3.2) and (2.6,0.8)] (Fig. 3).

The algorithm is shown as follows. In our algorithm, it is assumed that each nest contains one egg.

Step 1 Initialize positions of vector X of all host nests randomly.
Step 2 For each host nest, assign the document to the nearest cluster by using Cosine Distance.
Step 3 Evaluate the fitness of each host nest by using Eq. (16).
Step 4 Update *pbest* and *gbest* using Eqs. (5) and (6).

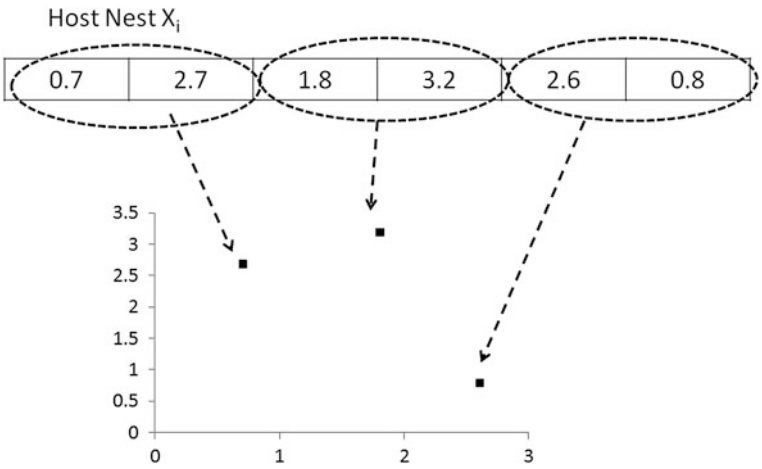


Fig. 3 Example of a host nest with 3 centriods

Step 5 Replace all worse nests by Cuckoo Eggs produced by using Eqs. (7) and (8).
Step 6 A fraction pa of worse nests are abandoned and new ones are built randomly.
Step 7 Keep the best solution (or nests with quality solutions).
Step 8 Rank the solution and find the current best.
Step 9 Repeat Step 2 to Step 8 until a stop criterion is satisfied or a predefined number of iterations are completed.
Step 10 Consider the clustering Solution represented by best solution.

The alien eggs discovery is preformed for each component of each solution in terms of probability matrix such as:

$$P_{ij} = \begin{cases} 1 & \text{if } rand < pa \\ 0 & \text{if } rand \geq pa \end{cases}, \tag{13}$$

where *rand* is a random number in [0, 1] interval and *pa* is the discovering probability. Existing eggs are replaced considering quality by the newly generated ones from their current positions through random walks with random step size.

$$stepsize = rand * (nest(randperm(n), :) - nest(randperm(n), :)) \tag{14}$$

$$new\_nest = nest + stepsize. * P, \tag{15}$$

where randperm is the random permutation function used for different rows permutation applied on nests matrix and P is the probability matrix. The system design is as shown in Fig. 4.

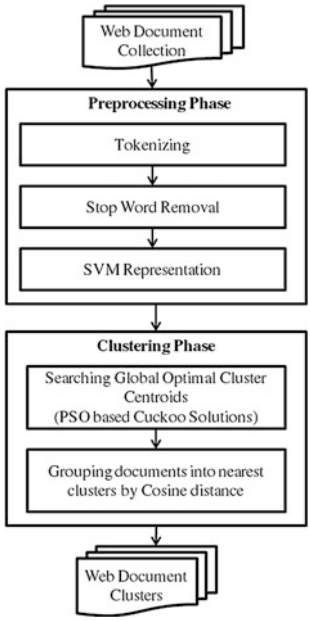


Fig. 4 System design

The documents to be clustered are collected first. The proposed method includes two phases: preprocessing phase and clustering phase.

In the preprocessing phase, each document will be tokenized and the stop words such as a, an, the etc., will be removed. The features from remaining words will be represented in Vector Space Model with their TFIDF weight values.

In the clustering phase, the distance from the centroids to the other documents will be measured by Cosine distance measure. The documents to the nearest center will go to this cluster. For next center selection, the old center is moved to the new center by PSO based Cuckoo Solutions. This means that the Cuckoo will find the better PSO position to lay their eggs. This clustering process will be performed for a defined number of criteria. The algorithm will finally produce the user-defined number of document clusters.

7 Experimental Evaluation

7.1 Evaluation Measures

The clustering algorithms aim to optimize some target function. But it is not clear whether this target function always divides the data into clusters according to the true nature of the data. The model cab not be fully representative or biased in some negative way. Sometimes, the documents express a greater depth than the algorithm can cover and the algorithm cannot cover the greater depth of the documents.

In the unsupervised clustering, evaluation methods evaluate the internal structure of the clustering. The density of the clusters can be calculated by calculating the *cohesion* of each cluster by using some distance function. Arguably a good clustering should provide dense clusters. Alternatively, the average *separation* between clusters can be measured. A good clustering should provide a good separation between internal and external objects.

So, we need to measure the cluster quality as a result of the clustering algorithm. In this chapter, the cluster quality is measured by fitness values and F-measure.

7.1.1 Fitness Measure

The fitness equation in Eq. (16) is used for the fitness value calculation and in the evaluation of the cluster quality. It indicates the value of the average distance between documents and the cluster centroid to which they belong. This is known as ADVDC value. It can be assumed that the smaller the ADVDC value, the more compact the clustering solution is.

The fitness value is measured by the equation below:

$$f = \frac{\sum_{i=1}^{N_c} \left\{ \frac{\sum_{j=1}^{p_i} d(o_i, m_{ij})}{p_i} \right\}}{N_c}, \tag{16}$$

- m_{ij} denotes the jth document vector, which belongs to cluster i.
- O_i is the centroid vector of ***i***th cluster.
- $d(o_i, m_{ij})$ is the distance between document m_{ij} and the cluster centroid O_i.
- P_i tends for the document number, which belongs to cluster C_i.
- N_c stands for the cluster number.

7.1.2 Precision, Recall and F-Measure

Some metrics such as *recall* and *precision* come from Information Retrieval. We can interpret a clustering as a retrieved set document given a query. Recall is defined as the proportion of relevant documents retrieved to all of relevant documents. Precision can be defined as the proportion of retrieved and relevant documents to all of the retrieved documents [21].

To calculate precision and recall, true positive, true negative, false positive and false negative need to be defined. We must first decide whether or not to assign the two documents to the same cluster given their calculated similarity. A true positive (TP) is the decision in which two similar documents are assigned to the same cluster. A true negative (TN) assignment is when two dissimilar documents are assigned to two different clusters. If two dissimilar documents are assigned to the

same cluster we have a false positive (FP) and two similar documents to different clusters we have a false negative (FN).

$$Precision(P) = TP/(TP + FP), \tag{17}$$

$$Recall(R) = TP/(TP + FN), \tag{18}$$

$$F - measure(F) = 2 * P * R/P + R. \tag{19}$$

7.2 Performance Evaluation

For the experimental set up of the proposed system, we use 7-sector benchmark dataset. The web documents are randomly drawn from the datasets and then clustered. The system is implemented with Java language. The platform's specifications used for the tests is: Intel(R) core(TM) i3 processor with 4 GB RAM.

7.2.1 Parameters and Their Specifications

The standard parameters in PSO-based Cuckoo Search Clustering Algorithm are pa, w_{start}, w_{end}, c1 and c2. We choose pa = 0.3 for Cuckoo Search. For the PSO movement of Cuckoo, the acceleration coefficient constants c1 and c2 are set to 1.4. Typically w(t) is reduced linearly, from w_{start} to w_{end}, each iteration, a good starting point is to set w_{start} to 0.9 and w_{end} to 0.4.

$$w(t) = \frac{(T_{max} - t) * (w_{start} - w_{end})}{T_{max}} + w_{end}. \tag{20}$$

The numbers of host nests are tested for 10, 15, 20 and 25. The parameter pa is tested for 0.15, 0.2, 0.25, 0.3 and 0.35. With pa = 0.3, the algorithm executes the best fitness value in early iterations. So, 0.3 is selected as the pa value of our algorithm.

From our simulations, we found that it is suitable for n = 15 and pa = 0.3. The tested pa values and its cluster quality are as shown in Table 1.

Table 1 Cluster qualities with different pa values

pa	0.15	0.2	0.25	0.3	0.35
Cluster quality	0.675 ± 0.647	0.641 ± 0.629	0.649 ± 0.636	0.631 ± 0.617	0.661 ± 0.638

7.2.2 Results and Discussion

The fitness values are recorded for 10 runs. If the fitness value is fixed then it shows the optimal solution. For each simulation, the centroids vector for clustering is randomly initialized.

The F-measure values are average of 100 simulations. Higher F-measures show the high accuracy. 150, 300, 450, 600 and 750 web documents are randomly selected from the dataset and are then clustered respectively. The results are shown in Fig. 5. It shows that the cluster accuracy increases according to the no of documents.

The fitness values of the algorithm over 10 runs are as shown in Fig. 6.

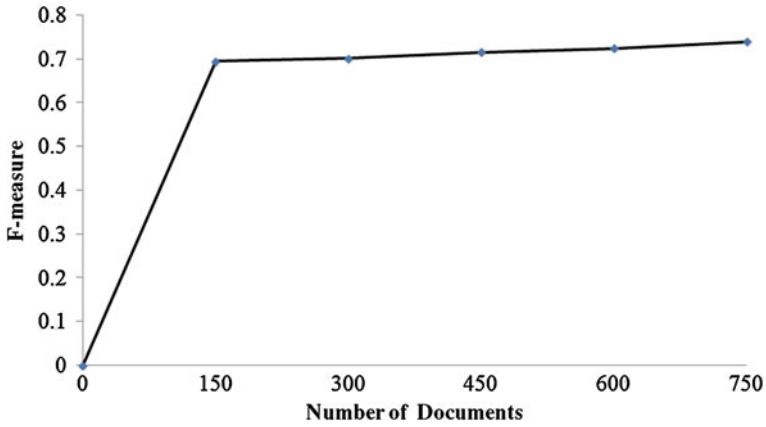


Fig. 5 Average F-measure

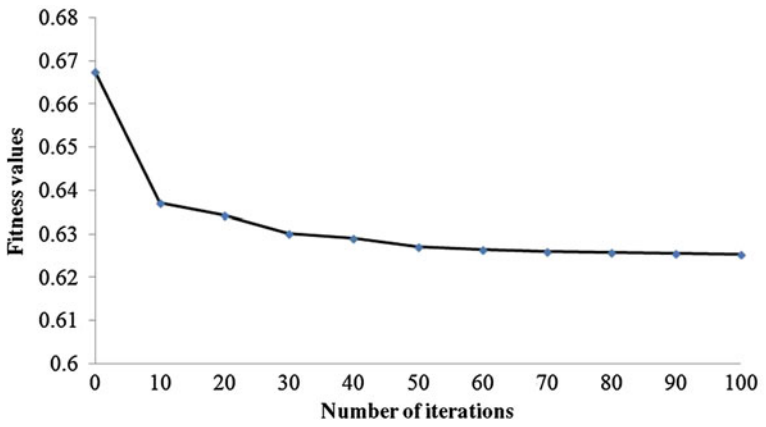


Fig. 6 Average fitness values

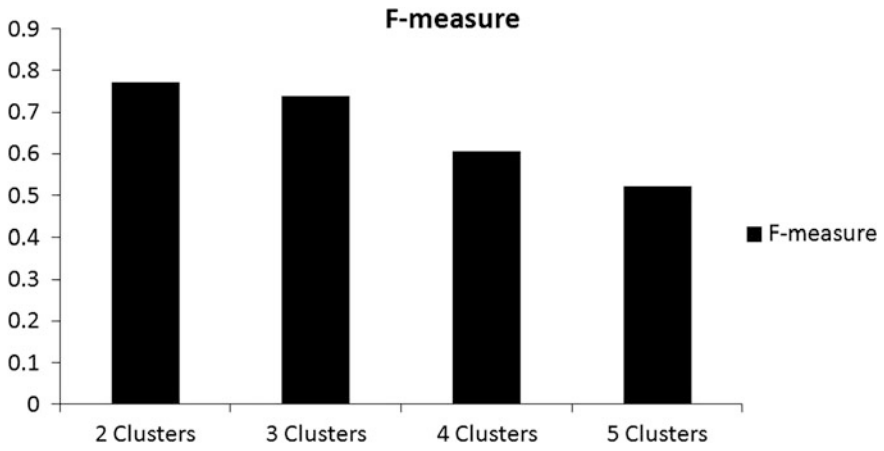


Fig. 7 F-measures over different number of clusters

In Fig. 6, we can know that PSO-based Cuckoo Search algorithm can find the optimal solution in early iterations. But, according to experiments, a large dataset will require more iteration to achieve optimal results.

The randomly selected web documents are then clustered into 2 clusters, 3 clusters, 4 clusters and 5 clusters respectively. The results are shown in Fig. 7. When 300 documents of 2 categories are clustered, it achieves the F-measure of 0.76. When the 750 documents are clustered into 5 clusters, it gains the F-measure of 0.53. The F-measure for 2 clusters is relatively high but, it decreases when the number of clusters increases.

8 Conclusion and Future Work

The PSO-based Cuckoo Search Clustering Algorithm is proposed and applied in web document clustering area. From the experimental results, we can see that the cluster quality and f-measure values obtained are good. So, we can conclude that the PSO-based Cuckoo Search Clustering Algorithm can perform well in web document clustering area. For future work, it can be suggested that the clustering accuracy for more number of clusters can be improved by means of semantic clustering with the help of word net, ontology or Wikipedia. Furthermore, the PSO based Cuckoo Search Clustering algorithm can be applied in other clustering areas. And it can also be compared to other swarm intelligence clustering algorithms.

References

1. Oikonomakou, N., Vazirgiannis, M.: A review of web document clustering approaches
2. Schenker, A., Last, M., Bunke, H., Kandel, A.: Clustering Of Web Documents Using a Graph Model (2003)
3. Huang, A.: Similarity Measures for Text Document Clustering, NZCSRSC 2008. Christchurch, New Zealand (2008)
4. Sridevi, K., Umarani, R., Selvi, V.: An analysis of web document clustering algorithms. Int. J. Sci. Technol. India (2011)
5. van der Merwe, D.W., Engelhrecht, A.P.: Data Clustering Using Particle Swarm Optimization, Evolutionary Computation 2003 Congress, IEEE, New York, Dec 2003
6. Ye, F., Chen, C.Y.: Alternative KPSO-clustering algorithm. Tamkang J. Sci. Eng. **8**(2), 165–174 (2005)
7. Cui, X., Potok, T.E.: Document clustering using particle swarm optimization. In: Proceedings of Swarm Intelligence Symposium (2005)
8. Zamir, O., Etzioni, O.: Web document clustering: a feasibility demonstration. In: Proceedings of 21st Annals Int'l ACM SIGIR Conference, pp. 46–54 (1998)
9. Goel, S., Sharma, A., Bedi, P.: Cuckoo Search Clustering Algorithm: A Novel Strategy of Biomimicry, World Congress on Information and Communication Technologies. IEEE publication, New York (2011)
10. Zaw, M.M., Mon, E.E.: Web document clustering using cuckoo search clustering algorithm based on levy flight. Int. J. Innov. Appl. Stud. **4**(1), 182–188 (2013)
11. AbdelHamid, N.M., Abdel Halim, M.B., Waleed Fakhr, M.: Bees algorithm-based document clustering. In: The 6th International Conference on Information Technology (2013)
12. Machnik, L.: Documents clustering method based on ants algorithms. In: Proceedings of the International Multi Conference on Computer Science and Information Technology, pp. 123–130 (2006)
13. Christopher, D.M., Prabhakar, R., Hinrich, S.: An Introductio to Information Retrieval, 1st edn. Cambridge University Press, Cambridge (2008)
14. Salton, G., Wong, A., Yang, C.S.: A vector space model for automatic indexing. Comm. ACM **18**(11), 613–620 (1975)
15. Yates, R.B., Neto, B.R.: Modern Information Retrieval. Addison-Wesley, New York (1999)
16. Larsen, B., Aone, C.: Fast and effective text mining using linear-time document clustering. In: Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (1999)
17. Nedjah, N., Mourelle, L.M.: Swarm Intelligent Systems. Springer, New York (2006)
18. Settles, M.: An introduction to particle swarm optimization, 7 Nov 2005
19. Yang, X.S., Deb, S.: Cuckoo Search via Lévy Flights. In: Proceedings of World Congress on Nature and Biologically Inspired Algorithms, pp. 210–214. IEEE publication, New York (2009)
20. Yang, X.S., Deb, S.: Engineering optimization by cuckoo search. Int. J. Math. Model. Num. Opt. **1**(4), 330–343 (2010)
21. Andrews O.N., Edward, A.F.: Recent Developments in Document Custering, Technical Report, Computer Science, Virginia Tech (2007)

Geometrically Nonlinear Analysis of Trusses Using Particle Swarm Optimization

Rasim Temür, Yusuf Sait Türkan and Yusuf Cengiz Toklu

Abstract Particle swarm optimization (PSO) algorithm is a heuristic optimization technique based on colony intelligence, developed through inspiration from social behaviors of bird flocks and fish schools. It is widely used in problems in which the optimal value of an objective function is searched. Geometrically nonlinear analysis of trusses is a problem of this kind. The deflected shape of the truss where potential energy value is minimal is known to correspond to the stable equilibrium position of the system analyzed. The objective of this study is to explore the success of PSO using this minimum total potential energy principle, in finding good solutions to geometrically nonlinear truss problems. For this purpose analyses are conducted on three structures, two plane trusses and a space truss. The results obtained show that in case of using 20 or more particles, PSO produces very good and robust solutions.

Keywords Particle swarm optimization · Structural analysis · Truss systems · Nonlinear analysis · Potential energy · TPO/MA

R. Temür
Department of Civil Engineering, Faculty of Engineering, Istanbul University,
34320 Istanbul, Turkey
e-mail: temur@istanbul.edu.tr

Y.S. Türkan
Open and Distance Education Faculty, Istanbul University,
34452 Istanbul, Turkey
e-mail: ysturkan@istanbul.edu.tr

Y.C. Toklu (✉)
Department of Civil Engineering, Faculty of Engineering,
Bilecik Seyh Edebali University, 11210 Bilecik, Turkey
e-mail: cengiz.toklu@bilecik.edu.tr

X.-S. Yang (ed.), *Recent Advances in Swarm Intelligence and Evolutionary Computation*,
Studies in Computational Intelligence 585, DOI 10.1007/978-3-319-13826-8_15

1 Introduction

Metaheuristic algorithms are a class of computational methods for solving optimization problems, inspired by nature, biological and social processes, music, physics, etc. These algorithms aim to find optimum value of objective functions which is defined uniquely for a particular problem by improving randomly formed solution sets using random numbers and certain rules. Some examples of these methods are genetic algorithms, simulated annealing, harmony search, ant colony optimization algorithm and particle swarm optimization algorithm.

Particle swarm optimization algorithm (PSO) was inspired by behaviors of bird flocking and fish schooling, and it is a heuristic optimization technique based on swarm intelligence successfully applied to various nonlinear NP-hard problems [1]. One of these problems is the geometrically nonlinear analysis of trusses.

Structural systems are in stable equilibrium phase when total potential energy is relatively minimum [2]. Based on this consideration, equilibrium configurations of structures can be determined by search algorithms looking for the minimization of the total potential energy of the system [3]. If the search algorithms are meta-heuristic, then the method applied becomes Total Potential Optimization using Metaheuristic Algorithms (TPO/MA). In TPO/MA method, total potential energy equation is written down based on deformation parameters of structural system, and deformation parameters minimizing total potential energy value are obtained by using heuristic algorithms. In trusses whose load case, geometry, cross-section and material properties are known, deformation parameters become joint coordinates. Since strain energy in the members can be expressed easily for finite deformations and for nonlinear materials, this method becomes applicable to nonlinear cases too [3].

2 Problem Formulation

In analysis of structural systems, total potential energy of a structure can be computed by Eq. (1).

$$U(\varepsilon) = \int_V e(\varepsilon) dV - \sum_{i=1}^{N_P} P_i \cdot u_i \tag{1}$$

In this equation, ε corresponds to strain, P_i corresponds to external loads, u_i corresponds to joint displacements relating to external loads, N_P is the number of loads acting on the system, V is volume of the structural element and $e(\varepsilon)$ corresponds to strain energy.

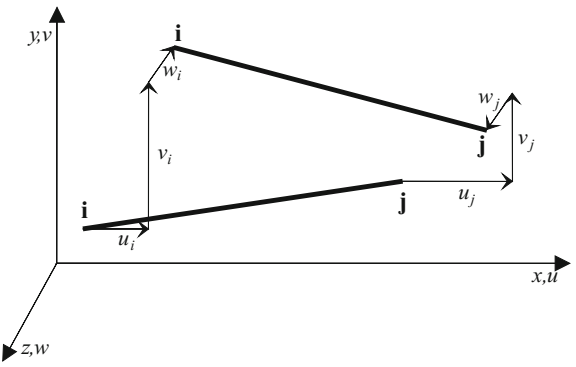


Fig. 1 Deformation of a space truss structural element

$$e(\varepsilon) = \int_0^{\varepsilon} \sigma(\varepsilon) d\varepsilon \tag{2}$$

$e(\varepsilon)$ value can be computed by using Eq. (2). The relation between stress (σ) and strain (ε) can be nonlinear as well as linear.

Initial coordinates and length of ij truss structural element (Fig. 1) are defined to be (x_i, y_i, z_i) and (x_j, y_j, z_j) and $L(0)$ (Eq. (3)), respectively. Final coordinates and length of element after the load are (u_i, v_i, w_i) and (u_j, v_j, w_j) and $L(c)$ (Eq. (4)), respectively.

$$L(0) = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2 + (z_j - z_i)^2} \tag{3}$$

$$L(c) = \sqrt{(x_j - x_i + u_j - u_i)^2 + (y_j - y_i + v_j - v_i)^2 + (z_j - z_i + w_j - w_i)^2} \tag{4}$$

Length change of member is $\Delta L(c) = L(c) - L(0)$. Strain value of a member can be computed by using Eq. (5).

$$\varepsilon(c) = \Delta L(c)/L(0) \tag{5}$$

For linear elastic cases, the stress–strain relation simply becomes $\sigma = E \cdot \varepsilon$ where E is the modulus of elasticity of the material. In this case Eq. (2) degenerates to $e = E \cdot \varepsilon(c)^2/2$.

The energy formed by internal forces of elements in structural system is obtained by multiplying strain energy and member volume. The energy formed by external forces can be computed by multiplying the point where force is applied and displacement along direction of force. Noting that A_j is the cross-section area of the member j and L_j is the length of the member j, the total potential energy of a structural system, can be obtained by Eq. (6) [3].

$$U(\varepsilon) = \sum_{j=1}^{N_m} e_j \cdot A_j \cdot L_j - \sum_{i=1}^{N_P} P_i \cdot u_i \tag{6}$$

3 Particle Swarm Optimization

Particle Swarm Optimization (PSO) developed by Eberhart and Kennedy [1], is based on a simulation of social behaviors of bird and fish swarms. PSO is inspired by the ability of swarm cooperation to work as a whole in finding their food source in a large area. The search for food and sanctuary whose places are not known by swarm is resembled to the search for mathematical solutions to engineering problems. Individuals of the swarm are called as "particles". Such significant social behavior of the particles within the swarm can be used in PSO algorithm. Accordingly, particles;

- are prone to go to best position among the positions where they have been before,
- follow the particle which is nearest to the target in swarm,
- have variable velocities for field scanning.

These behaviors form the basis of PSO algorithm. In particle swarm optimization, optimum value is searched with experiences of the individuals constituting the colony and sharing in between these individuals. In search process; every individual takes into account the experience of other individuals which constitute the swarm as well as makes use of its previous experiences.

In PSO algorithm every particle has to remember its coordinates, velocity, and best position it has obtained so far. In what way its velocity and direction will change each time while searching the target is related to its current position, best positions of its neighbors, and best position of itself. PSO algorithm is initialized with a group of random particles and each particle represents a possible solution. The best position related with the best fitness value of the particle is called personal best or local best. The best position among all particles in the swarm is stated as global best.

PSO process starts by assigning random positions to the particles (Fig. 2). Velocity of each particle is calculated using velocity equation; and their positions in the next step are obtained by using their initial positions and velocities. Velocity and positions of the particles are updated at each iteration. Velocity vector of the particle is calculated considering best position of the particle in the past (X_{lbest}) and the position of the particle which is nearest to the target (the best solution) in the swarm, as given in Eq. (7). New position of the particle i (X_i^{t+1}) is obtained by summation of current position of the particle X_i^t and particle velocity V_i^{t+1} (Eq. 8) [5, 6].

Fig. 2 Pseudo Code of PSO algorithm [4]

```
Initialize location and velocity of each particle
repeat
  for each particle
    evaluate objective function / at the particles location
  endfor
  for each particle
    update the personal best position
  endfor
  update the global best position
  for each particle
  update the velocity
  compute the new location of the particle
  endfor
until stopping criterion is met
```

$$V_i^{t+1} = V_i^t + c_1 \cdot r_1^t \cdot \left(X_{lbest} - X_i^t\right) + c_2 \cdot r_2^t \cdot \left(G_{gbest} - X_i^t\right) \tag{7}$$

$$X_i^{t+1} = X_i^t + V_i^{t+1} \tag{8}$$

In these equations, parameters are denoted as follows:

V_i^{t+1} Velocity of the particle;
V_i Previous velocity of the particle;
X_i Current position of the particle;
t Number of iterations;
r_1, r_2 Random numbers regularly distributed in the interval of {0;1}
c_1, c_2 Constants of acceleration;
X_{lbest} Best position of the particle in the past, local best;
X_{gbest} Position of the particle that is nearest to target in swarm, global best

In PSO algorithm each particle updates its position considering individual (personal-local) best position "X_{lbest}" and social global best "X_{gbest}" position. If a particle's current position is better than its previous best position then local best value is changed. The best solution of the objective function (fitness value) is also recorded and updated in each iteration. This loop continues until a predetermined condition is met. Exhibition of a sample iteration in the particle swarm optimization was presented in Fig. 3. The particle reaches to X_i^{t+1} position from X_i^t position after this iteration.

Shi and Eberhart [7] obtained the formula given below by adding inertia weight "w" to the formula which is used in calculation of velocity vector (Eq. 9).

$$V_i^{t+1} = w.V_i^t + c_1 \cdot r_1^t \cdot \left(X_{lbest} - X_i^t\right) + c_2 \cdot r_2^t \cdot \left(X_{gbest} - X_i^t\right) \tag{9}$$

Parameter "w" controls the influence of the previous velocity. It was proposed that inertia weight should take a value in the interval of {0.9; 1.2} for the algorithms to give better results [7]. Other parameters in the equation are extremely important

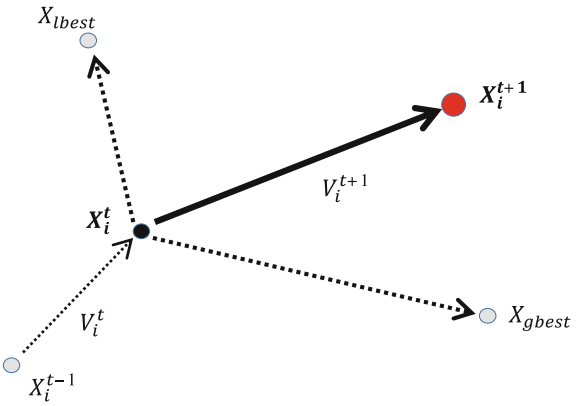


Fig. 3 Exhibition of particle swarm optimization

for the equilibrium in between exploration and exploitation. Parameter "c_1" which is one of the acceleration constants contributes to obtain a faster solution by focusing on local best position (solution) of the particle. Other constant that is "c_2" provides obtaining a better position in a larger field by evaluating best positions of all particles in the swarm.

In the search within solution space; "r_1, r_2 " which are random values prevent to be caught by the potential loops around local best values. Therefore; probability of finding global best or an approximate value to the global best is increased.

In some cases, velocity vector may take undesired values and converge to infinity by growing up excessively. To be able to prevent this situation; specifying minimum and maximum limits for velocity vectors is a method which is frequently applied in the searches.

$$\text{If } V_i^{t+1} > V_{up}, \quad V_i^{t+1} = V_{up} \tag{10}$$

$$\text{If } V_i^{t+1} < V_{low} \quad V_i^{t+1} = V_{low} \tag{11}$$

The values that velocity vector may take are shown in Eqs. (10)–(11) in case of the velocity vector exceeds maximum and minimum limits. The values that velocity vector may take in case of it exceeds the defined values are denoted as "V_{low}" and "V_{up}".

4 Implementation of PSO Algorithm

To carry out geometrically nonlinear analysis of truss structural systems in this study, joint displacements were defined to be unknowns in PSO algorithm. Based on this procedure, random displacements were defined for every freedom in structural system, and rules of PSO algorithm and the velocity equation in Eq. (9) were used. Lower and upper limits were defined for particle velocities. The lowest

velocity for particles was limited to V_{low} = 0.003 and the highest velocity for particles was limited to V_{up} = 0.5. When a velocity value that is out of limits in the velocity equation was obtained, that value was shifted to the closest velocity limit. Similarly, $w_{low} = 0.001$ and $w_{up} = 0.7$ values were used for inertia weight. Inertia weight decreases linearly through iterations from the upper value to the lower value. By this way, velocity is reduced in each iteration, which contributes to more sensitive obtainment of joint displacements. Velocity coefficients were determined to be $c_1 = c_2 = 2$. Total potential energy equation given in Eq. (6) was defined to be the objective function in PSO. The solution set, where total potential energy value of structural system is observed to be minimum in the end of iterations, is accepted to be the final solution.

In order to implement PSO algorithm into problem of analysis of truss structural systems, a software application written in Visual Basic 2012 was developed.

5 Numerical Examples

In this study, analyses of three different truss structural systems in literature were performed by using PSO algorithm. Five different particle numbers such as 5, 10, 20, 30 and 40 were used in analyses. Results of analyses were given in comparison with results of previous studies.

5.1 Example 1: 2 Bar Plane Truss

In 2-bar truss structure which is depicted in Fig. 4 [8–10], all members characterized by the cross-sectional area A = 9,677 mm^2 and modulus of elasticity E = 68,941 N/mm^2.

Load-displacement diagram obtained by 10 different P load is given in Fig. 5. The system behavior under loads applied on the system is geometrically nonlinear according to results of analysis by FEM (Fig. 5). Results of analysis carried out by PSO algorithm suggest that the system has geometrically nonlinear behavior. PSO analysis performed with different number of particles gave the same results. Examining energy values presented in Table 1, energy values found by PSO match up with FEM nonlinear results.

5.2 Example 2: 6 Bar Plane Truss

Within the scope of this study, the examined plane truss structure sample is the system with 6 bar elements solved first by Toklu [3]. The geometry of the system is

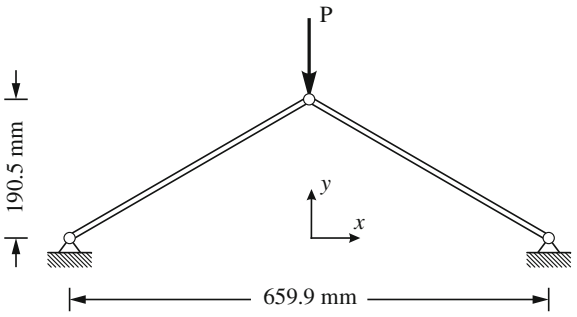


Fig. 4 Bar plane truss system [8, 10]

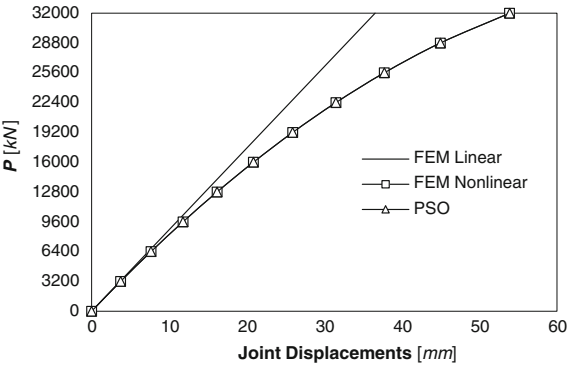


Fig. 5 Load-displacement diagram of two-bar plane truss structural system

Table 1 Energy values of 2-bar plane truss structural system

P [kN]	Total potential energy [kNm]		
	PSO	FEM nonlinear	FEM linear
3,200	−5.93	−5.93	−5.93
6,400	−24.11	−24.11	−24.07
9,600	−55.17	−55.17	−54.93
12,800	−99.83	−99.83	−98.95
16,000	−159.00	−159.00	−156.81
19,200	−233.77	−233.77	−228.73
22,400	−325.52	−325.52	−315.66
25,600	−436.10	−436.10	−420.58
28,800	−568.12	−568.12	−535.55
32,000	−725.72	−725.72	−669.33

presented in Fig. 6. All members have modulus of elasticity of 200,000 N/mm^2. Cross-sectional area of elements with numbers 1, 5 and 6 is 200 mm^2; cross-sectional area of elements with numbers 2, 3 and 4 is 100 mm^2. The magnitude of *P* force affecting on joint number 4 is 150 kN.

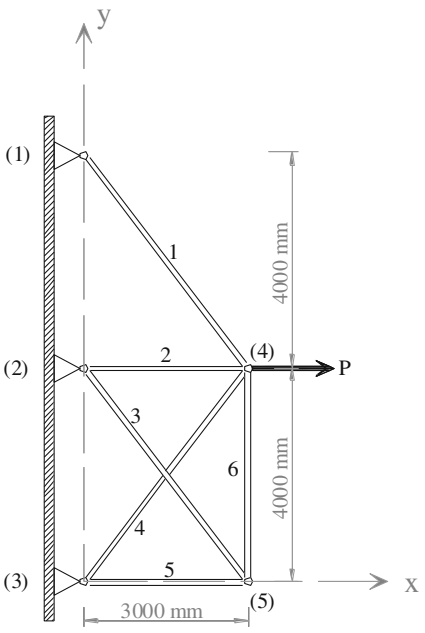


Fig. 6 6-bar plane truss system [3]

The system, which has four degrees of freedom, was analyzed by local search (LS) algorithm, genetic algorithm (GA), ant colony algorithm (ACO), harmony search (HS) and finite elements method (FEM). In the first of these studies, that gave way to the method Total Potential Optimization using Metaheuristic Algorithms (TPO/MA), LS was applied for determination of the deflected shape of the system, taking into account various material properties and the effect of failed members [3]. In the second study, GA was applied to minimize the energy of the system using a binary coded system with 10 digits, and considering the probabilities of cross-over and mutation as 0.7 and 0.1, respectively [11]. In ACO application for solving this truss, 1 member force is considered as the principal unknown of the problem, which is then determined so as to make the total energy a minimum [12]. The HS solution of the problem is performed by following the rules of TPO/MA, using 5, 10, and 20 vectors, with classical HS parameters $HMCR$ = 0.9 and par = 0.4 [13]. In the same study FEM solutions are also given obtained by linear and nonlinear applications.

Displacements and total potential energy values achieved by PSO analysis were compared with LS, GA, ACO, HS and FEM results (Table 2).

Examining total potential energy amounts presented in Table 2, it is seen that results obtained by PSO algorithm are matching with results obtained by LS, GA, HS, ACO and FEM Nonlinear. The reason why energy values obtained by FEM linear are greater compared to other analysis is that the system has geometrically nonlinear behavior under current loads.

Table 2 Analysis results of 6-bar plane truss structure system by different methods

		FEM		LS [3]	GA [11]	ACO [12]	HS [13]	PSO
		Linear	Nonlinear					
Joint displacements [mm]	u4	14.150	14.120	14.12	–	14.12	14.118	14.12
	v4	2.843	2.828	2.83	–	2.83	2.830	2.83
	u5	0.300	0.302	0.30	–	0.30	0.304	0.30
	v5	2.309	2.317	2.32	–	2.32	2.319	2.32
Member forces [N]	1	49,725.13	49,810.25	49,811	51,015.0	49,811	49,789.18	49,811
	2	94,331.33	94,140.09	94,142	94,140.6	94,142	94,131.58	94,142
	3	−6,669.14	−6,688.40	−6,688	−6,552.8	−6,688	−6,687.50	−6,688
	4	43,055.99	42,973.59	42,974	42,029.5	42,974	42,977.97	42,974
	5	4,001.49	4,034.16	4,035	3,963.8	4,035	4,074.45	4,035
	6	5,335.31	5,351.45	5,352	5,284.3	5,352	5,354.77	5,352
Total potential energy [kNm]		−1.059727	−1.059735	−1.059735	−1.0560	−1.059735	−1.059734	−1.059735

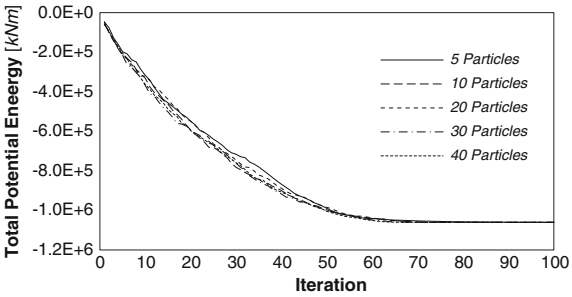


Fig. 7 Change of total potential energy amounts in analysis carried out in the 6-bar truss with different particle numbers

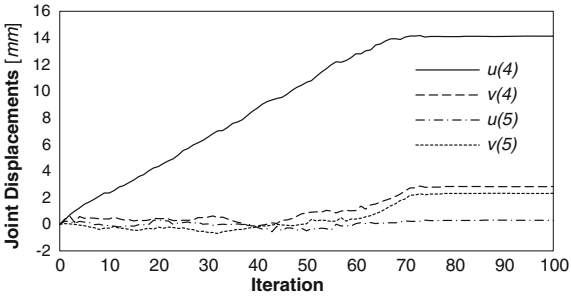


Fig. 8 Change of displacements in the analysis of 6-bar plane truss structure system with 20 particles

Analysis with different particle numbers resulted in equal energy values. The same energy values were obtained as a result of using particle numbers ranging between 5 and 40. On the other hand, there are not considerable differences among total potential energy values which change depending on number of iterations (Fig. 7). Change of displacement values in PSO analysis depending on iteration number is given in Fig. 8.

5.3 Example 3: 25 Bar Space Truss

The truss structure system used in PSO algorithm analysis includes 25 bar elements, 10 joints and 4 fixed bearings [14]. In the system whose geometry is presented in Fig. 9, modulus of elasticity of members is 200,000 N/mm^2 and cross-sectional area of members is 10 mm^2. The system was analyzed considering 3 different load cases.

Loading 1:
at joint 1, $P_y = 80$ kN and $P_z = -20$ kN
at joint 2, $P_y = -80$ kN and $P_z = -20$ kN

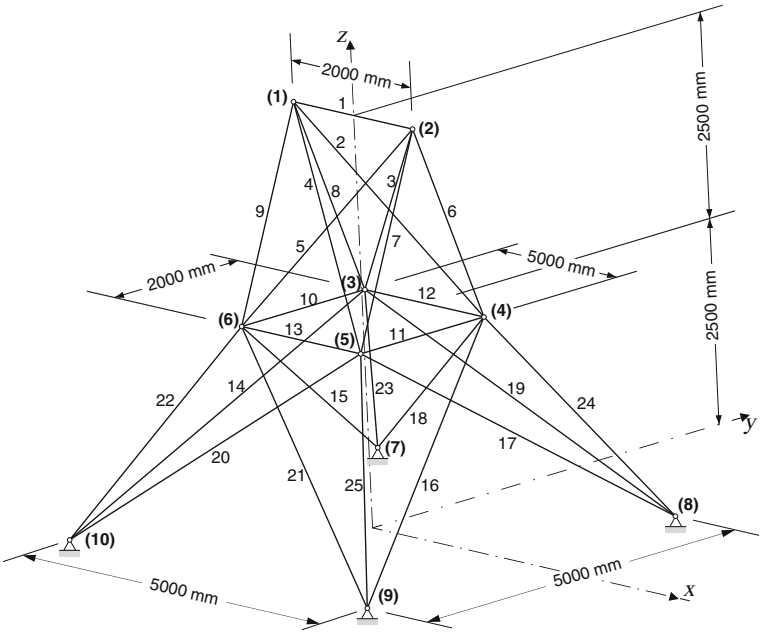


Fig. 9 25-bar space truss system

Loading 2:
at joint 1, P_y = 800 kN and P_z = −200 kN
at joint 2, P_y = −800 kN and P_z = −200 kN
Loading 3:
at joint 1, P_y = 800 kN, P_z = 800 kN and P_z = −200 kN

Displacement and member force values obtained as a result of analysis by HS, FEM Nonlinear [13] and PSO methods by using these loads were presented in Tables 3 and 4. There are differences in values of displacement and member force but total potential energy values are equal. Percentages of differences in analysis results are limited to 0.44 % for displacements and 0.13 % for member forces (Table 5).

In order to determine effects of particle quantities on results of analyses, change of total potential energy amount depending on iteration number was examined for three loading cases. According to results of Loading 1 in Fig. 10, same energy values were obtained for all number of particles. As number of particles increase, number of iterations to find final solution decreases.

According to results of Loading 2 in Fig. 11, equal potential energy values were obtained by analysis carried out with 10, 20, 30 and 40 particles. Higher energy level was observed in the analysis with 5 particles.

Table 3 The displacements of 25-bar space truss system by different methods

		Loading 1			Loading 2			Loading 3		
		HS [13]	FEM nonlinear	PSO	HS [13]	FEM nonlinear	PSO	HS [13]	FEM nonlinear	PSO
Joint displacements [mm]	u(1)	−0.076	0.000	0.06	1,257.533	1,261.101	1,260.63	2,522.412	2,522.974	2,521.86
	v(1)	37.910	37.847	38.01	−527.509	−528.823	−529.80	1,840.283	1,840.856	1,839.88
	w(1)	−37.214	−37.199	−37.27	−455.548	−456.667	−455.35	−3,078.900	−3,083.146	−3,076.40
	u(2)	−0.064	0.000	0.06	−1,264.620	−1,261.101	−1,261.15	2,437.300	2,441.255	2,435.57
	v(2)	−37.766	−37.847	−37.68	529.739	528.823	529.20	2,522.971	2,523.148	2,522.31
	w(2)	−37.153	−37.199	−37.16	−457.905	−456.667	−455.38	−1,435.857	−1,441.611	−1,433.56
	u(3)	0.892	0.867	0.86	−48.858	−48.610	−47.50	−393.940	−394.610	−393.83
	v(3)	−1.731	−1.744	−1.73	−288.344	−288.313	−287.03	−371.211	−371.344	−371.22
	w(3)	−16.342	−16.392	−16.45	−362.120	−362.743	−361.33	−904.675	−905.218	−903.69
	u(4)	0.924	0.867	0.88	−203.107	−202.750	−203.19	579.074	579.039	579.02
	v(4)	1.750	1.744	1.76	−296.619	−296.792	−295.57	198.927	199.088	198.35
	w(4)	−16.355	−16.392	−16.38	−67.535	−68.466	−68.84	44.972	44.423	45.45
	u(5)	−0.822	−0.867	−0.85	48.774	48.610	47.10	1,013.812	1,015.303	1,012.54
	v(5)	1.751	1.744	1.78	288.714	288.313	286.57	1,037.329	1,038.580	1,036.29
	w(5)	−16.402	−16.392	−16.35	−363.583	−362.743	−361.26	−353.069	−356.179	−351.08
	u(6)	−0.844	−0.867	−0.87	202.232	202.750	202.95	407.347	408.348	406.51
	v(6)	−1.746	−1.744	−1.70	297.259	296.792	295.18	802.637	803.160	802.21
	w(6)	−16.430	−16.392	−16.40	−69.186	−68.466	−68.67	−36.622	−36.688	−36.68
Energy [kNm]		−3.7645	−3.7645	−3.7645	−1,444.6	−1,444.6	−1444.6	−2,860.5	−2,860.5	−2,860.5

Table 4 The member forces of 25-bar space truss system by different methods

		Loading 1			Loading 2			Loading 3		
		HS [13]	FEM nonlinear	PSO	HS [13]	FEM nonlinear	PSO	HS [13]	FEM nonlinear	PSO
Member forces [kN]	1	75.676	75.693	75.690	692.590	692.496	691.627	107.640	106.578	107.691
	2	3.915	3.893	3.902	−334.408	−334.607	−334.677	273.603	274.288	273.432
	3	3.883	3.893	3.869	73.046	72.764	73.231	−310.372	−310.686	−310.660
	4	3.890	3.893	3.898	72.343	72.764	73.245	247.023	246.394	247.501
	5	3.885	3.893	3.881	−334.711	−334.607	−334.761	−70.532	−70.015	−70.574
	6	−13.844	−13.883	−13.879	275.294	274.675	275.049	−10.172	−10.677	−10.110
	7	−13.875	−13.883	−13.874	−189.137	−189.233	−189.519	359.301	359.449	359.219
	8	−13.899	−13.883	−13.909	−189.279	−189.233	−189.291	187.014	187.700	186.599
	9	−13.894	−13.883	−13.907	273.927	274.676	275.003	471.844	472.059	471.760
	10	1.736	1.734	1.730	−132.535	−132.732	−132.998	−110.993	−111.426	−111.052
	11	1.745	1.734	1.730	−132.868	−132.732	−132.948	−180.275	−180.523	−179.696
	12	−3.482	−3.489	−3.490	35.618	35.767	35.618	−26.908	−26.811	−26.821
	13	−3.497	−3.489	−3.480	35.798	35.767	35.781	−106.789	−106.798	−106.819
	14	−3.376	−3.395	−3.413	−52.294	−52.346	−51.883	−260.641	−260.940	−260.427
	15	−3.412	−3.395	−3.402	−125.339	−125.277	−125.189	−237.250	−237.647	−236.904
	16	−3.366	−3.395	−3.385	−125.143	−125.277	−125.368	238.850	238.741	238.851
	17	−3.412	−3.395	−3.385	−52.509	−52.346	−51.825	−177.597	−178.359	−177.046
	18	−4.657	−4.656	−4.660	116.226	116.025	115.587	−125.783	−125.978	−125.457
	19	−4.643	−4.656	−4.664	−174.651	−174.822	−174.273	−248.091	−248.093	−247.944
	20	−4.654	−4.656	−4.656	−175.121	−174.822	−174.142	−190.078	−190.717	−189.732
	21	−4.662	−4.656	−4.643	115.946	116.025	115.450	332.335	332.665	332.074
	22	−7.378	−7.367	−7.385	−46.776	−46.168	−45.834	−52.549	−52.285	−52.744
	23	−7.355	−7.367	−7.397	−54.925	−55.264	−55.429	−106.707	−106.579	−106.436
	24	−7.365	−7.367	−7.361	−45.581	−46.168	−45.918	−50.752	−50.960	−50.697
	25	−7.358	−7.367	−7.333	−55.421	−55.264	−55.700	542.045	542.082	541.876

Table 5 Comparison of PSO and FEM Nonlinear analysis results of 25-bar space truss structural system

	Parameter	Diff. (%)	FEM nonlinear	PSO
Loading 1	v_2	0.44	37.847 mm	37.680 mm
	f_1	0.004	75.693 kN	75.690 kN
Loading 2	u_1	0.04	1,261.101 mm	1,260.630 mm
	f_1	0.13	692.496 kN	691.627 kN
Loading 3	w_1	0.22	3,083.146 mm	3,076.40 mm
	f_{25}	0.04	542.082 kN	541.876 kN

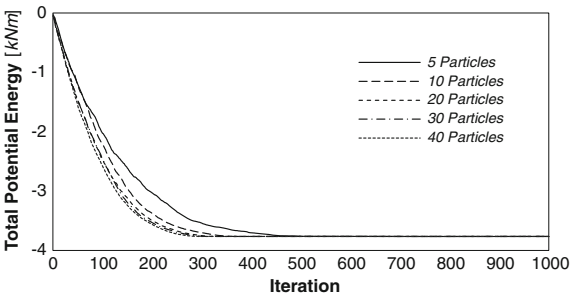


Fig. 10 Change of total potential energy amounts in analyses carried out in the 25-bar space truss with different particle numbers for Loading 1

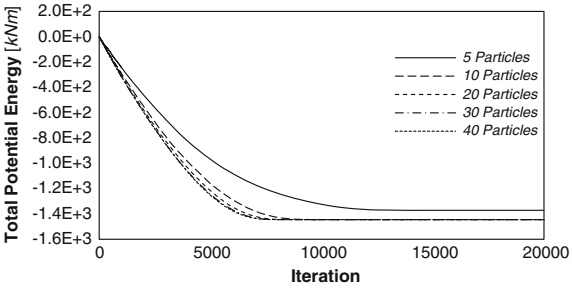


Fig. 11 Change of total potential energy amounts in analyses carried out in the 25-bar space truss with different particle numbers for Loading 2

In Fig. 12, where results of analyses regarding to Loading 3 were presented, shows that energy values obtained by using 20, 30 and 40 particles were equal, while energy values obtained by using 5 and 10 particles were slightly higher.

In order to measure consistency of analysis by PSO algorithm, 100 independent analysis were performed for each loading case. The maximum and the minimum total potential energy values, average of total potential energy values and their standard deviations were given in Table 6. When average and standard deviation

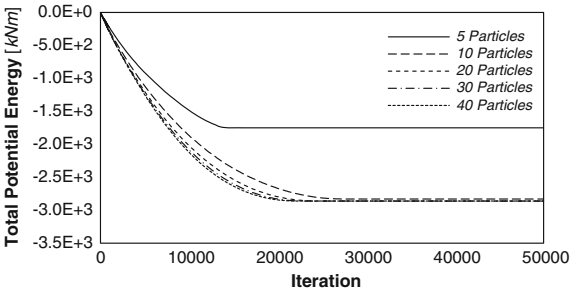


Fig. 12 Change of total potential energy amounts in analyses carried out in the 25-bar space truss with different particle numbers for Loading 3

Table 6 Statistical evaluation of 100 independent analyses by PSO algorithm

	Particle	Total potential energy [*kNm*]			
		Max	Min	Aver.	St. Dev.
Loading 1	5	−3.70726	−3.76451	−3.76388	5.69×10^{-3}
	10	−3.76447	−3.76451	−3.7645	9.06×10^{-6}
	20	−3.76449	−3.76451	−3.7645	7.57×10^{-6}
	30	−3.76449	−3.76451	−3.76451	5.19×10^{-6}
	40	−3.76449	−3.76451	−3.76451	4.84×10^{-6}
Loading 2	5	−1,125.1	−1,439.12	−1,368.99	58.15
	10	−1,444.56	−1,444.57	−1,444.57	1.05×10^{-3}
	20	−1,444.57	−1,444.57	−1,444.57	0.50×10^{-3}
	30	−1,444.57	−1,444.57	−1,444.57	0.54×10^{-3}
	40	−1,444.57	−1,444.57	−1,444.57	0.56×10^{-3}
Loading 3	5	−1,142.41	−1,849.26	−1,536.61	155.09
	10	−2,707.84	−2,860.47	−2,843.86	25.46
	20	−2,860.47	−2,860.48	−2,860.48	2.19×10^{-3}
	30	−2,860.47	−2,860.48	−2,860.48	1.75×10^{-3}
	40	−2,860.48	−2,860.48	−2,860.48	1.23×10^{-3}

values were examined, it was seen that analysis carried out with 20, 30 and 40 particles gave consistent results for all load cases. Usage of 5 particles for Loading 2 and usage of 5 and 10 particles for Loading 3 did not result in minimum energy levels. Also, their standard deviations were slightly higher.

When change of standard deviation was examined in analysis with 20, 30 and 40 particles, bars with higher contribution to potential energy amount had lower standard deviation, and bars with lower contribution to potential energy amount had slightly higher standard deviation (Fig. 13).

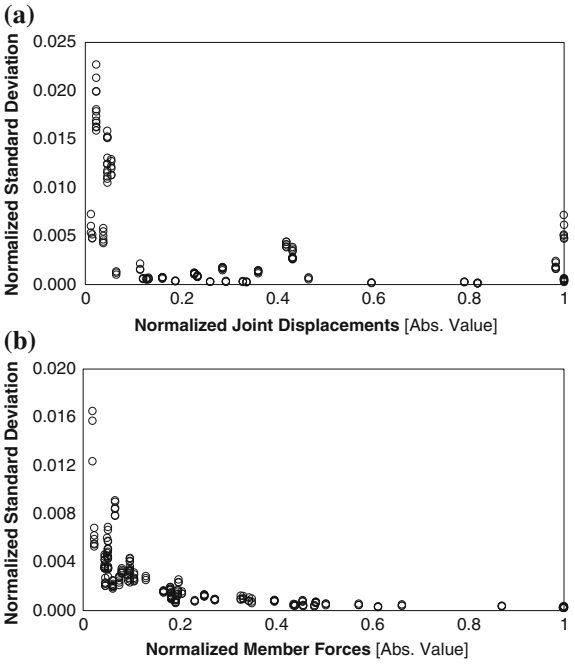


Fig. 13 Normalized standard deviations in 100 independent runs for the 25-bar space truss under Loading 1, solutions with 20, 30 and 40 particles **a** normalized joint displacements **b** normalized member forces

6 Discussion and Conclusion

Two plane and one space truss structure systems with different geometries were examined. In analysis conducted by TPO/MA method, particle swarm optimization algorithm was used. Velocity of particles was limited to $V_{low} = 0.003$ as minimum and $V_{up} = 0.5$ as maximum. $w_{low} = 0.001$ and $w_{up} = 0.7$ values were used for inertia weight. Velocity coefficients were set as $c_1 = c_2 = 2$. Analyses with 5, 10, 20, 30 and 40 particles were conducted so as to determine the influence of number of particles on analysis results. The computations have shown that results with 20 and more particles match with those obtained by other methods. Additionally, it was seen that standard deviations of results with 20 and more particles is lower and more consistent in 100 independent analyses with each number of particles. Energy levels of results with 5 and 10 particles were determined to be greater than energy levels of results with other numbers of particles and by other methods. Besides, standard deviation values were also higher. Therefore, it is suggested that 20 or higher number of particles are to be used in analysis of truss structural systems by PSO method.

By definition, the results of analyses represent nonlinear behavior of the structural system since the method used in this study is based on energy principles. According to results obtained in this study, PSO algorithm gives acceptable results in geometrical nonlinear analysis of truss structural system by TPO/MA method when it is compared to GA, LS, HS algorithms and finite elements method.

References

1. Eberhart, R.C., Kennedy, J.: A new optimizer using particle swarm theory. In: Proceedings of the 6th International Symposium on Micro Machine and Human Science, Nagoya, Japan (1995). doi:10.1109/MHS.1995.494215
2. Oden, J.T.: Mechanics of Elastic Structures. McGraw-Hill, New York (1967)
3. Toklu, Y.C.: Nonlinear analysis of trusses through energy minimization. Comput. Struct. **82** (20–21), 1581–1589 (2004). doi:10.1016/j.compstruc.2004.05.008
4. Merkle, D., Middendorf, M.: Swarm Intelligence. In: Burke, E.K., Kendall, H. (eds.) Search Methodologies Introductory Tutorials in Optimization and Decision Support Techniques, p. 417. Springer, New York (2005)
5. Kennedy, J., Eberhart, R.C.: Particle swarm optimization. In: Proceedings IEEE International Conference on Neural Networks, November–December 1995, vol. 4, p. 1942. Perth, Australia (1995). doi:10.1109/ICNN.1995.488968
6. Ortakcı, Y., Göloğlu, C.: Determination of cluster numbers via particle swarm optimization. In: Proceedings of 14th Conference on Academic Information 1–3 February 2012, Usak University, Usak, Turkey, pp. 335–341 (2012)
7. Shi, Y.H., Eberhart, R.C.: (1998) A modified particle swarm optimizer. In: Proceedings of the IEEE Congress on Evolutionary Computation, pp. 69–73. doi:10.1109/ICEC.1998.699146
8. Mises, R.V.: Über die stabilitätsprobleme der elastizitätstheorie. ZAMM-J Appl. Math. Mech./Z. für Angew Math. und Mechanik **3**(6), 406–422 (1923). doi:10.1002/zamm.19230030602
9. Mises, R.V., Ratzersdorfer, J.: Die knicksicherheit von fachwerken. ZAMM-J. Appl. Math. Mech./Z. für Angew. Math. und Mechanik **5**(3), 218–235 (1925). doi:10.1002/zamm.19250050305
10. Kondoh, K., Atluri, S.N.: Influence of local buckling on global instability: Simplified, large deformation, post-buckling analyses of plane trusses. Comput. Struct. **21**(4), 613–627 (1985). doi:10.1016/0045-7949(85)90140-3
11. Kaveh, A., Rahami, H.: Nonlinear analysis and optimal design of structures via force method and genetic algorithm. Comput. Struct. **84**(12), 770–778 (2006). doi:10.1016/j.compstruc.2006.02.004
12. Kaveh, A., Hassani, M.: Ant colony algorithms for nonlinear analysis and optimal design of structures. Int. J. Optim. Civil Eng. **1**(4), 571–595 (2011)
13. Toklu, Y.C., Bekdaş, G., Temur, R.: Analysis of trusses by total potential optimization method coupled with harmony search. Struct. Eng. Mech. **45**(2), 183–199 (2013). doi:10.12989/sem.2013.45.2.183
14. Venkayya, V.B.: Design of optimum structures. Comput. Struct. **1**(1–2), 265–309 (1971). doi:10.1016/0045-7949(71)90013-7