Studies in Computational Intelligence 592

Satchidananda Dehuri
Alok Kumar Jagadev
Mrutyunjaya Panda *Editors*

Multi-objective Swarm Intelligence

Theoretical Advances and Applications

Springer

Studies in Computational Intelligence

Volume 592

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

Satchidananda Dehuri · Alok Kumar Jagadev
Mrutyunjaya Panda
Editors

Multi-objective Swarm Intelligence

Theoretical Advances and Applications

Springer

Editors
Satchidananda Dehuri
Department of Information and Communication Technology
Fakir Mohan University
Balasore
India

Alok Kumar Jagadev
Department of Computer Science and Engineering
Siksha 'O' Anusandhan University
Bhubaneswar
India

Mrutyunjaya Panda
Gandhi Institute for Technological Advancement
Bhubaneswar
India

ISSN 1860-949X ISSN 1860-9503 (electronic)
Studies in Computational Intelligence
ISBN 978-3-662-46308-6 ISBN 978-3-662-46309-3 (eBook)
DOI 10.1007/978-3-662-46309-3

Library of Congress Control Number: 2015932068

Springer Heidelberg New York Dordrecht London

Printed on acid-free paper

Springer-Verlag GmbH Berlin Heidelberg is part of Springer Science+Business Media (www.springer.com)

Satchidananda Dehuri dedicates this work to his Father: Late Kunja Bihari Dehuri.

Alok Kumar Jagadev dedicates to his Wife and Kids.

Mrutyunjaya Panda dedicates this work to his Parents.

Preface

The purpose of this volume entitled "Multi-objective Swarm Intelligence: Theoretical Advances and Applications" is to attract a wide range of researchers and readers from all fields of science and engineering performing interdisciplinary research based on swarm intelligence methodologies to optimize intractable problems, which are either in the form of single or multi-objective optimization.

Multi-objective optimization deals with the simultaneous optimization of two or more objectives which are normally conflicting with each other. Since multi-objective optimization problems are relatively common in real-world applications, this area has become a very popular research topic since the 1970s. However, the use of bio-inspired meta-heuristics for solving multi-objective optimization problems started in the mid-1980s and remained popular until the mid-1990s. Nevertheless, the effectiveness of multi-objective evolutionary algorithms has made them very popular in a variety of domains.

Swarm intelligence refers to certain population-based meta-heuristics that are inspired on the behavior of groups of entities (i.e., living beings) interacting locally with each other and with their environment. Such interactions produce an emergent behavior that is modeled in a computer in order to solve problems. The two most popular meta-heuristics within swarm intelligence are particle swarm optimization (which simulates a flock of birds seeking food) and ant colony optimization (which simulates the behavior of colonies of real ants that leave their nest looking for food). These two meta-heuristics have become very popular in the last few years, and have been widely used in a variety of optimization tasks, including some related to data mining and knowledge discovery in databases, bio-informatics, computational finance, social networks, etc. However, such work has been mainly focused on single objective optimization models. Furthermore, it is not feasible to prioritize these objectives a priori. The robust nature and population-based approach of swarm intelligence and their cross-hybridization seem to be good methods for multi-criterion problems. Therefore, we are motivated toward multi-objective swarm intelligence and cross-fertilization among methodologies of swarm intelligence to solve various practical problems. In this volume an empirical and theoretical work on ant colony, particle swarm, bacteria foraging, and artificial bee

colony-based optimization has been discussed along with a wide range of applications that are applied in real-world problems.

In the chapter "A Comprehensive Review on Bacteria Foraging Optimization Technique", Selva and Kumar have focused on the behavior of biological bacterial colony followed by the optimization algorithm based on bacterial colony foraging. Further, they explored variations in the components of BFO algorithm (Revised BFO), hybridization of BFO with other Evolutionary Algorithms (Hybrid BFO), and multi-objective BFO. Finally, they have analyzed some applications of BFO algorithm in various domains.

In the chapter "Swarm Intelligence in Multiple and Many Objectives Optimization: A Survey and Topical Study on EEG Signal Analysis", Mishra, Dehuri, and Cho present Swarm Intelligence (SI) methods for optimization of multiple and many objective problems. They have provided the fundamental difference between Multiple and Many Objective Optimization problems. The three forefront swarm intelligence methods, i.e., Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), and Artificial Bee Colony Optimization (ABC) have been deeply studied to understand their ways of solving multiple and many objective problems distinctly. Finally, they have presented a survey on the study of the behavior of real ants, bird flocks, and honey bees in solving EEG signal analysis.

In the chapter "Comparison of Various Approaches in Multi-objective Particle Swarm Optimization (MOPSO): Empirical Study", Devi, Jagadev, and Dehuri present an empirical study with a comparison of various approaches in Multi-objective Particle Swarm Optimization (MOPSO). They have attempted to analyze these methods, discussing their strengths, and weaknesses. A multi-objective particle swarm optimization algorithm, named dynamic multiple swarms in multi-objective PSO, is compared with other well-known PSO techniques in which the number of swarms is adaptively adjusted throughout the search process via dynamic swarm strategy.

Many optimization problems involve selecting the best subset of solution components. Besides, many other optimization problems can be modeled as a subset problem. The chapter "Binary Ant Colony Optimization for Subset Problems" by Nadia Abd-Alsabour focuses on developing a new framework of Ant Colony Optimization (ACO) for optimization problems that require selecting (subset problems- selecting a subset of items from some larger set subject to certain constraints) rather than ordering (where the order between solutions components is important and the next component to be added to a partial solution is affected by the last added component) with an application to feature selection for regression problems as a representative for subset problems. This is addressed through three points as follows: explaining the main guidelines of developing an ant algorithm, demonstrating different solution representations for subset problems using ACO algorithms, and proposing a binary ant algorithm for feature selection for regression problems.

In the chapter "Ant Colony for Locality Foraging in Image Enhancement", Simone, Gadia, Farup, and Rizzi have presented an ant colony for locality foraging

in image enhancement. Further, they have presented a spatial color algorithm called Termite Retinex and the problem of filtering locality in this family of algorithms for image enhancement, inspired by the human vision system. The algorithm is a recent implementation of Retinex with a colony of agents, which uses swarm intelligence to explore the image, determining in this way the locality of its filtering. This results in an unsupervised detail enhancement, dynamic range stretching, color correction, and high dynamic range tone rendering. In the chapter, they describe the characteristic of glocality (glocal = global + local) of image exploration, and after a description of the Retinex spatial color algorithm family, they present the Termite approach and discuss results.

Scheduling is the process of deciding how to commit resources to varieties of possible jobs. It also involves the allocation of resources over a period of time to perform a collection of jobs. Job scheduling is the sequencing of the different operations of a set of jobs in different time intervals to a set of machines. Rani in the chapter "Hybridization of Evolutionary and Swarm Intelligence Techniques for Job Scheduling Problem", uses Hybridization of Evolutionary and Swarm Intelligence Techniques for Job Scheduling Problem. The existing systems have utilized techniques such as Artificial Bee Colony (ABC), Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), Simulated Annealing (SA), etc., and hybrid techniques were derived from the combination of these algorithms. These hybrid techniques mostly concentrate on two factors such as the minimization of the makespan and completion time. The performance of these hybrid techniques lacks in the scheduling process because they have not considered efficient factors like (i) turnaround time, (ii) throughput, and (iii) response time during the job scheduling process. The main aim of this work is to provide a better hybrid job scheduling technique by overcoming the problems that exist in the literary works and to minimize the completion and makespan time. The proposed technique considered the efficient factors (i) turnaround time, (ii) throughput, and (iii) response time which were left out in the existing hybrid techniques for job scheduling process. The performance of the proposed hybrid job scheduling technique was analyzed with the existing hybrid techniques. The experimental results proved that the proposed job scheduling technique attained higher accuracy and efficiency than the existing hybrid techniques.

Odisha, India, November 2014

Satchidananda Dehuri
Alok Kumar Jagadev
Mrutunjaya Panda

Contents

A Comprehensive Review on Bacteria Foraging Optimization Technique

B. Selva Rani and Ch. Aswani Kumar

Abstract Intelligent applications using evolutionary algorithms are becoming famous because of their ability to handle any real time complex and uncertain situations. Swarm intelligence, now-a-days has become a research focus which studies the collective behavior existing among the natural species which lives in group. Bacteria Foraging Optimization (BFO) is an optimization algorithm based on the social intelligence behavior of E.coli bacteria. Literature has witnessed the applications of BFO algorithm and the results reported are promising with regard to its convergence and accuracy. Several studies based on distributed control and optimization also suggested that algorithm based on BFO can be treated as global optimization technique. In this chapter, we have focused on the behavior of biological bacterial colony followed by the optimization algorithm based on bacterial colony foraging. We have also explored variations in the components of BFO algorithm (Revised BFO), hybridization of BFO with other Evolutionary Algorithms (Hybrid BFO) and multi-objective BFO. Finally, we have analyzed some applications of BFO algorithm in various domains.

Keywords Bacteria foraging · Evolutionary computing · Hybrid BFO · Multi-objective optimization · Optimization · Revised BFO · Swarm intelligence

1 Introduction

Optimization is the process of finding the best solution from a set of choices with respect to an objective function and some constraints. Engineering, mathematics, biology, computer science, finance etc. utilize optimization either to maximize

B. Selva Rani · Ch. Aswani Kumar (✉)
School of Information Technology and Engineering,
VIT University, Vellore 632014, India
e-mail: cherukuri@acm.org

B. Selva Rani
e-mail: bselvarani@vit.ac.in

S. Dehuri et al. (eds.), *Multi-objective Swarm Intelligence*,
Studies in Computational Intelligence 592, DOI 10.1007/978-3-662-46309-3_1

or minimize an objective function for which systematic inputs are given. Several classical optimization techniques exist with their own strengths and flaws. Optimization algorithms generally obtain many solutions for any problem; their performance was not satisfactory in several cases due to their iterative property producing non-optimal solution. Evolutionary computing techniques are capable of overcoming the above said problem by obtaining the optimal solutions for both multi-objective and multi-modal optimization problems.

Evolutionary Computing is a branch of Artificial Intelligence which makes it easier to solve any kind of optimization problems. It is encompassed of many algorithms as given in Fig. 1. Evolutionary Algorithms are techniques motivated by biological evolution which involves a search space with solutions as individuals in a population. Selection, cross over, mutation, reproduction and finding the fittest are the main steps involved to solve any problem. Artificial Life simply known as ALife, is a study of examining the system correlated to life, its procedures, and its progress. It makes use of simulations such as computer models, robots, and biochemistry to explore the perception of ALife. Based on the technique used to solve a problem whether it is software-based, hard-ware based or biochemistry-based, ALife is categorized as three approaches:soft, hard or wet. Swarm Intelligence studies the cooperative behavior among the natural species. They frame and follow very simple rules in their entire life cycle, communicate locally in an efficient manner so as to assist each other in any situations. Many researchers reviewed such techniques and their advantages [1–7]. The paradigm consists of two dominant sub-fields (1) Ant Colony Optimization that investigates probabilistic algorithms inspired by the stigmergy

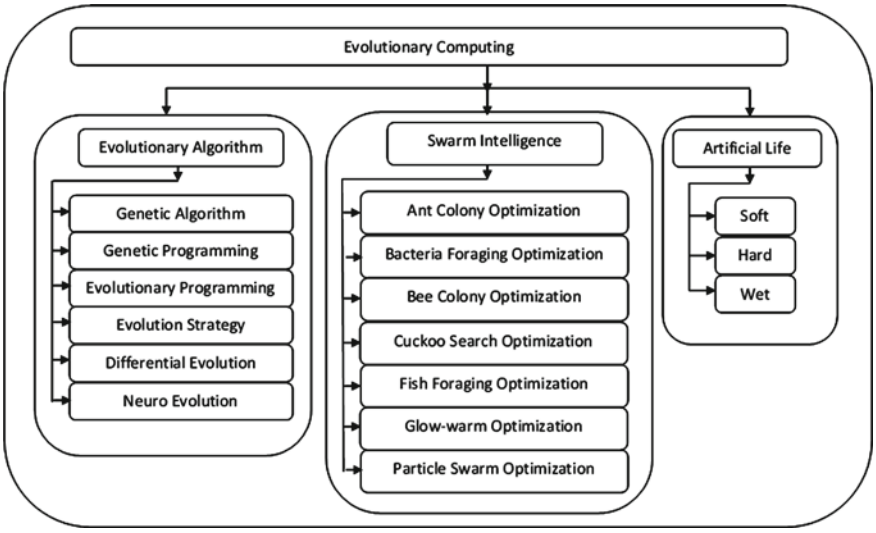


Fig. 1 Popular types of evolutionary computing techniques

and foraging behavior of ants, and (2) Particle Swarm Optimization that investigates probabilistic algorithms inspired by the flocking, schooling and herding [8].

Among their activities, foraging takes an important role in swarms since it affects the survival of fitness in their environment. This is known as search strategy. For many animals, this strategy is usually broken down to three steps namely: searching for or locating the prey, attacking the prey and ingesting the prey [9]. Bacteria, being the first form of life on Earth [10, 11] have developed many potentials in order to communicate among them and live together into structured colonies. They need to cooperate with each other to survive in the unpleasant atmosphere, to reproduce the next generation and for many more activities. If we observe closely the behavior of bacterial colony, their social intelligence recommends us a lot of inspirations for cracking the optimization problems in computer science, mathematics, biology etc. In this chapter, we have explored such intelligence behind E.coli bacterial colonies, their life cycle and its equivalent mathematical model along with the algorithm, variations in bacteria foraging algorithm, hybridization of bacteria foraging algorithm with other algorithms followed by some applications of BFO in various domains.

2 Behavior of Bacterial Colony

Bacteria are tiny organisms with single cell. They appear in different forms whose size is always mentioned as micrometer. Different types of bacteria hold their own characteristics and behavior. Not only among themselves, but also bacteria communicate with other organisms like animals and plants. They are grouped depending on DNA sequencing, shapes, environment they live, the way of obtaining nutrition, cell-wall content etc. It has been proven that all types of Bacteria on Earth are having Social-IQ. Their size is about a very few micrometers and appear in various forms like spirals, sphere or rods. Bacteria are there in almost all the parts of the world. There are approximately 5×10^{30} bacteria on Earth [12] and they play an important role in our day-to-day life. Apart from some harmful types, we depend solely on Bacteria for producing fermented foods (curd, cheese, wine, pickles etc.), for preparing medicines (insulin, pesticides) and even vitamins (riboflavin). Thus, by living in a colony with 10^9–10^{13} bacteria, they know everything about what is happening in and around their colony.

2.1 Phases in the Life Cycle of Bacteria

Bacteria reproduce the next generation asexually by binary fission, dividing itself into two cells and hence both the cells are genetically equal. If we look at the life span of any bacteria, it consists of a cycle with four phases viz. lag phase, growth/exponential phase, maturation/growth phase and death phase. In the lag phase, bacteria are not capable of reproducing. It is the time period taken by the colony to become mature.

In the growth/exponential phase, bacteria start dividing themselves to produce youngsters. Every cell starts binary fission and this growth happens exponentially as the number of cells increases in the population. Actual growth rate depends on the type of the bacterial colony, the environmental conditions and the amount of nutrient available. In the stationary phase, the bacterial growth is restricted due to reasons like insufficient nutrients or environmental changes. Except few, others stop reproduction. In the final phase, bacteria die due to the shortage of nutrients.

2.2 Communication Among Bacteria

Starting from the lag phase throughout their life span, bacteria converse efficiently. Sometimes the technique they used to communicate seems to be simple. By pumping DNA directly into other through a thin tube, bacteria share their hereditary ideas. They also communicate indirectly by sharing signals (in order to make a move, to reproduce, to split into another colony or to change the structure) which is capable of changing the activities of entire bacterial colony with immediate effect. Under normal circumstances, a bacterial colony will stay alive and attempt to reproduce rapidly, but when the environment turns as aggressive to their colony they implement more complex tactics to survive and get ready to be adaptable.

2.3 Types of Bacteria

Depending on DNA sequencing, shapes, environment they live, cell-wall content, the way by which they obtain the nutrition, bacteria are classified as enormous in numbers. Based on Gram Staining method, bacterial communities are classified into two major classes such as Gram-positive and Gram-negative. Gram-positive bacteria generally have a single membrane surrounded by a thick peptidoglycan. Gram-negative bacteria generally possess a thin layer of peptidoglycan between two membranes [13]. E.coli (Escherichia coli) is one such a variety, appears as rods in shape commonly found in warm blooded organisms and human intestines. Majority forms of E.coli are harmless and they help in synthesizing vitamins in human bodies, indicating condemnation in drinking water, producing toxin to cure severe illness etc. In a colony of E.coli bacteria, they forage for nutrients, aggregate as groups, reproduce by binary fission, destroy due to insufficient nutrient or disperse to a new location. Inspired by these cooperative and intelligent behaviors of E.coli colony a mathematical model had been constructed to solve optimization problems in science and engineering by Passino [9] as Bacteria Foraging Optimization (BFO) algorithm.

3 E.coli Bacterial Colonies

3.1 Biological Inspiration

An E.coli bacterium is a cell with thin wall and filamentous organelles. Flagella are wipe-like structures projecting from the cell wall and thrust cells in liquids (swimming) or on surfaces (swarming) so that cells can move toward favorable environments [14]. The structure of a single cell is given in Fig. 2.

Generally, as a group, they will try to find food and avoid harmful phenomena,and when viewed under a microscope, you will get a sense that a type of intelligent behavior has emerged, since they will seem to intentionally move as a group [9]. There are various types of movements in bacteria such as aerotaxis, phototaxis, thermotaxis, magnetotaxis and chemotaxis based on their attraction towards oxygen, light, temperature, magnetic flex and chemicals respectively. Among these, chemotaxis is an important travel for bacterial colony. They swim towards the rich concentration places of food like protein, glucose etc., or away from the poisonous environment with phenol etc. During such foraging, locomotion is achieved by a set of tensile flagella [15]. During chemotaxis an E.coli bacterium can move in two different manner i.e., tumbling and swimming. When a bacterium rotates the flagella in clockwise direction, each flagellum pulls on the body of the cell. As a result there is a minor change in the position of the bacterium, it tumbles. The time gap between the consecutive tumbles is called as tumble interval. When it rotates the flagella in anti-clockwise direction, the total effects are gathered as a bundle as a composite propeller and push the cell body in one direction and it is known as swimming. The time interval between two consecutive runs (swims) is called as run interval. Thus, a bacterium tumbles in a pleasant environment and runs away from a harmful environment. Both the Chemotaxis activities are illustrated in Fig. 3.

In a friendly environment, bacteria colony moves for a longer distance. If the nutrient is sufficient, then they grow and start reproduction by dividing themselves into two halves as in Fig. 4.

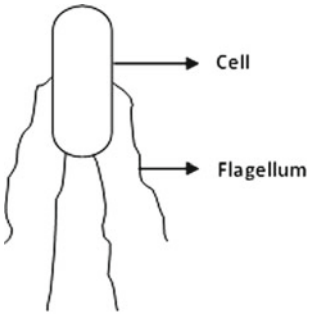


Fig. 2 Structure of an E.coli bacterium

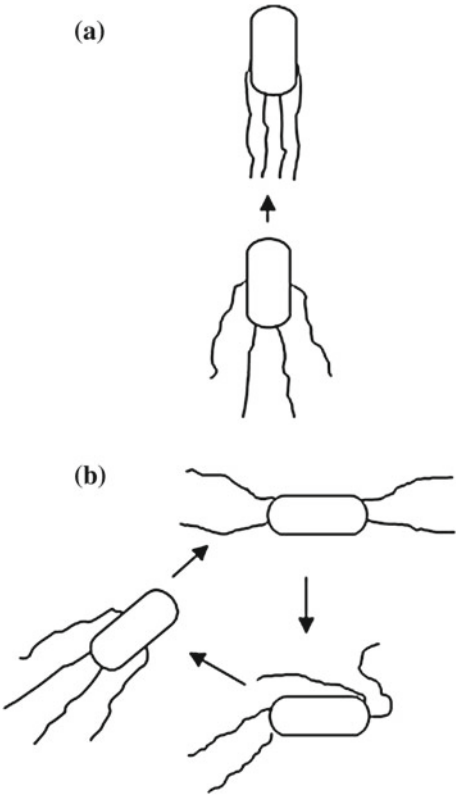


Fig. 3 Chemotaxis behavior of an E.coli bacterium: **a** Swimming. **b** Tumbling

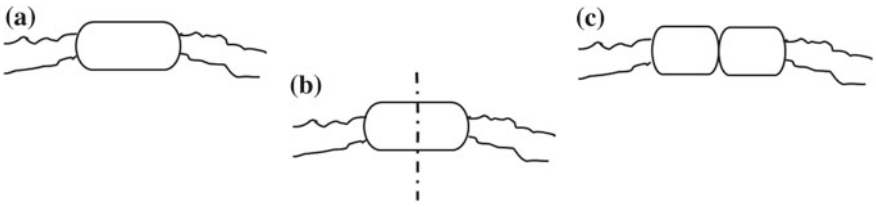


Fig. 4 Reproductions in E.coli by binary fission: **a** Single cell. **b** Binary fission. **c** Identical clone

In case of toxic environments, either the colony would be destroyed or a sub-group would disperse the environment. This concept behind move-or-die option for a bacterial colony is known as elimination-dispersal.

4 Description of BFO

Let us consider the objective function $J(\theta)$, which we want to minimize and $\theta \in \Re^p$, (i.e., θ is a vector of real numbers in p-dimension) where we are not supplied with measurements or analytical description of the gradient $\Delta J(\theta)$. These types of non-gradient optimization problems can be solved by bacterial foraging. Suppose θ is the position of a bacterium and $J(\theta)$ indicates the combined effects of attractants and repellents from the other bacteria in its environment. Based on the environment, nutrient-rich, neutral or noxious, the value of $J(\theta)$ could be less than, equal to or greater than 0 (zero) respectively. The original bacterial foraging contains chemotaxis, a foraging activity that optimizes the movement of the bacteria such that $J(\theta)$ is minimized, i.e., it helps the bacterial colony to climb up to the place wherenutrient concentration is high. The BFO algorithm mimics the four principal mechanisms namely: chemotaxis, swarming, reproduction and elimination-dispersal.

4.1 Chemotaxis

Basically, chemotaxis is a foraging behavior that implements a type of optimization where bacteria try to climb up the nutrient concentration and avoid noxious substances and search for ways out of neutral media [9]. In this E.coli foraging optimization model, an initial population with set of bacteria is defined. The number of chemotactic, reproduction and dispersal-elimination steps is defined during the creation of initial population itself. Passino defined chemotactic as a tumble followed by a tumble or a tumble followed by a run. Let j, k and l denote the indices of chemotactic, reproduction and elimination-dispersal events respectively. Then, the positions of each bacterium in the group of S bacteria at the jth chemotactic, kth reproduction steps and lth elimination-dispersal event are represented as below:

$$P(j,k,l) = \{\theta^i(j,k,l)|i = 1,2,3,\ldots,S\} \tag{1}$$

where, θ^i is the position of the ith bacterium at jth chemotactic, kth reproduction steps and lth elimination-dispersal event. Let the cost of the ith bacterium at this location be denoted as $J(i, j, k, l)$. The length of the lifetime (N_c) of the bacteria is measured by the number of chemotactic steps they take. The number of chemotactic step size of each bacterium is given as $C(i)(>0), i = 1, 2, 3, \ldots, S$. A tumble is represented by generating a unit length in random direction which is termed as $\phi(j)$. Now, the new position of the ith bacterium after a movement (tumbling) is defined as below:

$$\theta^i(j+1,k,l) = \theta^i(j,k,l) + C(i)\phi(j) \tag{2}$$

Here, $C(i)$ indicates the number of steps taken in the random direction indicated by the tumble. If the cost $J(i, j + 1, k, l)$ at the location $\theta^i(j + 1, k, l)$ is lower than

the cost at $\theta^i(j,k,l)$, then the bacterium will move in the same direction with step size $C(i)$. This continues until it meets lower cost values. But it is also restricted to the predefined maximum number of steps, N_s. Hence, in an increasing favorable environment, a bacterium will keep tumbling.

4.2 Swarming

In chemotactic behavior of bacterial colony, the cells swim or tumble individually. There is no cell-to-cell communication. If needed, they have to exchange information via attractant to swarm together or via repellent to isolate each other. The cell-to-cell communication signal by ith bacterium is represented as:

$$\sum_{i=1}^{S} J_{cc}\Big(\theta, \theta^i(j,k,l)\Big), i = 1, 2, 3, \ldots, S \tag{3}$$

The quantity and diffusion rate of attractant laid by an individual is indicated with depth and width of the attractant $d_{attract}$ and $w_{attract}$. Cells are repelling each other in order to indicate that the nutrients in its surrounding are consumed by it. Now repulsion is indicated with height and width of the repellent $h_{repellent}$ and $w_{repellent}$. Collectively such cell-to-cell attraction-repulsion communication is given by the following equation:

$$\begin{aligned} J_{cc}\Big(\theta, P(j,k,l)\Big) &= \sum_{i=1}^{S} J_{cc}\Big(\theta, \theta^i(j,k,l)\Big) \\ &= \sum_{i=1}^{S}\left[- d_{attract} \exp\left(- w_{attract} \sum_{m=1}^{p} (\theta_m - \theta_m^i)^2 \right)\right] \\ &\quad + \sum_{i=1}^{S}\left[h_{repell} \exp\left(- w_{repell} \sum_{m=1}^{p} (\theta_m - \theta_m^i)^2 \right)\right] \end{aligned} \tag{4}$$

where, $J_{cc}\big(\theta, P(j,k,l)\big)$ is the objective function value. Since the cells are releasing the attractant-repulsion chemicals while moving, the above is a time-varying function. Hence, $J_{cc}\big(\theta, P(j,k,l)\big)$ is added with the objective function which has to be minimized actually. If the quantity of the attractant becomes high, the other cells are also attracted towards that cluster of bacteria, which results in swarming. Hence any individual i swarms to search for the nutrient by carefully avoiding poisonous substances and also by keeping a distance from its neighbor without colliding. In swarming,the individuals climb as below:

$$J(i, j, k, l) = J(i, j, k, l) + J_{cc}(\theta, P) \tag{5}$$

4.3 Reproduction

Reproduction of next generation bacteria cells occur after certain number of chemotactic steps say N_c. Let the number of reproductive steps be N_{re}. The bacterium's health is measured as the sum of its fitness values throughout its life, i.e.,

$$\sum_{j=1}^{N_c} J(i, j, k, l) \tag{6}$$

where, N_c is the number of chemotactic steps. The bacteria are sorted in reverse order in terms of their healthiness,higher the cost lowers the health. The first half of the bacteria divides themselves into two equal parts by binary fission (equal division) and the other half die. By this approach, it is assured that the population of the bacteria in the colony is always constant.

4.4 Elimination-Dispersal

This event is also predefined as the number (N_{ed}) of elimination-dispersal steps. p_{ed} is the probability of elimination-dispersal event of a bacterium. The chemotaxis provides the basis for local search, and the reproduction process speeds up the convergence which has been simulated by the classical BFO [16]. But this is not adequate for global optimal search strategy. In case of avoiding a halt in local optima, bacteria will tend to disperse to other location in the search space with respect to the probability p_{ed} which is also defined initially. The hypothesis by Passino [9] states that the rate of occurrence of chemotactic steps is greater than that of reproduction steps which in turn greater than the rate of occurrence of elimination-dispersal events.

4.5 Convergence

The original BFO proposed by Passino [9] do not hold the fittest bacterium for the succeeding generation. As an alternative to the average value, the minimum value among the chemotactic step is maintained for determining the bacterium's health, which in turn speeds up the convergence. And, for swarming the distances of all the bacteria in a new chemotactic stage are evaluated from the globally optimal bacterium to these points and not the distances of each bacterium from the rest of the others as suggested by Passino [17]. More details on increasing the convergence speed by adaptive BFO could be found in [17].

5 Optimization Based on E.coli Bacterial Colony

BFO algorithmhas been modeled as an algorithm as given below:
Step 1: Initialize the parameters

P	Dimension of the search space
S	Total number of bacterium in the initial population
N_c	Number of chemotactic steps
N_s	Length of one swim
N_{re}	Number of reproduction steps
N_{ed}	Number of elimination-dispersal events
P_{ed}	Probability of elimination-dispersal event
$C(i)$	Size of the step taken in each swim/tumble

Step 2: Elimination-dispersal loop $l = l + 1$.
Step 3: Reproduction loop $k = k + 1$.
Step 4: Chemotaxis loop $j = j + 1$.

(a) For $i = 1$ to S take a chemotactic step for bacterium 'i' as below:
(b) Compute the fitness function $J(i, j, k, l)$ as

$$J(i, j, k, l) = J(i, j, k, l) + J_{cc}\Big(\theta^i(j, k, l), P(j, k, l)\Big) \tag{7}$$

(c) Let the result obtained in (b) be stored as below to compare with the values in the consequent runs.

$$J_{last} = J(i, j, k, l) \tag{8}$$

(d) Tumble: Generate a random vector $\Delta(i) \in R^i$ with each element $m(i)$ such that $m = 1, 2, \ldots, n$, a random number in the interval $[-1, 1]$.
(e) Move: Let

$$\theta^i(j + 1, k, l) = \theta^i(j, k, l) + C_i \frac{\Delta(i)}{\sqrt{\Delta^T(i)\Delta(i)}} \tag{9}$$

which is the result of step size $C(i)$ for each bacterium in its tumble direction.
(f) Compute

$$J(i, j + 1, k, l) = J(i, j, k, l) + J_{cc}\big(\theta^i(j + 1, k, l), P(j + 1, k, l)\big) \tag{10}$$

(g) Swim:

(i) Initialize the counter for swim length as m = 0.
(ii) While $m < N_s$ implies that it has not climbed down long,

- Let m = m + 1
- If $J(i, j+1, l, k) < J_{last}$, let $J_{last} = J(i, j+1, k, l)$, then another step of size $C(i)$ in this same direction will be taken as in step (e) and use the new generated $\theta^i(j+1, k, l)$ to compute the next $J(i, j+1, k, l)$.

(iii) Otherwise let $m = N_s$.

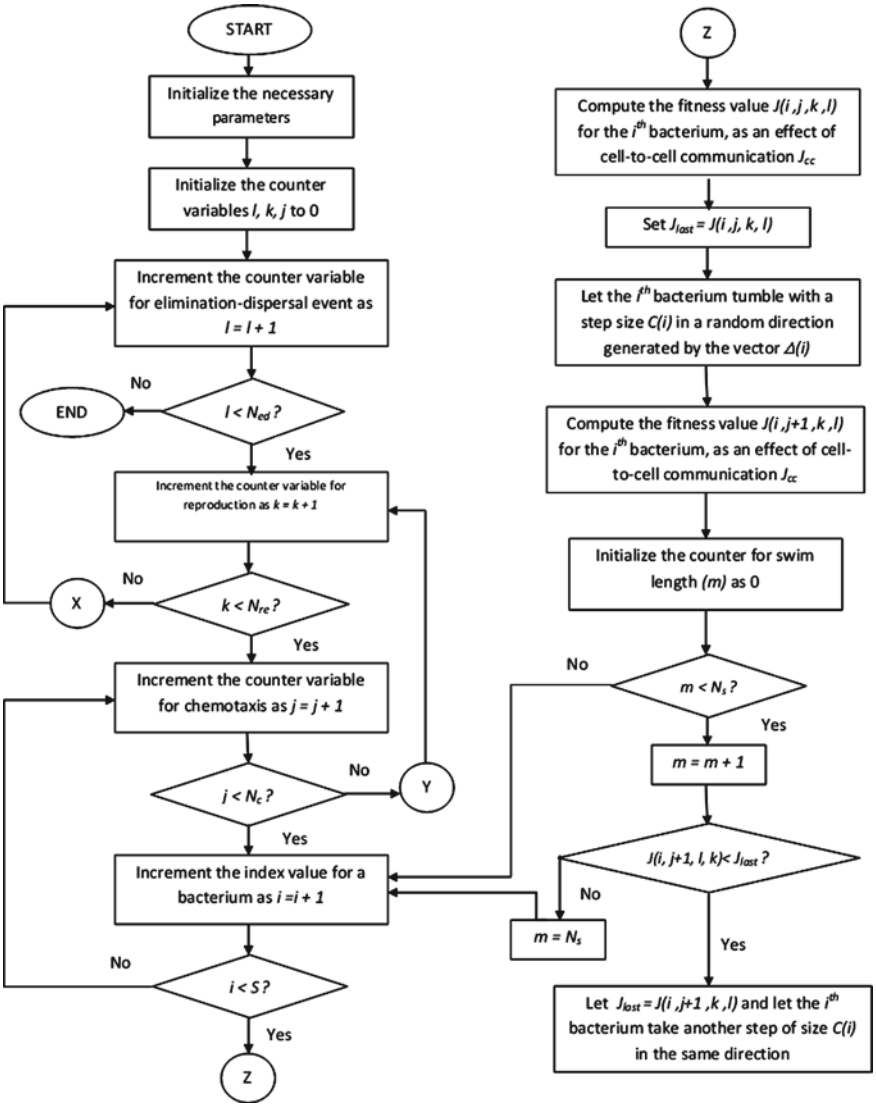


Fig. 5 Flow diagram of BFO algorithm

(h) Consider the next bacterium $(i + 1)$. If $i \neq S$, go to step (b) to process the next bacterium.

Step 5: If $j < N_c$, go to step (3).
Step 6: Reproduction:

(a) For each $i = 1, 2, \ldots S$, 'k' and 'l', compute the health of the bacteria as below and arrange the bacteria in ascending order of J_{health} values.

$$J^i_{health} = \sum_{j=1}^{N_c+1} J(i, j, k, l) \tag{11}$$

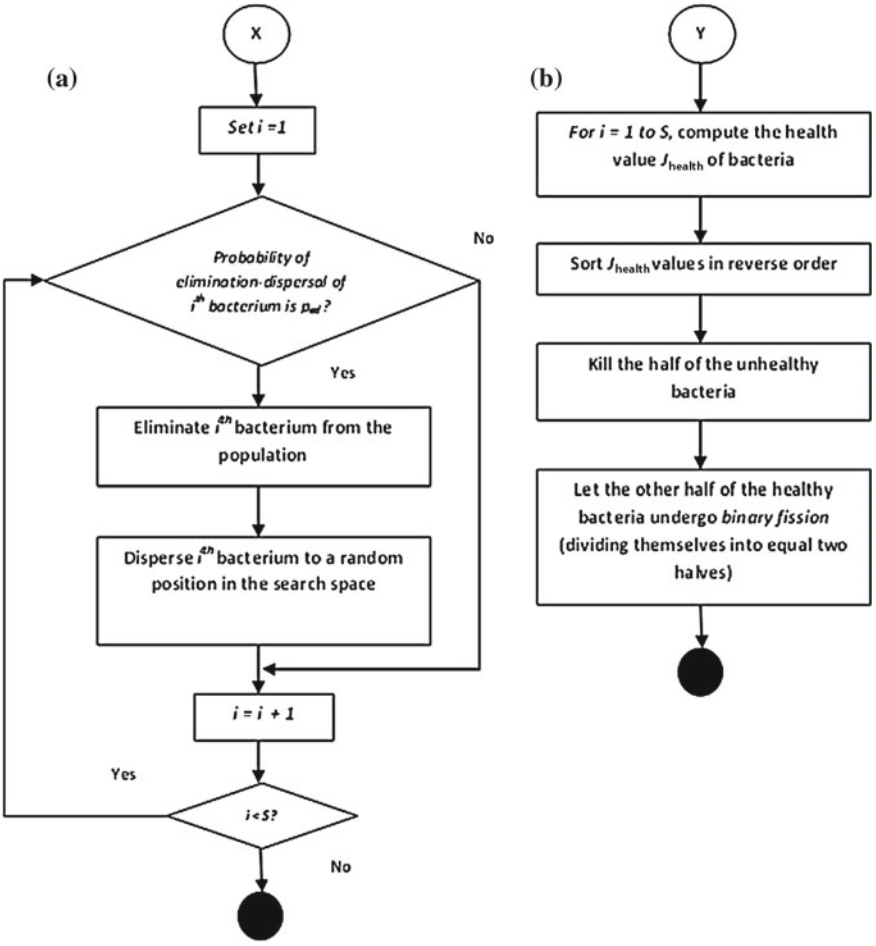


Fig. 6 Sub flows of BFO algorithm: **a** Elimination-dispersal event. **b** Reproduction

(b) The bacteria with the highest J_{health} values will die and others with best health values undergo binary fission and the copies that are made are placed at the same location as their parent.

Step 7: If $k < N_{re}$, then go to step (2).
Step 8: Elimination-dispersal:

For $i = 1, 2, \ldots S$, using the probability p_{ed}, eliminate and disperse the bacterium so that the size of the bacteria population remains constant. To perform this, after elimination of one bacterium, disperse another to a random location on the optimization domain.

If $l < N_{ed}$, then go to step (2),
else End.

We have provided the flow diagram of the heuristic strategy in Fig. 5 and that of elimination-dispersal event and the reproduction step in Fig. 6.

6 Classification of BFO Algorithm

The major success of any algorithm depends on the parameters used in it. We need several parameters to be set for representing the problem or application, solution or search space, boundary of the search space, essential constraints, termination check etc. Static or dynamic control of the values of these parameters plays a crucial role in producing good results. These parameters may be set by any one of the two methods viz. parameter tuning or parameter control. In the former method, parameter values are studied and good ones are identified well before the execution of the algorithm, whereas in the later method, parameter values are changed during the executions. Researchers are interested to experiment various alternatives in theexiting methodologies. Apart from conventional BFO algorithm, there emerged variations in two ways, either by modifying one or more parameters or procedures in the original algorithm (Revised BFO) or by incorporating one or more parameters or procedures from other evolutionary algorithms (Hybrid BFO) as depicted (Fig. 7).

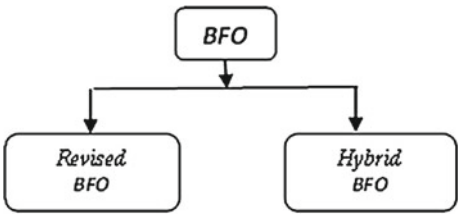


Fig. 7 Classification of BFO algorithm

6.1 Conventional BFO

A problem can obtain its best optimal solution from the existing alternatives if we apply any optimization technique. Biological processes like evolution or searching for food by ants initially motivated several researchers to innovate many new methods for optimizations viz. ant codes and evolutionary algorithms. In 2002, Passino [9] represented the way by which individual bacterium communicates with other bacteria and the group of bacteria forage in their environment for nutrients. These behaviors have been modeled as an optimization process and termed as BFO algorithm. By grouping the behaviors such as chemotaxis, swarming strategy in nutrient or noxious environment, reproduction and elimination-dispersal of bacterial colony, Passino constructed a mathematical model for the foraging strategy in the E.coli colony. Inspired by Passino's work, researchers started solving optimization problems using BFO algorithm which is a promising one among the nature-inspired evolutionary techniques.

6.2 Revised BFO

In the original procedure of BFO algorithm altering the values of the parameters can be termed as Revised BFO. We represented few such variations in BFO algorithm in Fig. 8.

Passino [9] derived an average scheme for making a decision on health of individuals in a bacterial colony. But this might not maintain the fittest individual for the following generations. Hence, it has been suggested to compute the swarming distance by referring to the distance of individuals with their neighbors. Tripathy and Mishra [18] proposed an improved BFO algorithm in which they retained the minimum values among the cost of each individual bacterium to decide its health, which in turn speedup the convergence towards the global optimum. They have also evaluated the swarming distance of individuals by referring the global optimal and shown the improved results over the original BFO algorithm.

Li et al. [19] proposed a revised BFO, BFAVP (Bacterial Foraging Algorithm with Varying Population). This paper explored the mechanisms behind the behaviors of bacterial colony such as chemotaxis, proliferation, quorum sensing etc. and suggested the concept of varying population in the colony. Conventional BFO was described using the chemotaxis behavior of the bacterial colony. Deviating from this, the authors included the other two behaviors (quorum sensing and proliferation) which permit the variations in the population of the colony in each generation. They applied BFAVP to solve OPF (Optimal Power Flow) problem that minimizes the cost of fuel in a power system and provided the merits by comparing the results obtained by PSO.

Based on [18, 19], Dasgupta et al. [17] analyzed the performance of two adaptive BFO techniques by comparing the performance with classical BFO, Bacterial Swarm

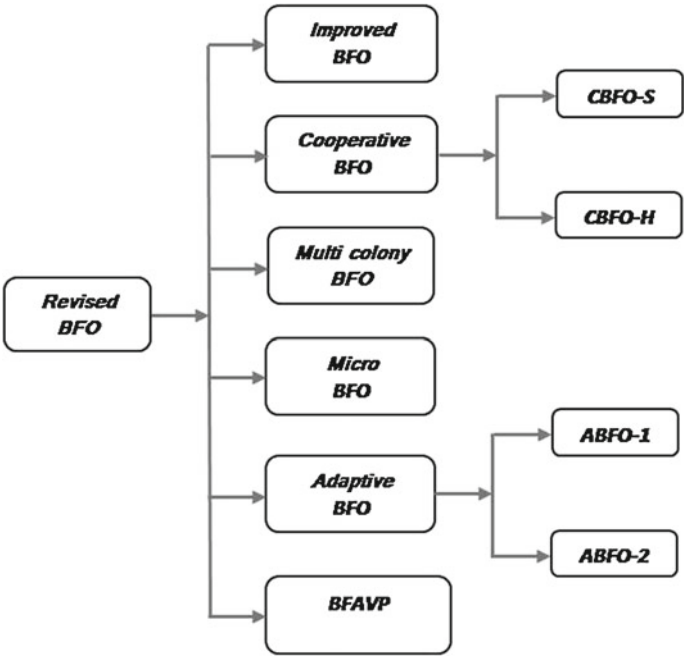


Fig. 8 Revised BFO algorithms

Optimization (BSO), GA and a variant of PSO. All the parameters of the original BFO are kept in ABFOA1 and ABFOA2 except the number of chemotactic steps. By considering the worthy of the final solution, the frequency in reaching the optimal solution and the speed of convergence it has been proved that both the adaptive BFO algorithms promised better results. Table 1 summarizes the above.

The two variations in the conventional BFO algorithm proposed by Chen et al. [16, 20] are Cooperative BFO algorithm and Multi-colony BFO algorithm. The former one is designed to have serial heterogeneous cooperation on the implicit

Table 1 Revised BFO algorithms

Category	Added/Modified features	Reference(s)
Improved BFO	Retained minimum cost	[18]
BFA-varying population	Quorum sensing, proliferation	[19]
Adaptive BFO	Varying chemotactic steps	[17]
Cooperative BFO	Explicit decomposition of search space	[16]
Multi colony BFO	Multiple colonies	[20]
Micro BFO	Smaller population	[21]

space decomposition level and the serial heterogeneous cooperation on the hybrid space decomposition level [16], whereas the later one [20] is designed to havemultiple colonies in which both intra and inter colony communications take place.

Dasgupta [21] introduced Micro-BFO which progress with a very smallpopulation. They modified the classical BFO algorithm in such a way that the best fitness bacterium is kept as it is in Micro-BFO algorithm whereas other bacteria are re-initialized.

6.3 Hybrid BFO

Experimentation with several benchmark functions reveal that the BFO algorithm possesses a poor convergence behavior over multi-modal and rough fitness landscapes as compared to other naturally inspired optimization techniques [22].

Hybrid Models are constructed by combining the methods of two techniques to solve optimization problems, which in turn gives more advantages when compared with the single system. There are proposals which combine the components of the original BFO algorithm with other evolutionary techniques to perform optimization more efficiently. Fusing the notion of other algorithm with the components of the BFO algorithm can be termed as Hybrid BFO. Some of such hybrid BFO techniques are shown in Fig. 9. Kim et al. [23] proposed a hybrid model in which the idea of genetic algorithm has been combined with the concept bacteria foraging to solve function optimization problems. Genetic Algorithm is a heuristic search method which imitates the survival of fittest. It involves selection, cross over and mutation operations. The candidate solutions of an optimization problem are initialized to a set

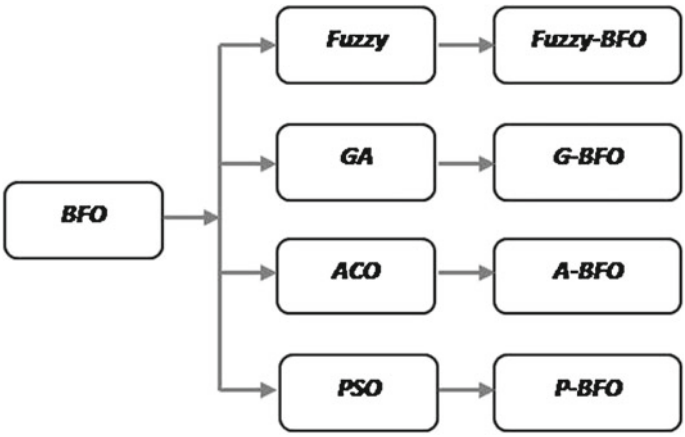


Fig. 9 Hybrid models of BFO with fuzzy, GA, ACO and PSO

of chromosomes. In each iteration, new solutions are generated by executing mutation and crossover operations and the fitness of the emerging solutions are estimated.

Among all, the set of solutions with certain threshold value will be chosen for the next generation. This procedure is repeated until a best individual is found. Genetic algorithms are ubiquitous now-a-days, having been successfully applied to numerous problems from different domains, including optimization, automatic programming, machine learning, operations research, bioinformatics, and social systems [23]. Bacterial Foraging is also an emerging technique with is inspired by the search behavior of bacterial colony in the environment. A bacterium with the help of flagella swims and tumbles alternatively according to the suitability of its surroundings. These movements are similar to the searching strategy. In pleasant environments, they move longer, but in unpleasant environments, they start repulsion. They used to send cell-to-cell signals in case of any disaster. An optimization problem can be modeled using bacterial foraging by defining the problem features into bacteria characteristics such as the position of bacteria, the number of chemotactic steps, reproduction steps, population density etc. and operations like swimming, tumbling, binary fission, dispersal and elimination. Kim [24] introduced a simple cross-over strategy in the hybrid version of BF-GA approach to tune the PID controller with improved learning and speedy convergence towards the optimum. The above said Hybrid BFO algorithms are summarized in Table 2.

Korani [27] introduced another hybrid approach of combining BFO with PSO for tuning the PID controller. PSO is also kind of nature-inspired computing, pioneered by Eberhart and Kennedy, which was inspired by the cooperative behavior of a flock of birds. The main advantage of PSO algorithm is its rapid convergence when compared with Genetic Algorithm. The basic PSO model consists of a swarm of particles, which are initialized with a population of random candidate solutions [30]. Using a utility measure called as a cost function or fitness function, the solution of each particle would be evaluated. In each iteration their best solution so far found are recorded as *gBest*, which is the best solution among the entire set and *pBest*, which

Table 2 Hybrid BFO algorithms

Combined with	Research goal	Reference(s)
Genetic algorithm	Global optimization	[23]
Genetic algorithm	PID controller tuning	[24]
Genetic algorithm	Automated experimental control design	[25]
Fuzzy	Automatic circle detection on images	[26]
Particle swarm optimization	PID controller tuning	[27]
Particle swarm optimization	Resonant frequency of rectangular micro strip antenna	[28]
Particle swarm optimization	Power system stabilizers design	[29]

is the best solution of each particle so far in the iterations. According to *gBest* and *pBest* the velocity and position of each particle will be updated. The particles move through the solution space to search for the best solution until certain number of iterations or the solution is found. BFO algorithm is good in finding the best solution but it delays in reaching the global best due to random tumbling. But PSO algorithm is good in communicating the information globally among all the particles. Hence Korani proposed a hybrid approach BF-PS optimization in which bacteria tumbling is defined with respect to the global best position and individual bacterium's best position.

BFO hybridized with PSO as VMBFO (Velocity Modulated BFO) Technique proposed by Gollapudia et al. [28]. While calculating the fitness value of each bacterium, a new velocity updating step had been included. Based on the new velocity, the position of individual bacterium was updated and thus the global best position was communicated. Thus, it had been proved that there is a reduction in convergence time with high accuracy.

7 Multi-objective Optimization Based on BF

7.1 Multi-objective Optimization

Multi-objective optimization involves the simultaneous optimization of several incommensurable and often competing objectives [31], i.e., it is the field of solving optimization problems having more than one criteria. Objectives are represented as a vector and the space in which the objective vector belongs is called the objective space [32]. Multi-objective optimization is also called as Pareto optimization or multi criteria/attribute optimization. Applications of multi-objective optimization varies from economics, science, finance, design and control in engineering and electrical systems, where an optimal decision is needed depending on more than one conflicting objectives. Here are two practical examples of multi-objective optimization problem,

- minimizing the energy consumption while maximizing the performance of an air conditioner and
- minimizing fuel consumption while maximizing the speed and performance of a car.

There cannot be a single solution for any significant multi-objective optimization problem, which optimizes all the objectives. Since they are conflict in nature, there exists a possibility of having a number of optimal solutions. A solution is called Pareto optimal if no objective values can be improved without degrading some values of other objective functions.

7.2 Multi-objective BFO Algorithm

Several methods based on artificial intelligent techniques have been proposed to solve such problems including GA and PSO. BFO algorithm could be extended to solve any problem with more than one objective. Guzman et al. [33] developed a multi-objective optimization algorithm to solve problems with more than one objective based on chemotaxis behavior of E.coli bacterial colony. They used a simple principle to change the positions of members in the bacterial colony, a communication strategy between strong and weak members and a nondominant sorting procedure to explore the search space in order to find more than one optimal solution. The performance of this algorithm was compared with GA and PSO while solving bench mark functions and proved to be good. Panigrahi et al. [34] discussed that economic dispatch is a constrained optimization problem in power system to distribute the load demand among the committed generators economically [34]. Niu et al. [35] derived BFO to solve multiple objectives of a problem. Usually during reproduction bacteria with the smaller health values are chosen for the nextgeneration. Meanwhile, the main goal of multi-objective optimization problems is to obtain a superior non-dominated front which is closed to the true Pareto front [35]. By identifying such features and integrating health sorting with Pareto dominance it has been proved that Pareto-optimal set of problems can be solved and better results can be obtained. We can find recent examples of solving multi-objective optimization problems using BFO in [36, 37].

8 An Overview of BFO Applications

BFO is an emerging as well as promising optimization algorithm which provides a great impact on the expected outcomes. Interested researchers have startedanalyzing the behaviors of the bacterial colonies and using the BFO algorithm for solving optimization problems in the domains such as mathematics, science, electrical engineering, computer science etc. Most of the optimization problems which were solved by conventional methods as well as other nature-inspired methods (ACO, ABC, and PSO etc.) can also be solved by BFO algorithm by identifying the proper objective function and the exact values for the parameters. Since BFO is having the cell-to-cell communication capabilities, they can gather information from their environment and neighbors very quickly. The individuals infer knowledge from the collected information and learn new things by referring to the past experience which helps to speed up the search towards the optimum. Hence, BFO has been proved as one of the promising nature-inspired optimization algorithm. We have provided an overview on some of the recent applications of BFO by the researchers in Table 3.

Abharian et al. [38] presented an optimal nonlinear stochastic controller based on a BFO method to decrease the number of infected cells in presence of stochastic parameters of HIV dynamic. Huge amount of protein sequences are accumulated by

Table 3 Applications of BFO in various domains

Application area	Research goal	Reference(s)
Biomedical	Control of HIV-1 infection	[38]
Biology	Protein sequence analysis	[39]
Business	Liquidity risk	[40]
Computer science	Image segmentation	[41]
	Edge detection in images	[42]
	Multilevel thresholding of MR brain images	[43]
	Face recognition	[44]
	Image edge detection	[45]
	Resource scheduling	[12]
Electrical and electronics	Power systems stabilizers design	[29]
	Adaptive control of DC motor	[46]
	Congestion management in generators	[47]
	Design of TCSC damping controller	[48]
	Dynamic economic load dispatch	[49]
	Design of PSS and SVC	[50]
	SVC damping controller design	[51]
	Tuning the I-PD controller	[52]
	IDD controller in a hydro thermal system	[53]
	Optimal dynamic load dispatch	[54]
	Multiple distributed generators	[55]
	Efficiency estimation of induction motor	[56]
	Optimal distributed power generation	[57]
	Manufacturing cell formation	[58, 59]
Numeric optimization	Optimization of global functions and tuning a PID controller	[24]
	Optimization of global functions and resonant frequency	[28]

the researchers in a database for analysis. This results in a need for synchronization of resources for searching in the database. Vivekanandana et al. [39] proposed the usage of BFO algorithm for identifying similar patterns in protein sequences in an effective manner by reducing the cost of search. In finance industries maximizing the expected return is termed as portfolio optimization which is a non-linear problem and contains many local optimal values.

Niu et al. [40] presented a BFO approach to solve portfolio optimization problem. In the field of computer science, many real world problems have been solved with the aid of BFO and its variations like image segmentation [41], edge detection in images [42], multi-level thresholding of MR brain images [43], face recognition [44], image edge detection [45] and resource scheduling [12]. Applications of BFO dominate in the field of electrical engineering. It has been applied in power systems stabilizer design [29], adaptive control of DC motor [46], congestion management in generators [47], design of TCSC damping controller [48], dynamic load dispatch [49, 54], SVC controller design [50, 51], tuning the I-PD Controller [52], IDD controller in a hydro thermal system [53], multiple distributed generators [55], efficiency estimation of Induction motor [56], distributed power generation [57] and many more. Nouria et al. [58, 59] introduced BFO in the field of manufacturing. They proposed BFO in cellular manufacturing to solve the cell construction problem by considering cell load variations along with a number of exceptional elements. BFO has been applied to work out many number of global optimization functions which are used as test functions to prove the efficiency of optimization algorithms [24, 28].

9 Conclusion

In this chapter, we have described the biological background of bacteria, cooperative behavior of E.coli bacterial colony and a mapping of their behaviors with mathematical models. We expressed the variations in the parameters of BFO, hybridization of BFO with other evolutionary techniques, solving multi-objective optimization using BFO and some applications of BFO algorithm in different domains. It is an evident that this chapter provided a better view on the potentials of BFO algorithm. It has been already proved and accepted that BFO algorithm is a powerful tool to solve broad range of engineering applications. Several researchers suggested that BFO can be used to obtain optimal solutions by designing more complicated variations and hybridization of BFO in different domains which now paved a new way in the future research. BFO algorithm can be hybridized with other existing techniques to improve their efficiency and robustness.

Appendix

(See Table 4)

Table 4 Nomenclature

Symbol	Description
P	Dimension of the search space
S	Total number of bacteria in the initial population
N_c	Number of chemotactic steps
N_s	Length of one swim
N_{re}	Number of reproduction steps
N_{ed}	Number of elimination-dispersal events
$C(i)$	Size of the step taken in each swim/tumble
$P(j,k,l)$	Position of ith bacterium in jth chemotactic, kth reproduction and lth elimination-dispersal steps
$J(i,j,k,l)$	Fitness of ith bacterium in jth chemotactic, kth reproduction and lth elimination-dispersal steps
J_{last}	Previous fitness value of the bacterium
$\Delta(i)$	Random vector for the ith bacterium
$\theta(i)$	Position of the ith bacterium
$\phi(j)$	Unit length in random direction of the bacterium in the jth chemotactic step
J_{cc}	Cell-to-cell communication signal
$d_{attract}$	Depth of the attractant signal
$w_{attract}$	Width of the attractant signal
h_{repell}	Height of the repellent signal
w_{repell}	Width of the repellent signal
J_{health}	Health of a bacterium
p_{ed}	Probability of the elimination-dispersal event of a bacterium

References

1. Lei, W., Qi, K., Qi-di, W.: Nature-inspired computation effective realization of artificial intelligence. SETP **27**(5), 126–34 (2007)
2. de Castro, L.N.: Fundamentals of natural computing: an overview. Phys. Life Rev. **4**, 1–36 (2007)
3. Zang, H., Zhang, S., Hapeshi, K.: A review of nature-inspired algorithms. J. Bionic Eng. **7**(Suppl.), S232–7 (2010)
4. Schut, M.C.: On model design for simulation of collective intelligence. Inf. Sci. **180**, 132–55 (2010)
5. Dressler, F., Akan, O.B.: A survey on bio-inspired networking. Comput. Netw. **54**, 81–900 (2010)

6. Agrawal, V., Sharma, H., Bansal, J.C.: Bacteria foraging optimization: a survey. In: Proceedings of International Conference on SocProS 2011. AISC130, pp. 227–242 (2012)
7. El-Abd, M.: Performance assessment of foraging algorithms vs. evolutionary algorithms. Inf. Sci. **182**, 243–3 (2012)
8. Brownlee, J.: Clever algorithms nature inspired programming recipes (2012)
9. Passino, K.M.: Biomimicry of bacterial foraging for distributed optimization and control. IEEE Control Syst. Mag. **5**(3), 52–67 (2002)
10. Liebes, S.: A Walk Through Time: From Stardust to Us. Wiley (1998)
11. Margulies, L., Dolan, M.F.: Early Life: Evolution on the Precambrian Earth. Jones and Bartlett (2002)
12. Rajni, Chana, I.: Bacterial foraging based hyper-heuristic for resource scheduling in grid computing. Future Gener. Comput. Syst. **29**(3), 751–762 (2013)
13. Pedregal, P.: Introduction to Optimization. Springer International Edition (2004)
14. Terashima, H., Kojima, S., Homma, M.: Flagellar motility in bacteria: structure and function of flagellar motor. In: International Review of Cell and Molecular Biology, vol. 270 (2008)
15. Das, S., Biswas, A., Dasgupta, S., Abraham, A.: Bacterial Foraging Optimization Algorithm: Theoretical Foundations, Analysis, and Applications, pp. 23–55. Springer, Berlin (2009)
16. Chen, H., Zhu, Y., Hu, K.: Cooperative bacterial foraging optimization. Discret. Dyn. Nat. Soc. **2009**, 17 (2009)
17. Dasgupta, S., Das, S., Abraham, A., Biswas, A.: Adaptive computational chemotaxis in bacterial foraging optimization: an analysis. IEEE Trans. Evolut. Comput. **13**(4), 919–41 (2009)
18. Tripathy, M., Mishra, S., Lai, L.L., Zhang, Q.P.: Transmission loss reduction based on FACTS and bacteria foraging algorithm. In: Proceedings of PPSN, pp. 222–231 (2006)
19. Li, M.S., Tang, W.J., Tang, W.H., Wu, Q.H., Saunders, J.R.: Bacteria foraging algorithm with varying population for optimal power flow. In: Proceedings of EvoWorkshops 2007. LNCS, vol. 4448, pp. 32–41 (2007)
20. Chen, H., Zhu, Y., Hu, K.: Multi-colony bacteria foraging optimization with cell-to-cell communication for RFID network planning. Appl. Soft Comput. **10**, 539–47 (2010)
21. Dasgupta, S., Biswas, A., Das, S., Panigrahi, B.K., Abraham, A.: A Micro-Bacterial Foraging Algorithm for High-Dimensional Optimization (2009)
22. Biswas, A., Dasgupta, S., Das, S., Abraham, A.: Synergy of PSO and bacterial foraging optimization: a comparative study on numerical benchmarks. In: Proceedings 2nd International Symposium Hybrid Artificial Intelligent Systems (HAIS). Advances Soft Computing Series, Innovations in Hybrid Intelligent Systems. ASC, vol. 44, pp. 255–263. Springer, Germany (2007)
23. Kim, D.H., Abraham, A., Cho, J.H.: A hybrid genetic algorithm and bacterial foraging approach for global optimization. Inf. Sci. **177**, 3918–37 (2007)
24. Kim, D.H.: Hybrid GA-BF based intelligent PID controller tuning for AVR system. Appl. Soft Comput. **11**, 11–22 (2011)
25. Okaeme, N.A., Zanchetta, P.: Hybrid bacterial foraging optimization strategy for automated experimental control design in electrical drives. IEEE Trans. Ind. Inf. **9**, 668–8 (2013)
26. Dasgupta, S., Biswas, A., Das, S., Abraham, A.: Automatic circle detection on images with an adaptive bacterial foraging algorithm. In: GECCO'08, Atlanta, 12–16 July 2008
27. Korani, W.: Bacterial foraging oriented by particle swarm optimization strategy for PID tuning. In: GECCO'08 Proceedings of the Genetic and Evolutionary Computation Conference. ACM, pp. 1823–1826. Atlanta (2008)
28. Gollapudi, S.V.R.S., Pattnaika, S.S., Bajpaib, O.P., Devi, S., Bakwad, K.M.: Velocity modulated bacterial foraging optimization technique (VMBFO). Appl. Soft Comput. **11**, 154–65 (2011)
29. Abd-Elazim, S.M., Ali, E.S.: A hybrid particle swarm optimization and bacterial foraging for optimal power system stabilizers design. Electr. Power Energy Syst. **46**, 334–41 (2013)
30. Abraham, A., Guo, H., Liu, H.: Swarm intelligence: foundations, perspectives and applications. Stud. Comput. Intell. (SCI) **26**, 3–25 (2006)
31. Deb, K.: Multi-objective Optimization using Evolutionary Algorithms. Wiley, Chichester (2001)

32. Caramia, M., Dell Olmo, P.: Multi-objective Management in Freight Logistics Increasing Capacity, Service Level and Safety with Optimization Algorithms, Springer, London (2008). ISBN: 978-1-84800-381-1
33. Guzman, M.A., Delgado, A., De Carvalho, J.: A novel multiobjective optimization algorithm based on bacterial chemotaxis. Eng. Appl. Artif. Intell. **23**(3), 292–301 (2010)
34. Panigrahi, B.K., Pandi, V.R., Sharma, R., Das, S., Das, S.: Multiobjective bacteria foraging algorithm for electrical load dispatch problem. Energy Convers. Manag. **52**, 1334–42 (2011)
35. Niu, B., Wang, H., Wang, J., Tan, L.: Multi-objective bacterial foraging optimization. Neurocomputing **116**, 336–45 (2013)
36. Daryabeigi, E., Zafari, A., Shamshirband, S., Anuar, N.B., Kiah, M.L.M.: Calculation of optimal induction heater capacitance based on the smart bacterial foraging algorithm. Electr. Power Energy Syst. **61**, 326–34 (2014)
37. Daryabeigi, E., Dehkordi, B.M.: Smart bacterial foraging algorithm based controller for speed control of switched reluctance motor drives. Electr. Power Energy Syst. **62**, 364–73 (2014)
38. Abharian, A.E., Sarabi, S.Z., Yomi, M.: Optimal sigmoid nonlinear stochastic control of HIV-1 infection basedon bacteria foraging optimization method. Biomed. Signal Process. Control **104**, 184–91 (2013)
39. Vivekanandana, K., Ramyachitra, D.: Bacteria foraging optimization for protein sequence analysis on the grid. Future Gener. Comput. Syst. **28**, 647–56 (2012)
40. Niu, B., Fan, Y., Xiao, H., Xue, B.: Bacterial foraging based approaches to portfolio optimization with liquidity risk. Neurocomputing **98**, 90–100 (2012)
41. Sanyal, N., Chatterjee, A., Munshi, S.: An adaptive bacterial foraging algorithm for fuzzy entropy based image segmentation. Expert Syst. Appl. **38**, 15489–98 (2011)
42. Verma, O.P., Hanmandlu, M., Kumar, P., Chhabra, S., Jindal, A.: A novel bacterial foraging technique for edge detection. Pattern Recognit. Lett. **32**, 1187–96 (2011)
43. Sathya, P.D., Kayalvizhi, R.: Amended bacterial foraging algorithm for multilevel thresholding of magnetic resonance brain images. Measurement **44**, 1828–8 (2011)
44. Panda, R., Naik, M.K., Panigrahi, B.K.: Face recognition using bacterial foraging strategy. Swarm Evol. Comput. **1**, 138–46 (2011)
45. Verma, O.P., Sharmab, R., Kumar, D.: Binarization based image edge detection using bacterial foraging algorithm. Procedia Technol. **6**, 315–23 (2012)
46. Bhushan, B., Singh, M.: Adaptive control of DC motor using bacterial foraging algorithm. Appl. Soft Comput. **11**, 4913–20 (2011)
47. Venkaiah, Ch., Vinod Kumar, D.M.: Fuzzy adaptive bacterial foraging congestion management using sensitivity based optimal active power re-scheduling of generators. Appl. Soft Comput. **11**, 4921–30 (2011)
48. Ali, E.S., Abd-Elazim, S.M.: TCSC damping controller design based on bacteria foraging optimization algorithm for a multimachine power system. Electr. Power Energy Syst. **37**, 23–30 (2012)
49. Vaisakh, K., Praveena, P., Rama Mohana Rao, S., Meah, K.: Solving dynamic economic dispatch problem with security constraints using bacterial foraging PSO-DE algorithm. Electr. Power Energy Syst. **39**(1), 56–67 (2012)
50. Abd-Elazim, S.M., Ali, E.S.: Coordinated design of PSSs and SVC via bacteria foraging optimization algorithm in a multimachine power system. Electr. Power Energy Syst. **41**, 44–53 (2012)
51. Abd-Elazim, S.M., Ali, E.S.: Bacteria foraging optimization algorithm based SVC damping controller design for power system stability enhancement. Electr. Power Energy Syst. **43**, 933–40 (2012)
52. Rajinikant, V., Latha, K.: I-PD controller tuning for unstable system using bacterial foraging algorithm: a study based on various error criterion. Appl. Comput. Intell. Soft Comput. **2012**, Article ID 329389 (2012)
53. Saikia, L.C., Sinha, N., Nanda, J.: Maiden application of bacterial foraging based fuzzy IDD controller in AGC of a multi-area hydrothermal system. Electr. Power Energy Syst. **45**(1), 98–106 (2013)

54. Azizipanah-Abarghooee, R.: A new hybrid bacterial foraging and simplified swarm optimization algorithm for practical optimal dynamic load dispatch. Electr. Power Energy Syst. **49**, 414–429 (2013)
55. Mohamed Imran, A., Kowsalya, M.: Optimal size and siting of multiple distributed generators in distribution system using bacterial foraging optimization. Swarm Evol. Comput. **15**, 58–65 (2013)
56. Santos, V.S., Felipe, P.V., Sarduy, J.G.: Bacterial foraging algorithm application for induction motor field efficiency estimation under unbalanced voltages. Measurement **46**, 2232–7 (2013)
57. Devi, S., Geethanjali, M.: Application of modified bacterial foraging optimization algorithm for optimal placement and sizing of distributed generation. Expert Syst. Appl. **41**, 2772–81 (2014)
58. Nouria, H., Hong, T.S.: A bacteria foraging algorithm based cell formation considering operation time. J. Manuf. Syst. **31**, 326–6 (2012)
59. Nouria, H., Hong, T.S.: Development of bacteria foraging optimization algorithm for cell formation in cellular manufacturing system considering cell load variations. J. Manuf. Syst. **32**, 20–31 (2013)

Swarm Intelligence in Multiple and Many Objectives Optimization: A Survey and Topical Study on EEG Signal Analysis

B.S.P. Mishra, Satchidanand Dehuri and Sung-Bae Cho

Abstract This paper systematically presents the Swarm Intelligence (SI) methods for optimization of multiple and many objective problems. The fundamental difference of Multiple and Many Objective Optimization problems have been studied very rigorously. The three forefront swarm intelligence methods, i.e., Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), and Artificial Bee Colony Optimization (ABC) has been deeply studied to understand their ways of solving multiple and many objective problems distinctly. A pragmatic topical study on the behavior of real ants, bird flocks, and honey bees in solving EEG signal analysis completes the survey followed by discussion and extensive number of relevant references.

1 Introduction

Multiple and many objective optimization problems are alarming of high importance in recent scientific and the industrial world [1]. Some of the practical optimization problems include flight rescheduling [2], shape optimization [3], mobile network design [4], and minimization of shooting failure probability in the weapon target assignment [5], etc. In practice, the number of objectives in a multiple optimization problem are restricted with ≤ 3. However, number of objectives are at least 4 in the case of many objective optimization. Irrespective of the number of objects i.e.,

B.S.P. Mishra (✉)
School of Computer Engineering, KIIT University, Bhubaneswar-24, Odisha, India
e-mail: mishra.bsp@gmail.com

S. Dehuri
Department of Systems Engineering, Ajou University, San 5, Woncheon-dong, Yeongtong-gu, Suwon 443-749, South Korea
e-mail: satchi@ajou.ac.kr

S. Cho
Soft Computing Laboratory, Department of Computer Science, Yonsei University, 262 Seongsanno, Seodaemun-gu 120-749, South Korea
e-mail: sbcho@cs.yonsei.ac.kr

S. Dehuri et al. (eds.), *Multi-objective Swarm Intelligence*,
Studies in Computational Intelligence 592, DOI 10.1007/978-3-662-46309-3_2

whether multi or many the objectives are conflicting in nature. Therefore, in current intelligent computational era several evolutionary [6] and swarm based meta heuristic algorithms [7] (simulating natural phenomenon) have been proposed to give a Pareto optimal solution. These Pareto optimal solutions achieve a tradeoff. They are solutions for which any improvement in one objective results in worsting of at least other objectives. A decision vector $\overrightarrow{\mathrm{dv}} \in S$, where S is decision space, is said to be Pareto optimal when there is no other $\overrightarrow{\mathrm{dv}'} \in S$, that dominates $\overrightarrow{\mathrm{dv}} \in S$. An objective vector $\overrightarrow{\mathrm{ov}} \in \overrightarrow{\mathrm{OV}}$ is said to be Pareto optimal when there is no other $\overrightarrow{\mathrm{ov}'} \in \overrightarrow{\mathrm{OV}}$, that dominates $\overrightarrow{\mathrm{ov}'}$. The Pareto optimal solution consists of a set of solutions known as a Pareto optimal set. It presents a complete set of solutions for a multi objective problem. A plot of entire Pareto sets in the design space by considering design objectives as axis it gives a Pareto front. All the Pareto optimal points are non-dominated, so they are also termed as non dominated points. In case of many objective optimization problems, the set of non dominated solutions increases, so the Pareto front becomes very hazy, hence it becomes very critical for handling such huge number of solutions. Hence, recently there is a growing interest in the community of many objective optimization researchers for designing of efficient methods to handle large number of non-dominated solutions during optimization.

On the other hand, swarm based techniques for optimization problems are leaping forward by modeling the swarming behavior of animals in the nature. Their self-adaptive, self-organized, and full of vigor with vitality attracts researchers of many disciplines [8, 9] to apply diversified domain. In this paper, we discuss on the swarm based approaches to handle the multi and many objective problems.

Now-a-days high density EEG systems are available at an affordable costs, with which large number of attributes involve and create opportunities for effective analysis. Most of the methods of analysis explicitly/implicitly follows a pattern recognition tasks. These analysis have important applications in Brain Computer Interface (BCI), epileptic seizure identification, monitoring of sleep disorder and patients in critical condition in the ICUs, etc. However, automated analysis of EEG signal in different situations is a great challenge because of the volume of dataset and dynamic nature of the signals with high temporal resolutions. Further, the automated analysis of EEG signal can be viewed as a multiple objective problems. In contrast to non-population and non-stochastic based approaches, swarm intelligence strives to achieve better compromised results by balancing both exploration and exploitation of large search space in a tolerable amount of time. This paper puts a light on the use of swarm based techniques for analyzing EEG signals which is one of the BCI agents.

1.1 Outline of Swarm Intelligence

On the set of probabilistic meta-heuristic approaches, swarm intelligence is attracting lot of attentions [6, 7]. The swarm intelligence (SI) algorithm is functioning by the collective behavior of decentralized, and self organized agents. The term swarm is

used in a general manner to refer to any restrained collection of interacting agents or individuals. The classical example of swarm is ants, birds flocks and honey bees swarming around their hive. Swarm intelligence works on two basic principles: self organization and stigmergy (e.g., Fig. 1).

i. **Self organization**: This can be characterized by three parameters like structure, multi stability and state transitions. In swarms, Bonabeau et al. [1], interpreted the self-organization through four characteristics [10]: (i) positive feedback, (ii) negative feedback, (iii) fluctuations, and (iv) multiple interactions.
ii. **Stigmergy**: It means *stimulation by work*. Stigmergy is based on three principles: (i) work as a behavioral response to the environmental state; (ii) an environment that serves as a work state memory; and (iii) work that does not depend on specific agents.

According to Millonas in 1994 [9] five principles have to be satisfied by a swarm in order to be useful in any optimization problem.

i. **The proximity principle**: The swarm should be able to do simple space and time computations, to understand and conceptualize information more quickly.
ii. **The quality principle**: The swarm should be able to respond to quality factors in the environment such as the quality of foodstuffs or safety of the location.
iii. **The principle of diverse response**: The swarm should not allocate all of its resources along excessively narrow channels and it should distribute resources into many nodes.
iv. **The principle of stability**: The swarm should not change its mode of behavior upon every fluctuation of the environment.
v. **The principle of adaptability**: The swarm must be able to change behavior mode when the investment in energy is worth the computational price.

However, it is to be noted that the above principles are applicable for single, multiple, and many-criteria optimization. The description of different swarm inspired techniques like Wasp Colony Optimization (WCO), and Termite Colony Optimization (TCO), Bacteria Foraging Optimization (BFO) can be obtained in [11–13]. Though there are several numbers of swarm based algorithm [14], are developed, but the first three algorithms enlisted in Table 1 is quite mature, so we concentrate our study on those algorithms.

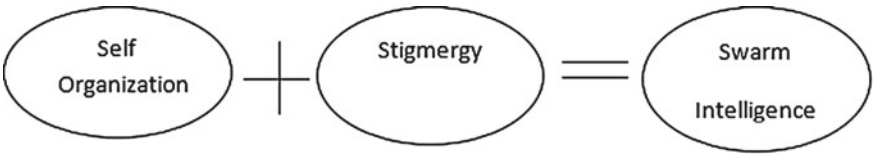


Fig. 1 Basic principles of swarm intelligence

Table 1 Swarm inspired algorithms

Name of the algorithm	Pioneer	Year of development	Motivation
ACO	M. Dorigo	1992	Ant colonies
PSO	J. Kennedy	1995	Group of birds
ABC	D. Karaboga	2005	Honey bees
WASPCO	P. Pinto	2005	Wasp
TCO	M. Roth and S.Wicker	2003	Termite
BATCO	Xin-She Yang	2010	Bat
BFO	K. M. Passino	2002	E. Coli and M. Xanthus

1.2 Multi-objective versus Many-objective Optimization

An optimization problem can be classified as a single, multi and many objective optimization problems based on the number of objectives. Again, we can classify the optimization problem in to two categories like constraint based and unconstrained. A single objective optimization problem can be stated as:

Problem with Constraint

$$\begin{aligned}&\text{Minimize/Maximize } f(\overrightarrow{x})\\&\text{Subject to,}\\&g_j(\overrightarrow{x}) \geq 0, j = 1, 2, 3, \ldots, p\\&h_k(\overrightarrow{x}) = 0, j = 1, 2, 3, \ldots, q\\&x_i^l \leq x_i \leq x_i^u; i = 1, 2, 3, \ldots, n\end{aligned} \tag{1}$$

Problem without Constraint

$$\begin{aligned}&\text{Minimize/Maximize } f(\overrightarrow{x})\\&x_i^l \leq x_i \leq x_i^u; i = 1, 2, 3, \ldots, n\end{aligned} \tag{2}$$

A solution x is a vector of n decision variables: $\overrightarrow{x} = (x_1, x_2, \ldots, x_n)$. In the case of problem with constraint, (e.g., Eq. 1) there are p inequality and q equality constraints associated and the terms $g_j(x)$ and $h_k(x)$ are called constraint functions. Although the inequality are treated as $\geq$ types, the $\leq$ constraints can also be considered in the above formulation by converting those to $\geq$ types simply by multiplying each constraint function by -1. However, there are problems, e.g., Eq. 2 which makes free from constraints but sometimes difficult to optimize, because of multi-modality, concavity, discrete, etc.

A multi-objective optimization problem can also be stated as constraint and unconstrained. (e.g., the general form is presented in Eqs. 3 and 4.

Problem with Constraint

$$\begin{aligned}&\text{Minimize/Maximize } f(\overrightarrow{\mathrm{x}}) = \{f_1(x), f_2(x), f_3(x)\}\\&\text{Subject to,}\\&g_j(\overrightarrow{\mathrm{x}}) \geq 0, j = 1, 2, 3, \ldots, p\\&h_k(\overrightarrow{\mathrm{x}}) = 0, j = 1, 2, 3, \ldots, q\\&x_i^l \leq x_i \leq x_i^u; i = 1, 2, 3, \ldots, n\end{aligned} \tag{3}$$

Problem without Constraint

$$\begin{aligned}&\text{Minimize/Maximize } f(\overrightarrow{\mathrm{x}}) = \{f_1(x), f_2(x), f_3(x)\}\\&x_i^l \leq x_i \leq x_i^u; i = 1, 2, 3, \ldots, n\end{aligned} \tag{4}$$

In multi-objective optimization, the three objective functions of $F(\overrightarrow{\mathrm{x}}) = \{f_1(x), f_2(x), f_3(x)\}$ can be either minimized or maximized or both. Like single objective optimization, the problem may have constraints or free from constraints. However, in this case, there is no single global optimal solution, rather a set of Pareto optimal solutions.

A multi-objective problem can be addressed broadly in three approaches, the first approach is weighted aggregated approach. In this case, we convert multi-objective problem into single objective problem, moreover, the most important concern is assignment of weights to the objectives, which is purely subjective. The second approach is known as a lexicographic approach. In this approach, specification of tolerance threshold and the corresponding degree of confidence for each objective is subjective and arbitrary. A third approach used is known as Pareto based approach, in this a trade-off of solutions known as non-dominated solutions are present on a front known as Pareto front. In this case, the user has his utmost autonomy to select the solution.

Most multi-objective optimization algorithms use the concept of domination to obtain a set of non-dominated solutions. The concept of domination is described in the following definitions (assuming, without loss of generality, the objective functions to be minimized).

Definition 1 Given two decisions or solution vectors $\overrightarrow{\mathrm{x}}$ and $\overrightarrow{\mathrm{y}}$, we say that decision vector $\overrightarrow{\mathrm{x}}$ weakly dominates (or simply dominates) the decision vector $\overrightarrow{\mathrm{y}}$ (denoted by $\overrightarrow{\mathrm{x}} \leq \overrightarrow{\mathrm{y}}$) if and only if $f_i(\overrightarrow{\mathrm{x}}) \leq f_i(\overrightarrow{\mathrm{y}})$ for $\forall i = 1, 2, 3$ (i.e., the solution $\overrightarrow{\mathrm{x}}$ is no worse than $\overrightarrow{\mathrm{y}}$ in all objectives) and $f_i(\overrightarrow{\mathrm{x}}) < f_i(\overrightarrow{\mathrm{y}})$ for at least one $i \in 1, 2, 3$ (i.e., the solution x is strictly better than y in at least one objective).

Definition 2 A solution $\overrightarrow{\mathrm{x}}$ strongly dominates a solution $\overrightarrow{\mathrm{y}}$ (denoted by $\overrightarrow{\mathrm{x}} < \overrightarrow{\mathrm{y}}$), if solution $\overrightarrow{\mathrm{x}}$ is strictly better than $\overrightarrow{\mathrm{y}}$ in all 3 objectives. However, if a solution $\overrightarrow{\mathrm{x}}$ strongly dominates a solution, $\overrightarrow{\mathrm{y}}$, the solution $\overrightarrow{\mathrm{x}}$ also weakly dominates solution $\overrightarrow{\mathrm{y}}$, but not vice versa.

Definition 3 The decision vector $\overrightarrow{\mathrm{x}} \in P$ (where P is the set of solutions or decision vectors) is non-dominated with respect to set P, if there does not exist another $\overrightarrow{\mathrm{x}^1} \in P$ such that $f_i^1(\overrightarrow{\mathrm{x}}) \leq f_i(\overrightarrow{\mathrm{x}})$.

Definition 4 Among a set of solution or decision vectors P, the non-dominated set of solutions or decision vectors P′ are those that are not dominated by any member of the set P.

Definition 5 A decision variable vector $\overrightarrow{\mathrm{x}} \in P$ where P is the entire feasible region or simply the search space is Pareto optimal if it is non-dominated with respect to P.

Definition 6 When the set P is the entire search space, the resulting non-dominated set P′ is called the Pareto optimal set. In other words, $P' = \{\overrightarrow{\mathrm{x}} \in P | \overrightarrow{\mathrm{x}}$ is *Pareto optimal*$\}$. The non-dominated set P′ of the entire feasible search space P is the global Pareto optimal set.

Definition 7 All Pareto optimal solutions in a search space can be joined with a curve (in two-objective space) or with a surface (more than two objective space), or with a pattern (with more than three objectives). This curve or surface is termed as a Pareto optimal front or simply Pareto front. In other words, $PF = \{f(\overrightarrow{\mathrm{x}}) | \overrightarrow{\mathrm{x}} \in P'\}$.

Now-a-days, many of the optimization problems are associated with more than three number of objectives. Any optimization problem which involves more than three number of objectives is known as many objective optimization problems. These objectives are also conflicting in nature. It states that many objectives need to be addressed simultaneously in an optimization problem.

Generally, a many-objective optimization problem is written as follows:

Problem with Constraint

$$\begin{aligned}
&\text{Minimize/Maximize } f(\overrightarrow{\mathrm{x}}) = \{f_1(x), f_2(x), f_3(x), \ldots, f_m(x)\} \\
&\text{where, } m > 3. \\
&\text{Subject to,} \\
&\quad g_j(\overrightarrow{\mathrm{x}}) \geq 0, j = 1, 2, 3, \ldots, p \\
&\quad h_k(\overrightarrow{\mathrm{x}}) = 0, j = 1, 2, 3, \ldots, q \\
&\quad x_i^l \leq x_i \leq x_i^u; i = 1, 2, 3, \ldots, n
\end{aligned} \tag{5}$$

Problem without Constraint

$$\begin{aligned}
&\text{Minimize/Maximize } f(\overrightarrow{\mathrm{x}}) = \{f_1(x), f_2(x), f_3(x), \ldots, f_m(x)\} \\
&\quad x_i^l \leq x_i \leq x_i^u; i = 1, 2, 3, \ldots, n
\end{aligned} \tag{6}$$

where $F(\overrightarrow{\mathrm{x}})$ is the m-dimensional objective vector, $f_i(\overrightarrow{\mathrm{x}})$ is the ith objective to be minimized, $\overrightarrow{\mathrm{x}}$ is the decision vector and Ω is the feasible search space. A many-objective based problem can be solved broadly in two approaches.

Preference ordering approach: This approach addresses the approximation/optimization of many objective problems by including the techniques like reducing the number of non-dominated points and assigning different ranks to non-dominated points, without avoiding user preference information.

Objective Reduction Approach: In this approach by using linear or nonlinear algorithms, the objectives are reduced from many to multi-objective. This leads to higher search efficiency and lower computational cost.

As preference ordering approach is widely popular among the research community, so our survey mostly concentrated on this approach only. This approach can be broadly classified into two categories Pareto dominance based approach and non-Pareto dominance based approach. The Pareto dominance based approach includes the techniques like:

Modification of Pareto dominance: In this approach the selection pressure is increased over the Pareto front by using the epsilon-dominance approach. The detail procedure can be obtained from [15–18].

Introduction of different ranking method: In this method different ranks are assigned to the non-dominated solutions by establishing different relations among them. In this technique, different approaches like favour relation, average ranking method, epsilon-dominance and fuzzy pareto dominance methods are widely used [19–23].

Non-Pareto dominance based approach include techniques like indicator function [24–26, 28–30, 175], scalarizing function (a kind of weighted sum approach) [31–37], and preference information [27, 38–44]. Out of the above three approaches indicator function approach is widely used. There are a number of performance indicator available to measure the quality of approximation to true Pareto front, such as hyper volume, Δp, and R2. Hypervolume [45, 176] and Δp indicators are associated with the drawback of high computational cost as the number of objective increases. R2 indicator is associated with less computational cost. Therefore, our main concern is to focus on this operator [46, 47]. R2 [48] indicator is defined as:

$$R2(ND, U) = -\frac{1}{|U|} \sum_{u \in U}^{max} \sum_{\vec{n_l} \in ND} \{u(\vec{n_l})\} \tag{7}$$

where, ND contains set of non-dominated solutions, U is set of utility functions. As utility function is defined over the weight vector, hence further we can replace U with W.

$$\begin{aligned} R2(ND, W) &= -\frac{1}{|W|}\left(\sum_{\vec{w} \in W}\left(\max_{\vec{n_l} \in ND}\left\{-\max_{i \in \{1,\ldots,m\}} w_i \left|\frac{n_i - z_i^*}{z_i^{nad} - z_i^*}\right|\right\}\right)\right) \\ &= \frac{1}{|W|}\left(\sum_{\vec{w} \in W}\left(-\max_{\vec{n_l} \in ND}\left\{-\max_{i \in \{1,\ldots,m\}} w_i \left|\frac{n_i - z_i^*}{z_i^{nad} - z_i^*}\right|\right\}\right)\right) \\ &\qquad (\because min(\vec{w}) = -max(-\vec{w})) \\ &= -\frac{1}{|W|}\left(\sum_{\vec{w} \in W}\left(\min_{\vec{n_l} \in ND}\left\{\max_{i \in \{1,\ldots,m\}} w_i \left|\frac{n_i - z_i^*}{z_i^{nad} - z_i^*}\right|\right\}\right)\right) \end{aligned} \tag{8}$$

where, z_i^* is a lower bound of all the objective functions, z_i^{nad} is an upper bound of each objective function. The size of a weight vector and reference vector is equal to number of objectives.

Now the non-dominated sorting can be done based on the utility function adopted. The main objective is to group solutions that optimize the set of chosen utility functions and place such solutions as top rank.

Such points then be removed and the 2nd rank will be formed. In this case no-Pareto dominance is used. Now, the resultant rank equation is:

$$Rank_d = U_{\vec{w} \in W} \overset{min}{\vec{n} \in ND/B_d} \left\{ \max_{i \in \{1,\ldots,m\}} w_i \left| \frac{n_i - z_i^*}{z_i^{nad} - z_i^*} \right| \right\} \tag{9}$$

where, $B_d = \{U_x Rank_x | d \geq 2, i < x < d\}$ is the union of solutions with the lowest ranks.

The computational steps which are used to measure the quality of a Pareto front is given below.

Steps:

1. Archive Pareto optimal set.
2. Define Utility function by defining weight vector.
3. Call Tchebycheff(); // It calculates Tchebycheff value
4. Compare(); // Comapair Techbycheff value of each optimal solution with weight vector.
5. Rank solutions.

A small numerical example is illustrated to understand its efficiency and efficacy.

Problem (Weapon Target Assignment): In this problem the objective is to minimize simultaneously the probability of shooting failure and the number of weapons (or resources).

Figure 2, illustrates a typical Pareto front obtained to satisfy both the objectives. Let us see the influence of R_2 indicators for ranking solutions.

The non-dominated solutions of the Pareto front are enumerated as $ND = \{(1, 25), (2, 19), (3, 17), (4, 12), (5, 9.5), (6, 8), (7, 7.45), (8, 6.35)\}$. Let take weight vector $W = \{(0.1, 0.9), (0.4, 0.6), (0.6, 0.4), (0.9, 0.1)\}$. The value of z_i^* and z_i^{nad} derived from the ND as (1, 6.35), and (8, 25) respectively. The optimum Tchebycheff value of each solution calculated as: {0.1, 0.1285, 0.2284, 0.1817, 0.1520, 0.0795, 0.0857, 0.1} (reported in Table 2).

Now the ranking of the Pareto optimal solutions is made by comparing the Tchebycheff value and the weight vector. Since, solution n_1 and n_8 has the same value, therefore, the ranks of these two solutions can be computed using Manhattan norm. For example n_1 and n_8 has the same Tchebycheff value as 0.1, so by calculating the Manhattan norm, we find that n8 has a lower value than n_1. So, n_8 is better than n_1. If we see the Tchebycheff value from the Table 2, we can assign Rank 1 to n_6, n_7 as they are near to the weight vector value.

Table 2 Tchebycheff value of Pareto optimal set

Pareto optimal set (ND)	Number of weapons (F_1)	The probability of shooting failure (F_2)	Optimum Tchebycheff value
n_1	1	25.00	0.1000
n_2	2	19.00	0.1285
n_3	3	17.00	0.2284
n_4	4	12.00	0.1817
n_5	5	09.50	0.1520
n_6	6	08.00	0.0795
n_7	7	07.45	0.0857
n_8	8	06.35	0.1000

The rest of the paper is set out as follows: Sect. 2 discusses the optimization process of swarm intelligence in multiple conflicting objectives. A frame-work for optimizing many objectives based on swarm intelligence is discussed in Sect. 3. The study of swarm intelligence in EEG signal has been made in Sect. 4. Discussion and future research direction are made in Sect. 5.

2 Swarm Intelligence for Multi Objective Problems

In this section, we discuss the effectiveness of ACO, PSO, and ABC for optimizing multi-objective problems in Sect. 2.1, 2.2, and 2.3 respectively.

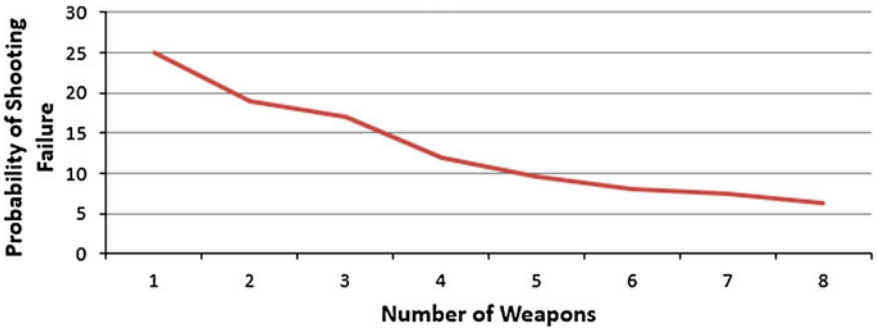


Fig. 2 Pareto front of weapon-target assignment

2.1 ACO for Multiple Objective Problems

Ant Colony Optimization [75] is a probabilistic metaheuristic technique in which a colony of artificial ants cooperates in finding better solutions to any optimization problem. Cooperation is the key element in the ACO. ACO algorithm can be used to solve both static and dynamic combinatorial optimization problems. Static problems are those problems in which the characteristics of the problems remain unchanged throughout its solution procedure. Dynamic problems in which the instance data, such as objective function values, decision parameters, or constraints, may change while solving the problems. ACO algorithm can be used to solve a multi-objective problem, in that case instead of one objective function two or three objectives functions evaluates competing criteria of solution quality. The steps to be followed while solving a problem using ACO can be enumerated as below:

Computational Steps:

1. The problem should be realized in the form of a weighted graph or sets of components and transitions on which ant can build the solutions.
2. Pheromone trails should be defined.
3. Heuristic preference for the ant should be defined while constructing the solution.
4. Choose a specific ACO algorithm and apply to the problem being solved.
5. Tune the parameter of the ACO algorithm.

The quality of the solution explored by ACO depends on the probability of proportion to the concentration of the pheromone. Where proportion of the pheromones leads to the length of the route. Concentration of the pheromones shows the number of ants used the route. The overall length of the route and the number of ants used the route influence the amount of pheromones accumulated on the route. Table 3 enumerates some of the variants of basic ACO algorithms.

Solving multiple-objective optimization problem (i.e., the number of objectives ≤ 3) using ACO is known as a Multi Objective Ant Colony Optimization (MOACO). On the other hand, solving problem with more than objectives (i.e., number of objectives >3) is popularly known as many objective optimization problems. Let us discuss some of the variants of MOACO.

In the last few years, many variants of algorithms are developed for handling the multi-objective problem by using ACO. While going through the literature it has noted that most of the multi objective algorithms are using the concept of Ant Colony System, Max-Min ant system, AQ ant system, etc. A generalized framework for MOACO is presented below:

Framework of MOACO:

1. Initialize 'n' pheromone matrices by using Initialize Pheromone trails ().
2. Make the archive of non-dominated solution empty at the beginning i.e., $A = \varphi$.
3. Execute Build solution once for each 'm' number of ants.
4. 'm' solution obtained at 't' iteration is stored in s (t).
5. Update the Archive by considering s (t) and A and executing Archive update().

Table 3 Variants of ACO algorithms

Name of the ACO variant	Authors and year	Pheromone update rule	Technique adopted	Application area
Ant colony systems (ACS)	Dorigo et al. 1991 [201]	$\tau_{ij} = (1-\varphi)\tau_{ij} + \varphi\tau_0$	1. Local pheromone update technique. 2. Diversity in the search performed by subsequent ants during iterations. 3. Pseudo random proportional rule	Travelling salesman problem (TSP)
Max-min ant systems	Stutzle [179, 180]	$\tau_{ij} = \tau_{ij} + \Delta\tau_{ij}^{best}$	1. The best ant only update the pheromone trials. 2. Value of the pheromone is bounded. Zmax and Zmin that is lower and upper bounds are decided on the basics of analytical consideration	Quadratic assignment problem
Elitist ant system	Colorni et al. 1992 [202]	$\tau_{ij}(t+1) = \rho\tau_{ij}(t) + \Delta\tau_{ij} + \Delta\tau_{ij}^{*}$	Global solution deposits pheromone on every iteration along with all other ants	Quadratic assignment problem
Ant Q	Gambardella et al. 1999 [54]	$\tau_0 = \gamma max_{j\in N_i^h}\{\tau_{ij}\}$	It involves multiple agents. These agents communicate, exchange information in the form of AQ-values	Irregation
Rank based ant system	Bullnheimer et al. 1997 [203]	$\tau_{ij}(t+1) = \rho\tau_{ij}(t) + \Delta\tau_{ij} + \Delta\tau_{ij}^{*}$	Solutions are ranked according to the density of the pheromone deposited for each solution	TSP
Best-worst ant system	Cordon et al. 2000 [204]	$\tau_{ij} = \tau_{ij} + \Delta\tau_{ij}$	It uses pheromone evaporation and best-worst pheromone updating	TSP

6. As per the new set 'A' of non-dominated solution, all the pheromone matrices are updated by considering objective values. Finally, after max iterations the archived set of non-dominated solution returned and can be found on the Pareto front.

In ACO, a MOP can be handled broadly in two different ways in ACO approach, by using multiple colonies or by using single ant colony [49, 152, 171, 173, 174, 177]. In later case, the pheromone matrix can be a single matrix for all objectives or multiple matrices. If they use multiple matrixes then an aggregation operation has to be carried out by weighted sum approach or weighted average or random approach. On the other hand if a multi-objective problem is handled using multiple ant colony then all the colonies can have a common pheromone information or according to the number of objectives each colony will have to be provided with that number of

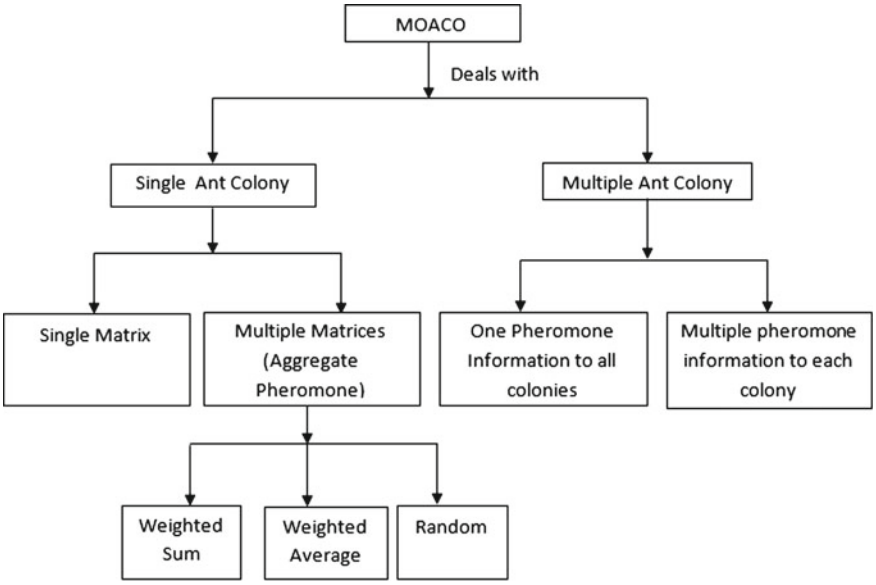


Fig. 3 Pictorial representation of different ways to handle MOP in ACO

pheromone information. A pictorial representation of handling a MOP in ACO is given in Fig. 3.

Most of the MOACO algorithms differ w.r.t. techniques adopting for pheromone trails updating, and definition of heuristic factor. Chaharsooghi and Kermani [50], have proposed a MOACO for handling multi objective resource allocation problem [181], in this algorithm they have increased the learning of ants and updated pheromone rules by simplifying probability calculations. Mariano and Morales [51, 178], have developed a MOACO by incorporating the reward mechanism in AQ ant algorithm for design of water distribution irrigation network. By, considering the pheromone updating technique Iredi et al. [52], have proposed a MOACO known as Bicriterion ANT for handling vehicle routing problem. Doerner et al. [53] proposed a Pareto based ACO algorithm for handling MOP known as P-ACO, he handled the MOP under the category of Pheromone trail matrix. Gambardella et al. [54] proposed a MOACO for handling vehicle routing problem with time window, in this they used the rank based approach with respect to the objectives, the MOACO is known as MACS-VRPTW. To handle multi-objective issues in a network MONACO is also proposed. Doerner et al. [55] proposed a COMPET ants MOACO to handle transportation problem. Gravel et al. [56], have suggested Multiple Objectives ACO Metaheuristics (MOACOM) for handling scheduling problems. Ali et al. [57], have proposed a MOACO for load balancing of distributed systems based on multiple ant colony optimization. Lopez-Ibanez and Stutzle [58, 59], have proposed a general framework by considering some MOACOs, which facilitates automatic configuration

of algorithms for handling multi-objective problems. Few frequently used MOACO algorithms are analyzed and summarized in Table 4.

While discussing several MOACO algorithms in the literature, it has found that there are some key parameter exists which can take different values at different situations. One of such parameter is heuristic matrix this can take a single or multiple value in an algorithm. If the MOACO uses an aggregation method for pheromone update then it can take the form of weighted product, weighted sum or random. The setting of the weights can be done in a dynamic manner while solving the problem or it can be taken fixed value throughout the process. Evaluation of the solution can be Pareto based or non Pareto based. The archive Pareto can be maintained offline, online, or elitist. Some algorithms also don't maintain an archive. The pheromone matrix used by the ant can be a single matrix or multiple matrices. Individual pheromone updating can be done for each ant or global updating procedure can be adopted.

2.2 PSO for Multiple Objective Problem

Particle Swarm Optimization [172, 197] is a population-based search algorithm based on the simulation of the social behavior of birds within a flock. Kennedy and Eberhart [60] originally proposed the PSO algorithm for single objective optimization. As a basic principle, in PSO, a set of randomly generated particles in the initial swarm are flown (have their parameters adjusted) through the hyper-dimensional search space (problem space) according to their previous flying experience. Changes to the position of the particles within the search space are based on the social, psychological tendency of individuals to emulate the success of other individuals. Each particle represents a potential solution, to the problem being solved. The position of a particle is determined by the solution, it currently represents. The position of each particle is changed according to its own experience and that of its neighbors. These particles propagate towards the optimal solution over a number of generations (moves) based on larger amount of information about the problem space that is assimilated and shared by all members of the swarm. The PSO algorithm finds the global best solution by simply adjusting the trajectory of each individual toward its own best location (pbest) and the best particle of the entire swarm (gbest) at each time step (generation). In this algorithm, the trajectory of each individual in the search space is adjusted by dynamically altering the velocity of each particle according to its own flying experience and the flying experience of the other particles in the search space.

The position and velocity vector of the ith particle in the d-dimensional search space can be expressed as $\overrightarrow{\mathrm{x_i}} = \langle x_{i1}, x_{i2}, \ldots, x_{id} \rangle$ and $\overrightarrow{\mathrm{v_i}} = \langle v_{i1}, v_{i2}, \ldots, v_{id} \rangle$ respectively. According to a user defined fitness function, the best position of each particle (which corresponds to the best fitness value obtained by that particle at time t) is $\overrightarrow{\mathrm{p_i}} = \langle p_{i1}, p_{i2}, \ldots, p_{id} \rangle$, denoted as *pbest* and the fittest particle found so far in the entire swarm is $\overrightarrow{\mathrm{p_g}} = \langle p_{g1}, p_{g2}, \ldots, p_{gd} \rangle$, denoted as *gbest*. Then the new velocities and the new positions of the particles for the next fitness evaluation are calculated at time $t + 1$ using the following two self-updating equations:

Table 4 Summary of MOACO algorithms

Algorithm	Application area	Type of colony	Pheromone update technique	Aggregation technique	Number of solutions	MOO approach used	Archive
MORAP	Resource allocation	Single	ND	–	Single	Pareto	No
MOAQ	Network	Multiple	ND	–	Multiple	Pareto	No
Bicriterian ANT	Vehicle routing	Multiple	ND	Weighted product	Multiple	Pareto	No
P-ACO	Portfolio selection problem	Single	BO	Weighted product	Multiple	Pareto	Yes
MACS-VRPTW	Vehicle routing	Multiple	ND	Weighted product	Multiple	Pareto	Yes
MOACS	Vehicle routing	Multiple	ND	–	Multiple	Pareto	Yes
MONACO	Networking	Single	BOW	Weighted product	Single	No	No
COMPET ants	Transportation	Multiple	BO	Weighted sum	Multiple	Yes	No
MOACOM	Scheduling	Multiple	BO	Weighted product	Single	No	No
ACOAMO	Scheduling	Multiple	ND	Weighted product	Single	No	No
MOACO	Load balancing	Multiple	BO	Random	Single	Yes	Yes

$$v_{id}(t+1) = wv_{id}(t) + c_1 rand_1()(p_{id}(t) - x_{id}(t) + c_2 rand_2()(p_{gd}(t) - x_{id}(t)) \tag{10}$$

and,

$$x_{id}(t+1) = x_{id}(t) + v_{id}(t) \tag{11}$$

The pseudo-code for a basic PSO algorithm for single objective optimization is illustrated below:

```
Begin
Parameter settings and initialization of swarm.
Evaluate fitness and locate the leader (i.e., initialize p_best and g_best).
I=0          // I = Iteration count
While (the stopping criterion is not met, say, I < Imax)
    Do
        For each particle
            Update velocity (flight) and position as per Eqs. 10 and 11.
            Evaluate fitness
            Update p_best
        End For
    Update leader (i.e., g_best)
    I++
End While
End
```

First, the swarm is initialized. This initialization includes both positions and velocities. The corresponding p_{best} of each particle is initialized and the leader is located (the g_{best} solution is selected as the leader). Then, for a maximum number of iterations, each particle flies through the search space updating its position (using equations) and its p_{best} and, finally, the leader is updated too. Based on the nature of the problem (Discrete/Continious), influence of parameters (towards exploration and exploitation), number of swarms, and combination with other heuristic methods, the PSO algorithms are classified as follows:

Discrete/Binary *PSO*: Kennedy and Eberhart [60, 154] immediately after proposing the framework for particle swarm optimization have given an alternative algorithm to operate on discrete binary variables. In this binary version, the trajectories are changes in the probability that a coordinate will take on a zero or one value. The binary PSO can handle the discrete optimization problems. There are different types of Binary PSO available in the literature like PSOLS, DPSO, CPSO, PBPSO (Table 5).

Adaptive PSO: Shi and Eberhart [61] after doing an empirical study over PSO, by using four different benchmark functions suggested a self-adapting strategy for adjusting the inertia weight to overcome the disadvantage that PSO may fail to find the required optima in cases when the problem to be solved is too complicated and complex. There are different types of adoptive PSO's like APSO, MPSO, AsCFPSO, APSO, ACPSO, etc. are reported in the literature (Table 6).

Table 5 List of discrete/binary PSO

Authors	Year	Algorithm	Application area
Lio et al. [182]	2005	Particle swarm optimization using local search (PSO-LS)	Flow shop scheduling
Correa et al. [183]	2006	Discrete PSO (DPSO)	Attribute selection
Wang [184]	2007	Discrete PSO (DPSO)	Knapsack problem
Jarboui et al. [185]	2007	Combinatorial PSO	Resource constraint project scheduling
Zhen et al. [186]	2008	Probability based binary PSO (PBPSO)	WFGD problem

Table 6 List of adoptive PSO

Authors	Year	Algorithm	Technique
Carlsie and Dozier [187]	2000	APSO	Each particle to reset its record of its best position as the environment changes
Zhang and Liu [188]	2005	APSO	Adjusts the parameters automatically, based on the fitness values of particles during the optimization process.
Zhen et al. [189]	2007	Modified PSO (MPSO)	Present a new adaptive mutation particle swarm optimizer, which is based on the variance of the population's fitness
Chunxia and Youhong [190]	2008	Adaptive simple PSO with constriction factor (AsCFPSO)	Used the concept of chaotic map
Zhan et al. [191]	2009	APSO	Used the concept of elitist learning
Hongwu [192]	2009	Adaptive chaotic PSO (ACPSO)	Applies a short term chaotic search to the best particle in the iteration

Multi-swarm PSO: Bergh and Engelbrecht [62], have proposed a variation on the traditional PSO algorithm, called the cooperative particle swarm optimizer, or CPSO, employing cooperative behavior, to significantly improve the performance of the original algorithm. This is achieved by using multiple swarms to optimize the different components of the solution vector cooperatively. Different Multi-swarm PSO are CPSO, Dual PSO, and MSBPSO (Table 7).

Hybrid PSO: This method combines different techniques to make the PSO algorithm more smatter. Some the hybrid PSO algorithms are: DPSO, SCEPSO, PSOwSA, OPSO, NF-PSO, ESCA (Table 8).

Table 7 List of multi-swarm PSO

Authors	Year	Algorithm	Technique
Bergh and Engelbrecht [62]	2004	Cooperative particle swarm optimizer (CPSO)	Potter's technique
Jian et al. [193]	2008	Dual PSO	The stochastic ranking algorithm is employed
Li and Xiao [194]	2008	Multi-swarm and multi-best particle swarm optimization (MSBPSO)	Aggregation

Table 8 List of hybrid PSO

Authors	Year	Algorithm	Basic components	Application area
Ling et al. [195]	2005	DPSO	DPSO + SA	Vehicle routing problem
Pan et al. [196]	2006	SCEPSO & PSOwSA	EA + PSO PSO + SA	Bench mark functions
Tian and Li [198]	2009	NFPSO	FLC + PSO	System performance
Lung and Dumitrescu [199]	2009	ESCA	EA + PSO	Moving peaks benchmark (MPB)
Mousa et al. [200]	2012	Hybrid PSO	LS + GA + PSO	Bench mark functions

The relative simplicity of PSO and the population-based technique as well as the information sharing mechanisms associated with PSO, makes it a natural candidate for solving multiple and many objective problems by redefining the notion of the guide. Let us discuss PSO for multi-objective optimization problem.

The two basic PSO equations restrict additional heuristics related to the real-world problem to be incorporated into the algorithm. Thus, PSO in its basic form will not perform well in searching complex multi-objective solution spaces, which are the case for many complex real world scenarios. Changing a PSO to a MOPSO requires a redefinition of guide, in order to obtain a front of optimal solutions (Pareto front). In MOPSO, the Pareto optimal solutions are used to determine the guide for each particle. In order to apply the PSO strategy for solving multi-objective optimization problems, the original scheme has to be modified. The algorithm needs to search a set of different solutions (the so-called Pareto front) instead of a single solution (as in single objective optimization). We need to apply (MOPSO) to search towards the true Pareto front (non-dominated solutions). Unlike the single objective particle swarm optimization, the algorithm must have a solution pool to store non-dominated solutions found by searching upto stopping criterion (say, upto iteration *Imax*). Any of the solutions in the pool can be used as the *gbest* particle to guide other particles in the swarm during the iterated process. The plot of the objective functions whose non-dominated solutions are in the solution pool would make up for the Pareto front. The basic MOPSO algorithm can be stated as below:

```
Begin
Parameter settings and initialize swarm
Evaluate fitness and initialize leaders in a leader pool or external archive
Archive the top best leader from the external archive through evaluation of some sort
of quality measure for all leaders.
I = 0          // I = Iteration count
While (the stopping criterion is not met, say, I < Imax)
    Do
        For each particle
            Select leader in the external archive
            Update velocity
            Update position
            Mutate periodically /*optional */
            Evaluate fitness
            Update pbest
        End For
    Crowding of the leaders
    Update the top best into the external archive
    I = I + 1.
End While
Report results in the external archive
End
```

In the above general MOPSO algorithm, first the swarm is initialized. Then, a set of leaders is also initialized with the non-dominated particles from the swarm. This set of leaders is stored in an external archive. Later on, some sort of quality measure is calculated for all the leaders in order to select, usually one leader for each particle of the swarm. At each generation, for each particle, a leader is selected and the flight is performed. Most of the existing MOPSOs apply some sort of mutation operator after performing the flight. Then, the particle is evaluated and its corresponding p_{best} is updated. A new particle replaces its pbest particle usually when this particle is dominated or if both are incomparable (i.e., they are both non-dominated with respect to each other). After all the particles have been updated, the set of leaders is updated, too. Finally, the quality measure of the set of leaders is re-calculated. This process is repeated for a certain fixed number of iterations.

Coello and Lechuga [63] extended PSO to deal with multi-objective optimization problems using the similar approach of Pareto dominance to determine the flight direction of a particle and their MOPSO algorithm maintains previously found non-dominated vectors in a global repository (secondary memory) that is later used by other particles to guide their own flight. Their approach is population based as well as geographically based to maintain diversity. Fieldsend and Singh [64] utilize the dominated tree data structure to enable the selection of an appropriate Pareto archive member to act as the global best for any given particle and also maintains a local set of best solutions for each swarm member. They have demonstrated that this approach is significantly better than the method used by Coello and Lechuga [63] and also

PAES derived from the unified model proposed by Laumanns et al. [65]. They have demonstrated that by including a stochastic turbulence variable within MOPSO, its performance has been significantly increased. Even though Coello and Lechuga [63] maintain an archive of global best solutions, Fieldsend and Singh [64] pointed out that there is a better way to select from this archive than by simple density based selection. Thus, they have included a new data structure called dominated tree, as this data structure facilitates rapid selection of an appropriate archive member for their new MOPSO method. Hu and Eberhart [66] attempt to optimize MOPSO having two objectives through the a priori knowledge of the test function properties. Parsopoulos and Vrahatis [67] introduced two methods to optimize MOP having two objectives. One uses a weighted aggregate approach and another is loosely based on Schaffer's MOEA [68] i.e. vector evaluated PSO (VEPSO) method. The second method—the vector evaluated particle swarm optimizer (VEPSO) of Parsopoulos and Vrahatis [67] uses one swarm for each objective. According to them, the best particle of the second swarm is used to determine the velocities of the first swarm (act as its global best), and vice versa. Mostaghim and Teich [69] introduce a new method called sigma method for finding best local guides for each particle of the population from a set of Pareto optimal solutions. According to them, such a technique has a great impact on the convergence and diversity of solutions, especially when optimizing problems with a high number of objectives. The sigma method which uses clustering techniques for fixing the archive size has a better computational time, diversity and covergence than the dominated tree method of Fieldsend and Singh [64] and strength pareto evolutionary algorithm (SPEA). Coello et al. [70] proposed an improved version of the MOPSO algorithm in which they have added a constraint handling mechanism and a mutation operator that considerably improves the exploratory capabilities of their original algorithm. Their MOPSO is validated using several standard test functions reported in the specialized literature. They have compared this improved MOPSO algorithm against three highly competitive evolutionary multi-objective (EMO) algorithms, NSGA-II and PAES. Fieldsend [71] compare a number of selection regimes for the choosing of global best (*gbest*) and personal best (*pbest*) for swarm members in MOPSO. He has shown two distinct *gbest* selection techniques, one that does not restrict the selection of archive members and the other with distance based *gbest* selection techniques. According to him, these two methods promote two types of search. He has also described the potential problem of particle clumping in MOPSO. Ray and Liew [72] propose an algorithm which uses Pareto dominance and combines concepts of evolutionary techniques with the particle swarm. Pulido and Coello [73] use the concept of Pareto dominance to determine the flight direction of a particle. The authors adopt clustering techniques to divide the population of particles into several swarms. This aims to provide a better distribution of solutions in decision variable space. This approach does not use an external archive since elitism in this case is an emergent process derived from the migration of leaders.

Mostaghim and Teich [74] propose particle swarm inspired evolutionary algorithm (PSEA) which is a hybrid between PSO and an evolutionary algorithm. The main aim is to use EA operator (mutation, for example) to emulate the workings of PSO mechanisms. Different methods for selecting and deleting particles (leaders) from the

archive are analyzed to generate a satisfactory approximation of the Pareto front. The authors provide some statistical analysis in order to access the impact of each of the parameters used by their approach. Li [76] proposes an approach which incorporates the main mechanisms of the NSGA-II of Deb et al. [77] to the PSO algorithm. In this approach, once a particle has updated its position, instead of comparing the new position only against the *pbest* position of the particle, all the *pbest* positions of the swarm and all the new positions recently obtained are combined in just one set (given a total of 2N solutions, where N is the size of the swarm). Then, the approach selects the best solutions among them to conform the next swarm (by means of a non-dominated sorting). The author doesn't specify which values are assigned to the velocity of pbest positions, in order to consider them as particles. This approach also selects the leaders randomly from the leaders set (stored in an external archive) among the best of them, based on two different mechanisms: a niche count and a nearest neighbor density estimator. This approach uses a mutation operator that is applied at each iteration step only to the particle with the smallest density estimator value (or the largest niche count). Sierra and Coello [78] propose an approach which is based on Pareto dominance and the use of a nearest neighbor density estimator for the selection of leaders (by means of a binary tournament). This proposal uses two external archives: one for storing the leaders currently used for performing the flight and another for storing the final solutions.

Ho et al. [79] propose a novel formula for updating velocity and position of particles, based on three main modifications to the known flight formula. The authors introduce a craziness operator in order to promote diversity within the swarm. The craziness operator is applied (with certain probability) to the velocity vector before updating the position of a particle. Finally, the authors introduce one external archive for each particle and one global external archive for the whole swarm. The archive of each particle stores the latest Pareto solutions found by the particle and the global archive stores the current Pareto optimal set. Every time a particle updates its position, it selects its personal best from its own archive and the global best from the global archive. In both cases, the authors use a roulette selection mechanism based on the fitness values of the particles and on an age variable that the authors introduce and that is increased at each generation. Villalobos-Anias et al. [80] propose a new mechanism to promote diversity in multi-objective optimization problems. Although the approach is independent of the search engine adopted, they incorporate it into the MOPSO proposed in Coello et al. [70]. Salazar-Lechuga and Rowe [81] propose an approach whose main idea is to use PSO to guide the search with the help of niche counts to spread the particles along the Pareto front. The approach uses an external archive to store the best particles (non- dominated particles) found by the algorithm. Since this external archive helps to guide the search, the niche count is calculated for each of the particles in the archive and the leaders are chosen from this set by means of a stochastic sampling method (roulette wheel). Also, the niche count is used as a criterion to update the external archive. Each time the archive is full and a new particle wants to set in, its niche count is compared with the niche count of the worst solution of the archive. If the new particle is better than the worst particle, then the new particle enters into the archive and the worst particle is deleted. Niche counts are

updated when inserting or deleting a particle from the archive. Janson and Merkle [82] proposed a hybrid particle swarm optimization algorithm for multi objective optimization, called ClustMPSO. ClustMPSO combines the PSO algorithm with clustering techniques to divide all particles into several sub-swarms. For this aim, the authors use the K-means algorithm. Each sub-swarm has its own non-dominated front and the total non-dominated front is obtained from the union of the front of all the sub-swarms. Lewis [83] has proposed a novel MOPSO algorithm called LoCost algorithm through some modification in the velocity updating equation of the conventional PSO algorithm based on an extension of the concepts of spatial social networks using a model of the behavior of particular types of swarms known as crickets and locusts. He observes that the proposed algorithm has performed quite comparably to a conventional MOPSO algorithm in terms of convergence, and has achieved appreciably greater coverage of the approximation to the Pareto-front. Leong and Yen [84] present the improvement of two design components (swarm growing strategy and objective space compression and expansion strategies) from the existing multiple swarm MOPSO, namely dynamic swarm in multi-objective particle swarm optimization (DSMOPSO). The multiple-swarm concept has been incorporated into PSO to yield more efficient and effective designs, especially in enhancing the population diversity, and to counter PSO's tendency in undesirable premature convergence. Lewis and Ireland [85] have proposed an approach of hybridizing a multi-objective optimization method and subsequent single-objective search has been proposed as a means to automate the process of solution selection from the set of Pareto-optimal solutions typically delivered.

Cagnina et al. [86] have proposed a hybrid particle swarm approach called simple multi-objective particle swarm optimizer (SMOPSO) which incorporates Pareto dominance, an elitist policy, and two techniques to maintain diversity: a mutation operator and a grid which is used as a geographical location over objective function space. Laura and Mihai [87] have proposed a hybrid technique that combines a genetic algorithm (GA) and a PSO algorithm. Each GA chromosome is an array encoding a meaning for updating the particles of the PSO algorithm. The evolved PSO algorithm is compared to a human-designed PSO algorithm by using ten artificially constructed functions and one real-world problem. The model proposed in this paper is divided into two levels: a macro level and a micro level. The macro level is a GA algorithm that evolves the structure of a PSO algorithm. For this purpose, a particular function is used as a training problem. The micro level is a PSO algorithm used for computing the quality of a GA chromosome from the macro level. The array of integers encoded into a GA chromosome represents the order of update for particles used by a PSO algorithm that solves a particular problem. Goldberg et al. [88] present a particle swarm optimization algorithm for the multicriteria constrained minimum spanning tree problem. The operators for the particle's velocity are based upon local search and path-relinking approaches. In path-relinking approach, a velocity operator is developed and utilized when a particle goes toward the position of another particle. For the iterations where a particle follows its own way, a local search procedure is used. Ho et al. [89] proposes a novel intelligent multi-objective particle swarm optimization (IMOPSO) to solve multi-objective optimization problems.

Koppen and Veenhuis [90] introduce a new approach to multi-objective particle swarm optimization. The approach is based on the recently proposed Fuzzy-Pareto-Dominance (FPD) relation. Chiu et al. [91] present a local guide assignment strategy for MOPSO called cross searching strategy (CSS) which will distribute suitable local guides for particles to lead them toward the Pareto front and also keeping a diversity of solutions. A disturbance operation is also introduced to enhance the particle's searching ability to avoid local search. Peng and Zhang [92] have studied the application of PSO techniques to multiobjective optimization using decomposition methods. A new decomposition-based multi-objective PSO algorithm is proposed, called MOPSOID. It integrates PSO into a multi-obejective evolutionary algorithm based on decomposition (MOEAID). Like MOEAlD, each particle in MOPSOID carries one unique weight vector. Therefore, each particle has an unique search direction defined by its weight vector. Padhye et al. [93], have reviewed the several proposals for guide selection in multiobjective particle swarm optimization (MOPSO) and compare them with each other in terms of convergence, diversity and computational times. The new proposals made for guide selection, both *pbest* and *gbest*, are found to be extremely effective and perform well compared to the already existing methods. The combination of various selection methods is also studied and it turns out that there exist certain combinations which yield an overall superior performance outperforming the others. Cabrera and Coello [94] present a multi-objective particle swarm optimizer (MOPSO) which is characterized for using a very small population size so that it requires a very low number of objective function evaluations (only 3000 per run) to produce reasonably good approximations of the Pareto front of problems of moderate dimensionality.

The proposed approach first selects the leader and then selects the neighborhood for integrating the swarm. The leader selection scheme adopted is based on Pareto dominance and uses a neighbors' density estimator. Additionally, the proposed approach performs a re-initialization process for preserving diversity and uses two external archives: one for storing the solutions that the algorithm finds during the search process and another for storing the final solutions obtained. Furthermore, a mutation operator is incorporated to improve the exploratory capabilities of the algorithm. Wang and Yang [95] extend the NSGA-II-MOPSO algorithm, which is based on the combination of NSGA-II and multi-objective particle swarm optimizer (MOPSO) for unconstrained multi-objective optimization problems, to accommodate constraints and mixed variables. In order to utilize the valuable information from the objective function values of infeasible solutions, a method called M+1 non-dominated sorting is proposed to check the non-domination levels of all infeasible solutions. Integer and discrete variables are dealt with using a method called stochastic approximation.

Goh et al. [96] propose a competitive and cooperative co-evolutionary approach to be adapted for multi-objective particle swarm optimization algorithm design. It appears to have considerable potential for solving complex optimization problems by explicitly modeling the co-evolution of competing and cooperating species. The competitive and cooperative co-evolution model helps to produce the reasonable problem decompositions by exploiting any correlation and interdependency among the

components of the problem. Each sub-swarm is assigned a probability of representing a particular variable and only two sub-swarms, the current sub-swarm and competing sub-swarm compete for the right to represent any variable at any one time. Tsai et al. [97] propose an improved multi-objective particle swarm optimizer with proportional distribution and jump improved operation, named PDJI-MOPSO, for dealing with multiobjective problems. PDJI-MOPSO maintains diversity of new found non dominated solutions via proportional distribution, and combines the advantages of wide ranged exploration and extensive exploitation of PSO in the external repository with the jump improved operation to enhance the solution searching abilities of particles.

Zheng and Liu [98] propose a hybrid vertical mutation and self-adaptation based MOPSO (VMAPSO) to overcome the disadvantages of existing MOPSOs. Wang and Yang [99] use a new optimality criterion based on preference order (PO) scheme to identify the best compromise in multi-objective particle swarm optimization (MOPSO). Preference order is a generalization of Pareto optimality. It provides a way to designate some Pareto solutions superior to others when the size of the non-dominated solutions set is very large. To find the "best compromise", the non-dominated solutions are ranked according to PO. The ranking procedure can be summarized in three steps: (i) identify the combinations of all subsets to m objectives; (ii) assign the order to all non-dominated solutions for each combination of all subsets based on PO; and (iii) identify the "best compromise" in all non-dominated solutions according to their order. The proposed algorithm is quite effective in maintaining the diversity of the solutions.

2.3 ABC for Multiple Objective Problem

ABC is one of the swarm based meta-heuristic algorithm introduced by Karaboga in 2005. It is based on the model proposed by Tereshko and Loengarov [100], Mishra et al. [14]. It is motivated by the intelligent behavior of honey bees. Honey bees are one of the interesting swarm in nature. They have the skills like photographic memories, space-age sensory, and navigation systems. Honey bees are social insects who live in colonies. There are three kinds of bees in a colony: queen, drones, and workers. Queen bee has the largest living years. She is the only egg laying female who is the mother of all the members of the colony. When the colony is lack of food sources, the queen produces new eggs. If the colony becomes too crowded, the queen stops laying. Drones are the father of the colony. They are produced from the unfertilized eggs; their life span is about six months. The main task of the drones is to fertilize with the queen. Workers have the task of collecting food, storing it, removing dead bees, ventilate the hive and give protection to the hive. The task of a worker bee is based on its age and the needs of the colony (Table 9).

At the initial stage the algorithm was used for numerical optimization, but later on it is widely used for combinatorial optimization and also for constraint and unconstrained function optimization problems. In the ABC location of food source represents a possible solution to the problem and the nectar amount of a food source

Table 9 Application of MOPSO on different discipline

Author	Number of objectives	Problem solved
Chauhan et al. [101]	2	Power system
Falcon et al. [102]	2	Clustering
de Carvalho et al. [103]	3	Fault prediction
Martin et al. [104]	2	Ultra wide band
Pang and Chen [105]	2	Signal processing
Hazra and Sinha [106]	2	Congestion management in power system
Qasem and Shamsuddin [107]	2	Medical
Pindoriya et al. [108]	2	Portfolio management
Sha and Lin [109]	2	Job scheduling
Wang and Singh [110]	2	Power sector
Montalvo et al. [111]	2	Water distribution
Liu [112]	2	Calibaration of rain fall
Zhang and Liu [113]	2	Power system
Cai et al. [114]	2	Environment & Economics
Ganguly et al. [115]	2	Electrical distribution system
Sankaran and Manne [116]	2	Channel equalization

corresponds to the quality of the solution. A general framework of ABC is presented below.

Framework of ABC:

Initialize the population of solution x_{ij}.
Evaluate the population
Cycle = 1
Repeat

Produce new solutions (food source positions) v_{ij}, in the neighborhood of x_{ij} for the employed bees $v_{ij} = x_{ij} + \varphi_{ij}(x_{ij} - x_{kj})$ and evaluate
Apply the greedy selection process between x_i and v_i
Calculate probability values P_i for the solutions x_i by means of their fitness value as

$$P_i = \frac{fit_i}{\sum_{i=1}^{SN} fit_i}$$

Calculate the fitness value of solutions

$$fit_i = \begin{cases} \frac{1}{1+f_i} & \text{if } f_i \geq 0 \\ 1 + abs(f_i) & \text{if } f_i < 0 \end{cases}$$

Table 10 Control parameter in ABC

Parameter	Meaning
Population size	Number of solutions
Termination condition	Maximum number of cycles/maximum CPU time/if the onlooker bees converge faster
Number of employed bees	Number of food sources in the hive

Normalize P_i values into [0, 1]
Produce the new solutions v_i for the onlookers from the solutions x_i selected depending on P_i and evaluate them
Apply the greedy selection process for the onlookers between x_i and v_i
Determine the abandoned solution, if exists and replace it with a new randomly produced solution x_i for the scout using the equation

$$x_{ij} = min_j + rand(0, 1) * (max_j - min_j)$$

Memorize the best food source position (solution) achieved so far
Cycle = cycle + 1
Until cycle = Maximum Cycle Number (MCN)

The ABC algorithm is simple, easy to implement, has few control parameters and mostly robust in nature so, it is widely used for a single objective optimization problem. Due to the aforesaid qualities of ABC, like other multi-objective heuristic algorithms, it can be extended to multi and many objective domains [14, 129]. Let us discuss some of the variants of MOABC (Tables 10 and 11).

In the last few years ABC is being used widely for solving multi criterion problems. It has been the point of attraction for the researchers due to its characteristics like use of less number of control parameter and its simplicity. The basic MOABC algorithm can be stated as below:

The basic MOABC algorithm can be stated as below:

1. Cycle = 1
2. Initialize the food source positions (solutions) $x_i, i = 1, \ldots, SN$
3. Evaluate the nectar amount (fitness fit_i) of food sources
4. The initialized solutions are sorted based on nondomination
5. Store nondominated solutions in the external archive (EA)
6. Repeat
7. Onlooker Bees' Phase
    For each onlooker bee
        Randomly chooses a solution from EA
        Produce new solution vi by using comprehensive learning strategy
        Calculate the value fiti

Table 11 Popular ABC algorithms

Author	Year	Application area
Karaboga [117]	2005	Numerical optimization
Teodorovic and Dell [118]	2005	Ride matching problem/transportation problem
Wedde and Farooq [119]	2005	Mobile adhoc N/W
Drias and Yahi [120]	2005	MAX-W-SAT problem
Chong et al. [121]	2006	Job shop scheduling
Quijano and Passino [122]	2007	Resource allocation problem
Karaboga and Akay [123]	2009	Engineering design problem
Karaboga and Ozturk [124]	2009	Neural network training
Karaboga [125]	2009	Image processing
Karaboga and Ozturk [126]	2011	Data clustering
Xu and Duan [127]	2010	Image processing
Yu et al. [128]	2013	Test functions

Apply greedy selection mechanism to decide which solution enters EA
End For
8. The solutions in the EA are sorted based on nondomination
9. Keep the nondomination solutions of them staying in the EA
10. If the number of nondominated solutions exceeds the allocated size of the EA, then use crowding distance to remove the crowded members
11. Cycle = cycle + 1.
12. Until (cycle = Maximum Cycle Number)

In 2011 Zou et al. [130] proposed a Multi-objective Artificial Bee Colony (MOABC) which allows the ABC algorithm to deal with multi-objective optimization problems. It is based on non-dominated sorting strategy [131] and used the concept of Pareto dominance [132, 133] to determine which solution vector is better. In addition, it uses an external archive to maintain non-dominated solution vectors. Akbari et al. [133] proposed a MOABC (encouraged from their earlier work [133, 134]), which utilizes different types of bees (i.e., employed bees, onlookers, and scouts) and a fixed-sized archive to maintain the good solutions. To maintain the archive they have used an ε-dominance method. The MOABC method is constituted of five parts: Initialization, Send Employed Bees, Send Onlooker Bees, Send Scout Bees and Update the archive. They tested MOABC over unconstrained and constraint based test functions and concluded that MOABC successfully solves the functions and obtains first rank among all other optimization algorithms (Table 12).

Akbari et al. [133] concluded that the MOABC can obtain better performance by increasing the number of individuals in the population and the number of iterations. The effectiveness of the MOABC depends on two factors: a population of different bee types with an efficient decision making process, and a grid for controlling the diversity over the external archive. In 2011, Omkar et al. [129] employed the concept of Vector Evaluated Artificial Bee Colony (VEABC), a variant of the classic ABC for multi-objective design optimization of composite structure, which proves to be very

Table 12 Enlist of MOABC algorithms

Name of MOABC	Authors and year	Technique	Application domain	Number of objectives
MOABC	Zou et al. 2011	Pareto based	Test functions	2
MOABC	Akbari et al. 2012	Pareto based, grid approach	Test functions	2
VEABC	Omkar et al. 2011	Pareto based	Composite structure	2
EPABC	Wang et al. 2012	Pareto based	Job shop scheduling	3
BCMO	Xinyi et al. 2012	Pareto based	Test functions	2
A-MOABC/PD	Bahriye 2013	Non Pareto	Test function	2
A-MOABC/NS	Bahriye 2013	Non Pareto	Test function	2
S-MOABC/NS	Bahriye 2013	Non Pareto	Test function	2
ICABCMOA	Zhou et al. 2013	Pareto based	Test function	2

appropriate for structural problems. In 2012, Xinyl et al. [135] proposed an artificial Bee Colony algorithm for Multi-objective Optimization problems (BCMO) by introducing the concept of Pareto sorting operation and Pareto dominance approach. By using the effective decoding scheme, hybrid initialization strategy and by using the exploration and exploitation ability of ABC Zhou et al. [136] proposed an MOABC for handling Multi-objective job shop scheduling problem. Wang et al. [137] proposed an enhanced Pareto based artificial bee colony algorithm EPABC for solving multi-objective flexible job shop scheduling problem. Based upon synchronous and asynchronous models using Pareto dominance and non dominated sorting Akay [138] proposed three MOABC named as asynchronous multi-objective ABC using only Pareto dominance rule (A-MOABC/PD), asynchronous multi-objective ABC using non-dominated sorting procedure (A-MOABC/NS) and synchronous multi-objective ABC using non- dominated sorting procedure (S-MOABC/NS). By considering several parameters like inverted general distance, spread performance metrics and running time, etc. and came to a conclusion that S-MOABC/NS is more scalable and efficient in comparison to other two algorithms. To increase the search efficiency in ABC Zhou et al. [139] proposed Immune based Chaotic Artificial Bee Colony Multi-objective Optimization Algorithm (ICABCMOA). In this approach in order to meet the requirements of Pareto based approaches they defines a new fitness based function based on the dominated number. They have used a high dimension chaotic method based on tent map to increase the search efficiency.

From the above discussion, it is derived that the MOABC with properties such as an effective trajectory adjusting strategy, and an efficient way for maintaining the diversity over the Pareto front, can be used as an alternate way for optimizing multiple and many objective problems.

3 Swarm Intelligence for Many Objective Optimization

Recall that population based swarm intelligence techniques are one of the active research areas in the field of multi-objective problems. However, their search ability severely decreases when it is mapped to a many objective problem. For example, the swarm intelligence techniques like ACO, PSO, and ABC, which uses a Pareto dominance approach faces a number of difficulties like:

i. Decrease in convergence property: As the number of objective increases most of the solutions in the population become non-dominated, this decreases the Pareto dominance based selection pressure towards the Pareto front.
ii. As the number of objective increases, thousands of non-dominated solutions are generated, a subset of the nondominated solutions has to be selected to be approximated to entire Pareto front, which is a tedious task.
iii. In a multi-objective problem few number of non-dominated solutions appear on the Pareto front, so the user/the decision maker can choose a solution by visualizing the results. But in case of a many objective as the number of non-dominated solutions are more so it becomes difficult for the user/decision maker to visualize the results and to take a decision.

The problems raised by the swarm intelligent algorithms in a many objective field can be handled by using any of the following techniques.

i. By changing the standard Pareto dominance Method: Rather than using the common Pareto dominance method, the angle of dominating region should be adjusted with the number of objectives, i.e., increase in number of objectives requires a large angle of dominated region. So, that number of non-dominated solutions in each population are decreased and the selection pressure towards the Pareto front can be strengthened.
ii. By ranking the non-dominated solutions: By using favour relation proposed by Drechsler et al. [140] the non-dominated solutions can be ranked. The relation is based on a number of objectives.
iii. By using the indicator function [141]: A number of performance indicators have been proposed to measure the quality of the non-dominated solution sets. They can be applied over to find out the solutions. For example one of the performance indicator used is the hyper volume indicator. Hyper volume indicator measures convergence to the Pareto front and diversity of the obtained fronts. The hyper volume computes the volume in the objective function space covered by members of a non dominated set of solution ND. For each non-dominated solution of Q a hyper cube v_i is constructed, with a reference point 'w' and the solution 'i' as the diagonal corners of the hypercube. The reference point 'w' can be found by constructing a vector with the worst objective function values. After that the union of all hyper cubes are found and then its hyper volume 'HV' is calculated as:

$$\text{HV} = \text{Volume}(U_{i=1}^{n} v_i) \tag{12}$$

Higher values of the hyper volume performance measure imply more desirable solution.

iv. Use of scalarizing function [142]: In this technique weighted sum of multiple objectives are calculated even though the number of objectives is large. There are many scalarizing functions available in the literature like weighted sum, reference vector etc. to reduce number of objectives. For example: Let us consider F(z) is a function which is to be maximized number of objectives are 'n' number. F(z) is the 'n' dimensional objective vector, & Z is the decision vector. So,

$$Maximize F(z) = \{F_1(Z), F_2(Z), \ldots, F_n(Z)\}$$

By applying frequently used weighted sum scalarizing function $w = \{w_1, w_2, \ldots, w_n\}$

$$fitness(z) = w_1 * F_1(z) + w_2 * F_2(z) + \cdots + w_n * F_n(z)$$

This is how the weighted sum approach works. The weight vector is a user input or decision makers input. Distance from the reference vector can be used as a scalarizing function. When a reference vector $RV = \{RV_1, RV_2, \ldots, RV_n\}$ is given as a desired point in the objective space, the distance from RV can be calculated as a scalaring function.

$$fitness(z) = distance(RV, F(z))$$

v. Use of preference Information: SI algorithms generally designed to search for a set of non-dominated solutions that approximates the Pareto front. As the number of solutions for a good approximation exponentially increases with the number of objectives, one can focus on a specific region of the Pareto front using decision maker's preference.

General Framework for solving a Many-objective Problem using SI

Step 1 Parameter setting of MOACO/MOPSO/MOABC.
Step 2 Generate non-dominated solution by MOACO/MOPSO/MOABC.
Step 3 Collect preference/weight information from the decision maker.
Step 4 Rank non-dominated solution using indicator based approach R2 analysis/TOPSIS
Step 5 Get the highest rank solution as the selected one.
Step 6 If the solution is satisfactory stop else go to Step 4.

There are a number of problems available which can be addressed as a many-objective problem. We are enlisting few of them, which will help the researcher to test their designed algorithm over many objective problems (Table 13).

Table 13 List of many objective test problem

Problems	Objectives
Knapsack	4, 5
Heuristic learning	7, 8
Nurse scheduling	25
DTLZ	8, 15
TSP	5, 10, 15, 20
Job shop scheduling	5, 10, 15, 20
DTLZ	10, 15, 20
Flight control system	8
Modified DTLZ	30, 5, 7, 9
Supersonic wing design	4
UF1, UF2, UF3	5
Inverted DTLZ1	5
C2-DTLZ2	4
Crash-worthiness in design of vehicle	4
Car side impact problem	4

4 Study of Swarm Intelligence for EEG Signal

Brain Computer Interface is a system that connects a human brain and machine. The main objective is to establish a new augmentative communication system that translates human intensions reflected by suitable brain signals [143]. Therefore, it is anticipated that the BCI system will become a technology not only for the average persons but also for the disabled persons. All types of activities start in a human being with the help of neurons, which is the basic element of the neural system of the human being. The actions or signals generated by the brain are electrical in nature and represent not only the brain function, but also the status of the whole body. Commonly three methods are used to know the changes within the brain with high temporal resolution of neuron interactions at the network level. They are EEG (Electroencephalography), MEG (Magneto encephalography), and fMRI (functional Magnetic Resonance Imaging) [144, 145]. Among them EEG signals are widely used and is the focus of this section/article. The EEG signal processing starts with acquisition of the signal and ends with a post processing task.

EEG signals of brain are very non-linear, random in nature, and dynamic. Human brain consists of number of cells they communicate with each other with the help of electrical impulses. The impulse can be measured by placing the electrode on the scalp of the agent. The EEG signals are generated through the cortical nerve cell inhibitory and excitatory post synaptic potential. These postsynaptic potential summate in the cortex and extend to the scalp surface where they are recorded as EEG. A typical EEG signal has the amplitude of about 10–100 v and the frequency

in the range of 1 Hz to about 100 Hz. During recording of the signal, their may be lots of chances that, different types of artifacts get added to the signal.

EEG signal can be recorded in two different ways. It can be recorded by finding the voltage difference between an active electrode on the scalp and a reference electrode on the ear lobe. In the other way the recording can be done by measuring the voltage difference between two scalp electrodes. The first method is known as the Monopolar whereas the latter is known as the Bipolar technique. EEG signals are associated with several descriptors like: (i) frequency or wave length, (ii) voltage, (iii) waveform, (iv) regulation, (v) manner of occurrence, (serial, continuous, random), (vi) locus, (vii) reactivity, and (viii) inter hemispheric coherence (Symmetry, Synchrony). Any change in the EEG signal pattern leads to abnormality [146, 147]. There are several methods used for measuring the dissimilarity among the patterns they are autocorrelation, high order statics, spectral error, and auto regressive modeling. EEG signals are characterized by delta waves: its frequency of oscillation is 1–4 Hz. It is most prominent at deep stage sleep, this wave characterize depth of sleep. Theta waves: its frequency of oscillation is 4–7 Hz. It is most prominent at dreaming sleep, this wave characterize drowsiness and sleep. Alpha waves: its frequency of oscillation is 8–15 Hz. It is most prominent at relaxation with closed eyes and it is characterized when the body is physically at rest. Beta waves: its frequency of oscillation is 25–100 HZ. This is most prominent at normal waking consciousness. There are various events which affect the EEG signals like sleep, epilepsy, reflexology, drug/anesthesia, diabetes, meditation, music and artifacts.

In a nutshell, while doing any type of analysis starting from pre to post processing operation in an EEG signal it becomes tedious to handle such a real life complex and huge amount of data [148]. On the other hand, it is also difficult for the analyst by using the non-population, non-parallel, and non-heuristic methods to handle such a situation. Hence, in this article, we turn our attention towards swarm based algorithm and its effectiveness in the domain of EEG analysis. Additionally, we discuss the implicit and explicit connections of multi-objective optimization in EEG analysis. The subsections are organized as follows: Sects. 4.1, 4.2, and 4.3 presents the application of ACO, PSO, and ABC for EEG signal analysis. In Sect. 4.4, we discuss the connectivity of multi and or many-objective optimization in EEG signal analysis (Fig. 4).

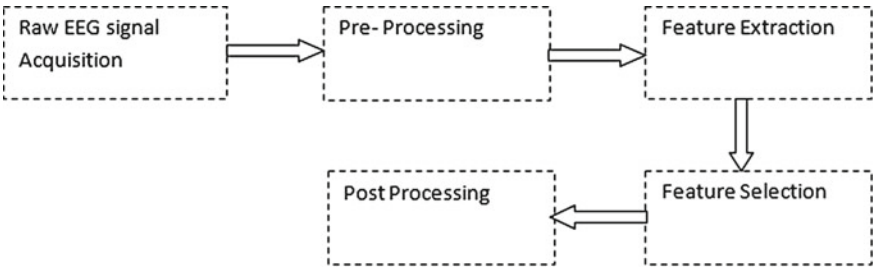


Fig. 4 Block diagram of EEG signal processing

4.1 ACO in EEG Signal Analysis

Bursa and Lhotska [150] and Dries and Peterson [149], have addressed the issue of ant-inspired clustering in the process of long-term electrocardiogram and electroencephalogram processing. They developed an ACO_DTree method which is based on the auto-catalytic collective behavior of real insect colonies. They have used their proposed method in EEG and compared it with WEKA Random Tree and found that their method provides more robust and improve performance. Bursa and Lhotska [150] also addressed automated classification of newborn sleep electroencephalogram using inductive classification methods. They have performed automated classification through ant colony approach (ACO_DTree algorithm) and the Group of Adaptive Models Evolution inductive models. Khushaba et al. [151] proposed a new feature extraction method which utilizes ant colony optimization in the selection of wavelet packet transform (WPT) and adopted in classifying bio-medical signals. The new algorithm named as intelligent artificial ants (AAI), which searches the wavelet packet tree for subsets of features that best interact together to produce high classification accuracies. The AAI method is the mixture of filter and wrapper approaches in the feature subset selection. The significance of the subsets selected by the ants is measured using linear discriminant analysis (LDA) classifier. The proposed method AAI is then tested on bio-signal driven applications, which is the brain computer interface (BCI) problem with 56 EEG channels. Results show that the proposed method achieves a maximum accuracy of 83 %. Again, Khushaba et al. [151] investigated the use of a combination of ACO and differential evolution called ANTDE for feature selection. They compare ANTDE, GA and BPSO, and reported that ANTDE's outperforms GA and BPSO due to the use of a mutual information based heuristic measure. In 2008, Khushaba et al. [152] have reported the application of clustering method inspired by the behavior of real ants in the nature in bio-medical signal processing. The ants cooperatively maintain and evolve a pheromone matrix which is used to select features. Their main aim was to design and develop a combination of feature extraction and classification methods, for automatic recognition of significant structure in biological signal recordings. The method is targeted towards speeding up and increasing objectivity of identification of important classes and may be used for online classification. The method can also be used in expert classification process. They have obtained significant results in EEG signals. Bursa and Lhotska [153] have presented a paper that describe the improved ant colony algorithm. They used the improved ant colony algorithm in emotion clustering of EEG signal to raise the efficiency of the image retrieval. Result shows that the improved algorithm has better clustering and accuracy rate.

4.2 PSO in EEG Signal Analysis

Qiu et al. in 2005 [155] did the feasibility study of EEG dipole source localization using PSO. Dipoles are widely used for approximating the sources of electrical activity in our brain. They showed that PSO is much more efficient than other evolutionary algorithms for EEG Dipole source localization. Qiu et al. [155] proposed an algorithm for EEG classification using radial basis PSO neural network for brain machine interfaces in 2007. In this, they proposed a mental task classification algorithm using PSO for a radial basis neural network. Features, were extracted from EEG signals that were recorded during five mental tasks like resting, mathematical multiplication, geometric figure rotation, letter composing and visual counting. In 2010 Nakamura et al. [157] and Pulraj et al. [156] proposed a method for evaluating the degree of human's preference based on EEG analysis. It is said that sense of touch is an important factor to decide what we like. They proposed a method for extracting the information on the sense of touch based on EEG analysis. They analyzed the EEG signals at the time of touching objects. They often used the frequency analysis for data analysis of the EEG. They applied PSO for selection of the significant component. For separating, support vector machine (SVM) was also used. Their proposed method has the capability of separating human's preference. Alp et al. [158] in 2009 proposed a PSO based technique on Dipole source reconstruction of brain signals. Their proposed method uses PSO for optimally choosing the dipole parameters. Simulation on synthetic data sets showed that their proposed method localized the dipoles into the actual location as well. In the real data sets, the actual dipole parameters are unknown. Due to this the fit error between the measured data and the reconstructed data is minimized. It is observed that their method reduces this error to the noise level by localizing only a few dipoles in the brain. In 2009, Satti et al. [159] worked on Spatio-spectral & temporal parameter searching using class correlation analysis and PSO for BCI. Distinct features play a vital role in enabling a computer to associate different EEG signals to different brain states. To ease the workload on the feature extractor and enhance separability between different brain states, numerous parameters, such as separable frequency bands, data acquisition channels and time point of maximum separability are chosen explicitly to each subject. Earlier research had shown that using subject specific parameters for the extraction of invariant characteristics specific to each brain state can significantly improve the performance and accuracy of a BCI. They developed a fast autonomous user-specific tuned BCI system using PSO to search for an optimal parameter combination based on the analysis of the correlation between different classes i.e. the R-Squared (R2) correlation coefficient rather than assessing overall system performance via performance measure such as classification accuracy. Lin and Hsieh [160] in 2009 proposed a neural classifier based on improved particle swarm optimization (IPSO) to classify an EEG of mental tasks for left-hand movement imagination, right-hand movement imagination, and word generation. First, the EEG patterns utilize principle component analysis (PCA) in order to reduce the feature dimensions. Then a three-layer neural network trained using PSO is used to realize a classifier.

The proposed IPSO method consists of the modified evolutionary direction operator (MEDO) and the traditional PSO. Their proposed MEDO combines the evolutionary direction operator (EDO) and the migration. The MEDO can strengthen the searching global solution. The IPSO algorithm can prevent premature convergence and outperform the other existing methods. Nasser Omer Sahel Ba-Karait et al. [161] proposed detection system of epileptic seizure in EEG signals which is based on Discrete Wavelet Transform (DWT) and Swarm Negative Selection (SNS) algorithm. DWT was used to analyze EEG signals at different frequency bands and statistics over the set of the wavelet coefficients were calculated to introduce the feature vector for SNS classifier. The SNS classification model uses negative selection and PSO algorithms to form a set of memory Artificial Lymphocytes (ALCs) that have the ability to distinguish between normal and epileptic EEG patterns. Thus, adapted negative selection is employed to create a set of self-tolerance ALCs. Whereas, PSO is used to evolve these ALCs away from self patterns towards non-self space and to maintain diversity and generality among the ALCs. The technique was approved to be robust and effective in detecting and localizing epileptic seizure in EEG recording. Wei and Wang [162] used Binary Multi-Objective Particle Swarm Optimization for Channel Selection in Motor Imagery Based Brain-Computer Interfaces in 2011. With the increase of channel numbers, multi-channel EEG signals need inconvenient recording preparation and complex calculation, this is time-consuming and lead to lower classification accuracy. To address this problem, they proposed a novel method, named binary multi-objective particle swarm optimization (BMOPSO) for channel reduction. In 2011, zbeyaz et al. [163] used PSO for Regularization and Kernel Parameters Optimization in EEG Signals Classification with SVM. In this study, firstly power spectrum was obtained by applying Auto-Regressive Burg (AR-Burg) method to the EEG signals. The data obtained from the analysis of AR-Burg was classified with Support Vector Machines (SVM). Classification achievements were investigated for some kernel functions used in SVM. Regularization parameter and kernel parameter that increase the success of classification were calculated with a novel approach of PSO, such that, global best results for classification were searched by investigating optimum values. As a result of this study a new algorithmic approach presented for the diagnosis of epilepsy patients. Kim et al. [164] proposed "A Binary PSO-Based Optimal EEG Channel Selection Method for a Motor Imagery Based BCI System". Brain-computer interface based on motor imagery is a system that transforms a subject's intention into a control signal by classifying EEG signals obtained from the imagination of movement of a subject's limbs. Using many channels cause other problems. When applying a common spatial pattern (CSP), which is an EEG extraction method, many channels cause an over fitting problem, in addition there is difficult using this technique for medical analysis. To overcome these problems, they suggested a PSO applied to CSP. In 2012, Arslan et al. [165] have used a hybrid Structure of ANN and PSO for EEG Signals Classification. ANN and PSO techniques designed in the form of a hybrid structure are used for diagnosis of epilepsy patients via EEG signals. Attributes of EEG signals are needed to be determined by employing EEG signals which are recorded using EEG. From this data, four characteristics are extracted for the classification process. 20 % of available data

is reserved for testing while 80 % of available data is being reserved for training. These actions were repeated five times by performing cross-validation process. PSO is used for updating the weights during training ANN and a program is constituted for classification of EEG signals. Atyabi et al. [166] proposed on Adapting Subject-Independent Task-Specific EEG Feature Masks using PSO in 2012. It is reported that dimension reduction is an important step toward asynchronous EEG based BCI systems, with EA based Feature/Electrode Reduction (FR/ER) methods showed significant potential for this purpose. A PSO based approach can reduce 99 % of the EEG data in this manner while demonstrating generalizability through the use of 3 new subsets of features/electrodes that were selected based on the best performing sub-set on the validation set, the best performing sub-set on the testing set, and the most commonly used features/electrodes in the swarm. Their study focused on applying the subsets generated from 4 subjects on a 5th one. Two schemes for this are implemented based on (i) extracting separate subsets of feature/electrodes for each subject (out of 4 subjects) and combining the final products together for use with the 5th subject, and (ii) concatenating the pre processed EEG data of 4 subjects together and extracting the desired subset with PSO for use with the 5th subject. In 2013, Shirvany et al. [167], have proposed a method for solving an inverse problem EEG-based source localization. To determine the location of the brain sources that are responsible for the measured potentials at the scalp electrodes. They proposed a new global optimization method based on PSO to solve the epileptic spike EEG source localization inverse problem. In a forward problem a modified subtraction method is proposed to reduce the computational time.

4.3 ABC in EEG Signal Analysis

In 2013, Ahirwal et al. [148] have used ABC to construct Adaptive Noise Canceller (ANC) for EEG filtering with modified range selection, described as bounded range ABC. They have also implemented ANC with RLS and LMS. They have performed the comparative study of conventional methods like LMS, RLS with ABC, and they found that ABC performs well than conventional methods. They also proposed a new form of controlled search space to stabilize the randomness of swarm intelligence, especially for the EEG signal. The proposed controlled search space technique was tested on each of the swarm intelligence techniques and found to be more accurate and powerful.

In 2013, Rakshit et al. [168] have used EEG based BCI to decode the various movements related data generated from the motor areas of the brain. One the issues in BCI research is the presence of redundant data in the features of a given data set, they have used an ABC cluster algorithm to reduce the features, and acquired their corresponding values. The result shows that it has the highest accuracy of 64.29 % and it also reduced the problem feature. From this study, we have concluded that there are lots of tasks which can be carried out in future by using ABC and its variants.

4.4 Towards Multiple and Many Objectives of EEG Signal

Now-a-days multiple and many objective optimization has been applied in many fields of science ranging from health to engineering sciences, where optimal decisions need to be taken in the presence of trade-offs between three or more objectives. In this section, our main concerned is to revealing conflicting objectives if any during the EEG signal analysis.

Recall that the non-invasive BCI uses EEG signals to capture the brain signal associated with predefined mental tasks. The number of channels used by an EEG system can vary according to the experiment held and the hardware design. It usually ranges between 16 and 256 channels. According to Event Related Desynchronization/Synchronization (ERD/ERS) research, motor imagery experiments can use only the channels at the contralateral hemispheres, which can be as few as 3–5 channels. Using a lot of channels for recording can be useful for medical and diagnostic purposes. For BCI system and especially when building online system, the number of channels should be as minimum as possible. In order to avoid a large number of channels one can choose several electrode positions that are known from neuroscience and psychology studies. Although this approach can be very useful, it ignores the fact that different subjects respond differently and the optimal positioning of the electrodes may vary. The other way around this problem is to use a large number of channels and use some methods to reduce the dimensionality of the input features or to select the best set of channels for each subject. Hence, this problem can be realized as a multi-objective optimization problems by minimizing the number of channels and maximize the classification accuracy. Sleep disorders, epilepsy identification problem, etc. can also be viewed as a multi-objective problem.

In the normal adult, there are two stages of sleep that alternate at about 90-min intervals. Rapid eye movement sleep can be described as a period when the brain is active and the body is paralyzed (except for eye movements, middle ear ossicles, and respiration). In non-rapid eye movement sleep, the brain is less active but the body can move. Non rapid eye movement sleep is composed of four stages that are differentiated on the basis of EEG characteristics. When normal individuals first fall asleep, they enter stage 1 (sleep drowsiness) and then progress through stages 2, 3, and 4 of non-rapid eye movement. Stages 3 and 4 (deep sleep) are often called slow wave sleep or delta sleep because they are characterized by high amplitude, slow waves (also called delta waves) on EEG. Slow wave sleep may last from a few minutes to an hour, depending on the person's age, before reversion back to stage 2 sleep. Shortly after this, the first REM sleep period begins, lasting about 15–20 min and is followed by another non-REM cycle. This alternating pattern continues throughout the night, but as the night progresses stages 3 and 4 are less apparent and the periods of REM sleep grow longer. These different phases of sleep can be analyzed by extracting the relevant features from the EEG signal. This helps in finding the sleep disorder.

Epilepsy is a neuron disorder from which the general function of the brain is affected. There are three typical stages of epilepsy like interictal, preictal and ictal. Therefore, a model can be designed to identify the epileptic and can be classified by

considering the objectives like maximization of classification accuracy, minimization of number of features, and minimization of number of instances. Hence, it can be inferred that many problems through EEG signal can be viewed as multi objective problems, therefore, suitable multi-objective swarm intelligence techniques can be used as a tool.

5 Discussion and Future Research

Over the years many swarm intelligence techniques like ant colony optimization, particle swarm optimization, artificial bee colony, Bat algorithm [169], Wasp colony optimization [170], etc. have been developed to solve many intractable/complex single, multiple, and many objective optimization problems. However, this paper restrict its discussion with ant colony optimization, particle swarm optimization, and artificial bee colony to solve multiple and many objective optimization problems. We have started our work with an intensive study on multi-objective optimization problems and how it differs from many objective problems. The exploration and exploitation capability of ACO, PSO, and ABC has been studied along with the diversity mechanism. Next part of the study, a basic framework of these swarm intelligent techniques is presented to explain how they can be used for handling many objective problems. This framework is a common one for all the three categories of swarm optimization technique. We have also discussed about some of the indicators which are used for handling many objective problem. This part presents a clear view on the many objective problems and how they can be handled.

At the last part of this work, we have studied the EEG signal analysis as it is now one of the promising area of BCI. This paper puts a light on the basic of EEG signal and in one section also we discuss multi objective issues associated with EEG signal analysis. We focus our study on how the swarm intelligence techniques are used for EEG signal analysis. In a nutshell, our paper presents a clear view on swarm intelligent techniques and their usages in handling multiple and many objective problems commendable in the field of EEG signal analysis.

References

1. Bonabeau, E., Dorigo, M., Thraulaz, G.: Swarm Intelligence: From Natural to Artificial Systems, vol. 23. Oxford University Press, Oxford (1999)
2. Rafie, F.M., Manzari, S.M., Khashei, M.: Scheduling flight perturbations with ant colony optimization approach. Int. J. Comput. Sci. Artif. Intell. **2**(2), 1–9 (2012)
3. Sashikumar, G.N., Mahendra, A.K., Gouthaman, G.: Multi-objective shape optimization using ant colony coupled computational fluid dynamics solver. Comput. Fluids **46**(1), 298–305 (2011)
4. Prasad, S., Zaheeruddin, Lobiyal, D.K.: Multi-objective multicast routing in wireless ad-hoc networks: an ANT colony approach. In: Proceedings of IEEE 3rd International Conference

in Advance Computing, pp. 511–514 (2013)
5. Goa, S.: Solving weapon-target assignment problems by a new ANT colony algorithm. International Symposium on Computational Intelligence & Design, pp. 221–224. IEEE Press (2008)
6. Yang, X.S.: Nature-inspired metaheuristic algorithms. Luniver Press, Beckington, UK (2008)
7. Lim, C.P., Jain, L.C., Dehuri, S. (eds.): Innovations in Swarm Intelligence. Springer, Berlin (2009)
8. Ishibuchi, H., Tsukamoto, N., Nojima, Y.: Evolutionary many-objective optimization: a short review. In: Proceedings of 2008 IEEE Congress on Evolutionary Computation, pp. 2424–2431. Hong Kong (2008)
9. Millonas, M.M.: Swarms, phase transition and collective intelligence. In: Langton, C.G. (ed.) Artificial Life III. Addison Wesley, Reading (1994)
10. Karaboga, D.: Artificial be colony algorithm. Scholarpedia **5**(3), 6915 (2010)
11. Dehuri, S., Cho, S.-B., Ghosh, S. (eds.): Integration of Swarm Intelligence and Artificial Neural Networks. World Scientific Press, New Jersey (2011)
12. Dehuri, S., Cho, S.-B. (eds.): Knowledge Mining Using Intelligent Agents. Imperial College Press, London (2010)
13. Passino, K.M.: Biomimicry of bacterial foraging for distributed optimization and control. IEEE Control Syst. Mag. **22**(3), 52–67 (2002)
14. Mishra, B.S.P., Dehuri, S., Wang, G.-N.: A state-of-the-art review of artificial bee colony in the optimization of single and multi-criteria. Int. J. Appl. Metaheuristics Comput. **4**(4), 23–45 (2013)
15. Sato, H., Aguirre, H.E., Tanaka, K.: Controlling dominance area of solutions and its impact on the performance of MOEAs. Lecture Notes in Computer Science 4403: Evolutionary Multi-Criterion Optimization- EMO 2007. Springer, Berlin (2007)
16. Ikeda, K., Kita, H., Kobayashi, S.: Failure of pareto-based MOEAs: does non dominated really mean near to optimal? In: Proceedings of 2001 IEEE Congress on Evolutionary Computation, pp. 957–962. Seoul (2001)
17. Branke, J., Kauler, T., Schmeck, H.: Guidance in evolutionary multi-objective optimization. Adv. Eng. Softw. **32**(6), 499–507 (2001)
18. Branke, J., Deb, K.: Integrating user preferences into evolutionary multi-objective optimization. In: Jin, Y. (ed.) Knowledge Incorporation in Evolutionary Computation, pp. 461–477. Springer, Heidelberg (2005)
19. Drechsler, N., Drechsler, R., Becker, B.: Multi-objective optimization based on relation. Lecture Notes in Computer Science 1993: Evolutionary Multi-Criterion Optimization - EMO 2001. pp. 154–166. Springer, Berlin (2001)
20. Slflow, A., Drechsler, N., Drechsler, R.: Robust multi-objective optimization in high dimensional spaces. Lecture Notes in Computer Science 4403: Evolutionary Multi-Criterion Optimization - EMO 2007, pp. 715–726. Springer, Berlin (2007)
21. Kppen, M., Yoshida, K.: Substitute distance assignments in NSGA-II for handling many-objective optimization problems. Lecture Notes in Computer Science 4403: Evolutionary Multi-Criterion Optimization - EMO 2007, pp.727–741. Springer, Berlin (2007)
22. Corne, D., Knowles, J.: Techniques for highly multi-objective optimization: some non-dominated points are better than others. In: Proceedings of 2007 Genetic and Evolutionary Computation Conference, pp. 773–780. London (2007)
23. Kukkonen, S., Lampinen, J.: Ranking-dominance and many objective optimization. In: Proceedings of 2007 IEEE Congress on Evolutionary Computation, pp. 3983–3990. Singapore (2007)
24. Wagner, T., Beume, N., Naujoks, B.: Pareto, aggregation and indicator-based methods in many-objective optimization. Lecture Notes in Computer Science 4403: Evolutionary Multi-Criterion Optimization - EMO 2007, pp. 742–756. Springer, Berlin (2007)
25. Ishibuchi, H., Tsukamoto, N., Nojima, Y.: Iterative approach to indicator-based multi-objective optimization. In: Proceedings of 2007 IEEE Congress on Evolutionary Computation, pp. 3697–3704. Singapore (2007)

26. Brockhoff, D., Zitzler, E.: Are all objectives necessary? on dimensionality reduction in evolutionary multiobjective optimization. Lecture Notes in Computer Science 4193: Parallel Problem Solving from Nature - PPSN IX, pp. 533–542. Springer, Berlin (2006)
27. Miettinen, K.: Nonlinear Multiobjective Optimization. Kluwer, Boston (1998)
28. Knowles, J.D., Corne, D.W.: On metrics for comparing non dominated sets. In: Proceedings of 2002 IEEE Congress on Evolutionary Computation, pp. 711–716. Honolulu (2002)
29. Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C.M., da Fonseca, V.G.: Performance assessment of multi-objective optimizers: ananalysis and review. IEEE Trans. Evol. Comput. **7**(2), 117–132 (2003)
30. Okabe, T., Jin, Y., Sendhoff, B.: A Critical Survey of Performance Indices for Multi-objective Optimization. In: Proceedings of 2003 IEEE Congress on Evolutionary Computation, pp. 878–885. Canberra (2003)
31. Ishibuchi, H., Doi, T., Nojima, Y.: Incorporation of scalarizing fitness functions into evolutionary multi-objective optimization algorithms. Lecture Notes in Computer Science 4193: Parallel Problem Solving from Nature - PPSN IX, pp. 493–502. Springer, Berlin (2006)
32. Ishibuchi, H., Nojima, Y.: Optimization of scalarizing functions through evolutionary multiobjective optimization. Lecture Notes in Computer Science 4403: Evolutionary Multi-Criterion Optimization - EMO 2007, pp. 51–65. Springer, Berlin (2007)
33. Hughes, E.J.: MSOPS-II: a general-purpose many-objective optimizer. In: Proceedings of 2007 IEEE Congress on Evolutionary Computation, pp. 3944–3951. Singapore, pp. 25–28 (2007)
34. Hughes, E.J.: Multiple single objective pareto sampling. In: Proceedings of 2003 IEEE Congress on Evolutionary Computation, pp. 2678–2684. Canberra (2003)
35. Ishibuchi, H., Murata, T.: A multi-objective genetic local search algorithm and its application to flowshop scheduling. IEEE Trans. Syst. Man Cybern. - Part C **28**(3), 392–403 (1998)
36. Jaszkiewicz, A.: Genetic local search for multi-objective combinatorial optimization. Eur. J. Oper. Res. **137**(1), 50–71 (2002)
37. Ishibuchi, H., Yoshida, T., Murata, T.: Balance between genetic search and local search in Memetic algorithms for multi-objective permutation flowshop scheduling. IEEE Trans. Evol. Comput. **7**(2), 204–223 (2003)
38. Fleming, P.J., Purshouse, R.C., Lygoe, R.J.: Many-objective optimization: an engineering design perspective. Lecture Notes in Computer Science 3410: Evolutionary Multi-Criterion Optimization - EMO 2005, pp. 14–32. Springer, Berlin (2005)
39. Deb, K., Sundar, J.: Preference point based multi-objective optimization using evolutionary algorithms. In: Proceedings of 2006 Genetic and Evolutionary Computation Conference, pp. 635–642. Seattle (2006)
40. Thiele, L., Miettinen, K., Korhonen, P.J., Molina, J.: A Preference Based Interactive Evolutionary Algorithm for Multiobjective Optimization. Helsinki School of Economics, Helsinki (2007). Working Paper, W-412
41. Zitzler, E., Brockhoff, D., Thiele, L.: The hypervolume indicator revisited: on the design of pareto-compliant indicators via weighted integration. Lecture Notes in Computer Science 4403: Evolutionary Multi-Criterion Optimization - EMO 2007, pp. 862–876. Springer, Berlin (2007)
42. Fonseca, C.M., Fleming, P.J.: Multi-objective optimization and multiple constraint handling with evolutionary algorithms - Part I: a unified formulation. IEEE Trans. Syst. Man Cybern. - Part A **28**(1), 38–47 (1998)
43. Coello, C.A.C.: Handling preferences in evolutionary multi-objective optimization: a survey. In: Proceedings of 2000 IEEE Congress on Evolutionary Computation, pp. 30–37. San Diego (2000)
44. Cvetkovic, D., Parmee, P.: Preferences and their application in evolutionary multi-objective optimization. IEEE Trans. Evol. Comput. **6**(1), 42–57 (2002)
45. Zitzler, E., Thiele, L.: Multi-objective evolutionary algorithms: a comparative case study and the strength pareto approach. IEEE Trans. Evol. Comput. **3**(4), 257–271 (1999)

46. Brockhoff, D., Wagner, T., Trautman, H.: On the properties of the R2 indicator. In: Proceedings of 2012 Genetic Evolutionary Computation Conference (GECCO2012), pp. 465–472. ACM Press, Philadelphia (2012)
47. Gomez, R.H., Coello, C.A.C.: MOMBI: A new Metaheuristics for many-objective optimization based on the R2 indicator. IEEE Congr. Evol. Comput. (CEC-2013) **1**, 2488–2495 (2013)
48. Beume, N., Naujoks, B., Emmerich, M.: SMS-EMOA: Multi-objective selection based on dominated hypervolume. Eur. J. Oper. Res. **180**(3), 1653–1669 (2007)
49. Garca-Martnez, C., Cordon, O., Herrera, F.: Discrete optimization a taxonomy and an empirical analysis of multiple objective ant colony optimization algorithms for the bi-criteria TSP. Eur. J. Oper. Res. **180**, 116–148 (2007)
50. Chaharsooghi, S.K., Kermani, A.H.M.: An effective Ant Colony Optimization Algorithm (ACO) for Multi-objective Resource Allocation Problem (MORAP). Appl. Math. Comput. **200**(1), 167–177 (2008)
51. Mariano, C.E., Morales, E.: MOAQ: an ant-Q algorithm for multiple objective optimization problems. In: Banzhaf, W., Daida, J., Eiben, A.E., Garzon, M.H., Hnavar, V., Jakiela, M., Smith, R.E. (eds.) In: Proceedings of the Genetic and Evolutionary Computing Conference (GECCO 99), pp. 894–901. San Francisco (1999)
52. Iredi, S., Merkle, D., Middendorf, M.: Bi-criterion optimization with multi colony ant algorithms. In: Proceedings First International Conference on Evolutionary Multi-criterion Optimization (EMO01). Lecture Notes in Computer Science 1993, pp. 359–372 (2001)
53. Doerner, K., Gutjahr, W.J., Hartl, R.F., Strauss, C., Stummer, C.: Pareto ant colony optimization: a metaheuristics approach to multi-objective portfolio selection. Ann. Oper. Res. **131**(1–4), 79–99 (2004)
54. Gambardella, L., Taillard, E., Agazzi, G.: MACS-VRPTW: a multiple ant colony system for vehicle routing problems with time windows. In: Corne, D., Dorigo, M., Glover, F. (eds.) New Ideas in Optimization, pp. 73–76. McGraw-Hill (1999)
55. Doerner, K., Hartl, R.F., Teimann, M.: Are COMPETants more competent for problem solving? the case of full Truckload transportation. Cent. Eur. J. Oper. Res. **11**(2), 115–141 (2003)
56. Gravel, M., Price, W.L., Gagne, C.: Scheduling continuous casting of aluminium using a multiple objective ant colony optimization Metaheuristics. Eur. J. Oper. Res. **143**(1), 218–229 (2002)
57. Ali, A.-D., Belal, M.A., Al-Zoubi, M.B.: Load balancing of distributed systems based on multiple ant colonies optimization. Am. J. Appl. Sci. **7**(3), 428–433 (2010)
58. Lopez-Ibanez, M., Stutzle, T.: The automatic design of multiobjective ant colony optimization algorithms. IEEE Trans. Evol. Comput. **16**(6), 861–875 (2012)
59. Lopez-Ibanez, M., Stutzle, T.: Automatic configuration of multi-objective ant colony optimization algorithms. In: Dorigo, M., et al., (eds.) Ants. Lecture Notes in Computer Science, vol. 6234, pp. 95–106. Springer, Heidelberg (2010)
60. Kennedy, J., Eberhart, R.C.: Particle swarm optimization. In: Proceedings of the IEEE International Conference on Neural Networks, pp. 1942–1948. Perth (1995)
61. Shi, Y., Eberhart, R.C.: Empirical study of particle swarm optimization. In: Proceedings of the 1999 Congress on Evolutionary Computation, pp. 1945–1950 (1999)
62. Bergh, F.V.D., Engelbrecht, A.P.: A cooperative approach to particle swarm optimization. IEEE Trans. Evol. Comput. **8**(3), 225–239 (2004)
63. Coello, C.A.C., Lechuga, M.: MOPSO: A proposal for multi-objective particle swarm optimization. In: Proceedings of the 9th IEEE World Congress on Computational Intelligence, pp. 1051–1056. Honolulu (2002)
64. Fieldsend, J.E., Singh, S.: A multi-objective algorithm based upon particle swarm optimization, an efficient data structure and turbulence. In: Proceedings of the Workshop on Computational Intelligence, pp. 37–44. Brimingham (2002)
65. Laumanns, M., Zitzler, E., Thiele, L.: A unified model for multi-objective evolutionary algorithm with Elitism. In: Proceedings of the IEEE World Congress on Evolutionary Computation, pp. 46–53. Piscataway (2000)

66. Hu, X., Eberhart, R.C.: Multi-objective optimization using dynamic neighborhood particle swarm optimization. In: Proceedings of the IEEE World Congress on Evolutionary Computation, pp. 1677–1681. Honolulu (2002)
67. Parsopoulos, K.E., Vrahatis, M.N.: Particle swarm optimization method in multi-objective problem. In: Proceedings of the ACM, Symposium on Applied Computing, pp. 603–607. Madrid (2002)
68. Schaffer, J.D.: Multi-objective optimization with vector evaluated genetic algorithms. In: Proceedings of the First International Conference on Genetic Algorithms, pp. 93–100 (1985)
69. Mostaghim, S., Teich, J.: Strategies for finding good local guides in MultiObjective Particle Swarm Optimization (MOPSO). In: Proceedings of the IEEE Symposium onSwarm Intelligence, pp. 26–33 (2003)
70. Coello, C.A.C., Pulido, G.T., Lechuga, M.S.: Handling multiple objectives with particle swarm optimization. IEEE Trans. Evol. Comput. **8**(3), 256–279 (2004)
71. Fieldsend, J.E.: Multi-objective particle swarm optimization methods, Technical Report No. 419. Department of Computer Science, University of Exeter (2004)
72. Ray, T., Liew, K.M.: A swarm metaphor for multi-objective design optimization. Eng. Optim. **34**(2), 141–153 (2002)
73. Pulido, G.T., C. A. C., Coello: Using clustering techniques to improve the performance of a particle swarm optimizer. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 225–237. Seattle (2004)
74. Mostaghim, S., Teich, J.: Particle swarm inspired evolutionary algorithm PSEA for multi-objective optimization problem. In: Proceedings of the IEEE World Congress on Evolutionary Computation, pp. 2292–2297. Canberra (2003)
75. Dorigo, M., Blum, C.: Ant colony optimization theory: a survey. Theor. Comput. Sci. **344**, 243–278 (2005)
76. Li, X.: A non-dominated sorting particle swarm optimizer for multi-objective optimization. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 37–38 (2003)
77. Deb, K., Pratap, A., Agrawal, S., Meyarivan, T.: A fast and Elitist multi-objective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. **6**(2), 182–197 (2002)
78. Sierra, M.R., Coella, C.A.C.: Improving PSO-based multi-objective optimization using crowding, mutation and dominance. In: Proceedings of the 3rd International Conference on Evolutionary Multi-criterion Optimization, pp. 505–519. Guanajuato (2005)
79. Ho, S.L., Shiyou, Y., Lo, E.W.C., Wong, H.C.: A particle swarm optimization b based method for multi-objective design optimization. IEEE Trans. Mag. **41**(5), 1756–1759 (2005)
80. Villalobos-Anias, M.A., Pulido, G.T., Coello, C.A.C.: A proposal to use stripes to maintain diversity in a multi-objective particle swarm optimizer. In: Proceedings of IEEE Swarm Intelligence Symposium, pp. 22–29. Pasadena (2005)
81. Salazar-Lechuga, M., Rowe, J.: Particle swarm optimization problems. In: Proceedings of IEEE World Congress on Evolutionary Computation, pp. 1204–1211. Edinburgh (2005)
82. Janson, S., Merkle, D.: A new multi-objective particle swarm optimization algorithms using clustering applied to automated docking. In: Hybrid Metaheuristics Second International Workshop, pp. 128–142. Barcelona (2005)
83. Lewis, A.: The effect of population density on the performance of a spatial social network algorithm for multi-objective optimization. In: Proceedings of IEEE International Symposium on Parallel & Ditributed Processing, pp. 1–6. Rome (2009)
84. Leong, W.F., Yen, G.G.: Impact of tuning parameters on dynamic swarms in PSO-based multi-objective optimization. In: Proceedings of IEEE Congress on Evolutionary Computation, pp. 1317–1324. Hong Kong (2008)
85. Lewis, A., Ireland, D.: Automated solution selection in multi-objective optimization. In: Proceedings of IEEE Congress on Evolutionary Computation, pp. 2163–2169 (2008)
86. Cagnina, L., Esquivel, S., Coello, C.A.C.: A Particle Swarm Optimizer for Multi-objective Optimization. J. Comput. Sci. Technol. **5**(4) (2005)
87. Laura, D., Mihai, O.: Evolving the structure of particle swarm optimization algorithm. Evolutionary Computation in Combinatorial Optimization, pp. 25–36. Springer, Berlin (2006)

88. Goldberg, E.F.G., de Souza, G.R., Goldberg, M.C.: Particle swarm optimization for the bi-objective degree constrained minimum spanning tree. IEEE Congress on Evolutionary Computation, pp. 16–21. Sheraton Vancouver Wall Centre Hotel, Vancouver (2006)
89. Ho, S., Ku, W., Jou, J., Hung, M., Ho, S.: Intelligent particle swarm optimization in multi-objective problems. PAKDD 2006. LNAI 3918, pp. 790–800. Springer, Berlin (2006)
90. Koppen, M., Veenhuis, C.: Multi-objective particle swarm optimization by fuzzy- Pareto-dominance meta-heuristic. Int. J. Hybrid Intell. Syst. **3**, 179–186 (2006)
91. Chiu, S., Sun, T., Hsieh, S.: Cross-searching strategy for multi-objective particle swarm optimization. Expert Syst. Appl. **37**(8), 5872–5886 (2010)
92. Peng, W., Zhang, Q.: A decomposition-based multi-objective particle swarm optimization algorithm for continuous optimization problems. In: IEEE International Conference on Granular Computing, pp. 534–537. Hangzhou (2008)
93. Padhye, N., Branke, J., Mostaghim, S.: Empirical comparison of MOPSO methods guide selection and diversity preservation. In: IEEE Congress on Evolutionary Computation, CEC09, pp. 2516–2523. Trondheim (2009)
94. Cabrera, J.C.F., Coello, C.A.C.: Micro-MOPSO: a multi-objective particle swarm optimizer that uses a very small population size. In: Proceedings of Studies in Computational Intelligence, vol. 261, pp. 83–104. Springer, Berlin (2010)
95. Wang, Y., Yang, Y.: Particle swarm optimization with preference order ranking for multi-objective optimization. Inf. Sci. **179**(12), 1944–1959 (2009)
96. Goh, C.K., Tan, K.C.B., Liu, D.S.B., Chiamb, S.C.: A competitive and cooperative co-evolutionary approach to multi-objective particle swarm optimization algorithm design. Eur. J. Oper. Res. **202**, 42–52 (2009)
97. Tsai, S., Sun, T., Liu, C., Hsieh, S., Wu, W., Chiu, S.: An improved multi-objective particle swarm optimizer for multi-objective problems. Expert Syst. Appl. **37**(8), 5872–5886 (2010)
98. Zheng, X., Liu, H.: A hybrid vertical mutation and self-adaptation based MOPSO. Comput. Math. Appl. **57**, 2030–2038 (2009)
99. Wang, Y., Yang, Y.: Particle swarm optimization with preference order ranking for multi-objective optimization. Inf. Sci. **179**(12), 1944–1959 (2009)
100. Tereshko, V., Loengarov, A.: Collective Decision-Making in Honey Bee Foraging Dynamics. Computing and Information Systems **9**(3), 1–7. University of the West of Scotland, UK (2005)
101. Chauhan, N.C., Kartikeyan, M.V., Mittal, A.: Design of RF window using multi-objective particle swarm optimization. In: Proceedings of International Conference on Recent Advances in Microwave Theory and Applications, pp. 34–37. Jaipur (2008)
102. Falcon, R., DePaire, B., Vanhoof, K., Abraham, A.: Towards a suitable reconciliation of the findings in collaborative fuzzy clustering. In: Proceedings of Eighth International Conference on Intelligent Systems Design and Applications, vol. 3, pp. 652–657. IEEE, USA (2008)
103. de Carvalho, A.B., Pozo, A., Vergilio, S., Lenz, A.: Predicting fault proneness of classes through a multi-objective particle swarm optimization algorithm. In: Proceedings of 20th IEEE International Conference on Tools with Artificial Intelligence, vol. 2, pp. 387–394 (2008)
104. Martin, J.E., Pantoja, M.F., Bretones, A.R., Garcia, S.G., de Jong Van Coevorden, C.M., Martin, R.G.: Exploration of multi-objective particle swarm optimization on the design of UWB antennas. In: Proceedings of 3rd European Conference on Antennas and Propagation, pp. 561–565. IEEE, Berlin (2009)
105. Pang, H., Chen, F.: An Optimization Approach for Intersection Signal Timing Based on Multi-Objective Particle Swarm Optimization. In: Proceeding of IEEE Conference on Cybernetics and Intelligent Systems. IEEE, Chengdu (2008)
106. Hazra, J., Sinha, A.K.: Congestion management using multi-objective particle swarm optimization. IEEE Trans. Power Syst. **22**(4), 1726–1734 (2007)
107. Qasem, S.N., Shamsuddin, S.M.: Radial basis function network based on time variant multi-objective particle swarm optimization for medical diseases diagnosis. Appl. Soft Comput. **11**(1), 1427–1438 (2011)

108. Pindoriya, M.N., Singh, S.N., Singh, S.K.: Multi-objective mean-variance skewness model for generation portfolio allocation in electricity markets. Electr. Power Syst. Res. **80**(10), 1314–1321 (2010)
109. Sha, D.Y., Lin, H.: A multi-objective PSO for job-shop scheduling problems. Expert Syst. Appl. **37**(2), 1065–1070 (2010)
110. Wang, L., Singh, C.: Stochastic combined heat and power dispatch based on multi-objective particle swarm optimization. Int. J. Electr. Power Energy Syst. **30**(3), 226–234 (2008)
111. Montalvo, I., Izquierdo, J., Schwarze, S., Prez-Garca, R.: Multi-objective particle swarm optimization applied to water distribution systems design: an approach with human interaction. Math. Comput. Model. (2010)
112. Liu, Y.: Automatic calibration of a rainfall-runoff model using a fast and Elitist multi-objective particle swarm algorithm. Expert Syst. Appl. **36**(5), 9533–9538 (2009)
113. Zhang, W., Liu, Y.: Multi-objective reactive power and voltage control based on fuzzy optimization strategy and fuzzy adaptive particle swarm. Int. J. Electr. Power Energy Syst. **30**(9), 525–532 (2008)
114. Cai, J., Ma, X., Li, Q., Li, L., Peng, H.: A multi-objective chaotic particle swarm optimization for environmental/economic dispatch. Energy Convers. Manag. **50**(5), 1235–1318 (2009)
115. Ganguly, S., Sahoo, N.C., Das, D.: Multi-objective particle swarm optimization based on fuzzy-pareto-dominance for possibilistic planning of electrical distribution systems incorporating distributed generation. Fuzzy Sets Syst. **213**, 47–73 (2013)
116. Sankaran, A., Manne, J.R.: Probabilistic multi-objective optimal design of composite channels using particle swarm optimization. J. Hydraul. Res. **51**(4) (2013)
117. Karaboga, D.: An idea based on honey bee swarm for numerical optimization, Technical Report-TR06. Erciyes University, Engineering Faculty, Computer Engineering Department (2005)
118. Teodorovic, D., Dell, M.O.: Bee colony optimization - a cooperative learning approach to complex transportation problems. In: Proceedings of 10th EWGT Meeting and 16th Mini EURO Conference, pp. 51–60 (2005)
119. Wedde, H., Farooq, M.: The wisdom of the hive applied to mobile ad-hoc networks. In: Proceedings of the Swarm Intelligence Symposium 2005, pp. 341–348. Pasadena (2005)
120. Drias, H.S.S., Yahi, S.: Cooperative bees swarm for solving the maximum weighted satisfiability problem. Computational Intelligence and Bioinspired Systems. LNCS, vol. 3512, pp. 318–325. Springer, Berlin (2005)
121. Chong, C.S., Sivakumar, A.I., Malcolm Low, Y.H., Gay, K.L.: A bee colony optimization algorithm to job shop scheduling. In: Proceedings of the 38th conference on Winter simulation WSC '06, pp. 1954–1961. California (2006)
122. Quijano, N., Passino, K.: Honey bee social foraging algorithms for resource allocation, Part-I: algorithm and theory. In: Proceedings of American Control Conference, ACC '07, pp. 3383–3388. IEEE, New York (2007)
123. Karaboga, D., Akay, B.: A comparative study of artificial bee colony algorithm. Appl. Math. Comput. **214**, 108–132 (2009)
124. Karaboga, D., Ozturk, C.: Neural networks training by artificial bee colony algorithm on pattern classification, pp. 279–292. Neural Network World, Institute of Computer Science AS CR v. v. i, Czech Republic (2009)
125. Karaboga, N.: A new design method based on artificial bee colony algorithm for digital IIR filters. J. Franklin Institute, **346**(4), 328–348 Elsevier, Netherlands (2009)
126. Karaboga, D., Ozturk, C.: A novel clustering approach: Artificial Bee Colony (ABC) algorithm. Appl. Soft Comput. **11**(1), 652–657 (2011)
127. Xu, C., Duan, H.: Artificial Bee Colony (ABC) optimized Edge Potential Function (EPF) approach to target recognition for low-altitude aircraft. Pattern Recognit. Lett. **31**(13), 1759–1772 (2010)
128. Yu, W.-J., Zhang, J., Chen, W.-N.: Adaptive artificial bee colony optimization, GECCO13. In: Proceedings of the Fifteenth Annual Conference on Evolutionary Computation Conference, pp. 153–158 (2013)

129. Omkar, S., Senthilnath, N., Khandelwal, J.R., Naik, G.N., Gopalakrishnan, S.: Artificial Bee Colony (ABC) for multi-objective design optimization of composite structures. J. Appl. Soft Comput. **11**(1), 489–499 (2011)
130. Zou, W., Zhu, Y., Chen, H., Zhang, B.: Solving multi-objective optimization problems using artificial bee colony algorithm. Dyn. Nat. Soc. (569784), 37. Hindawi Publishing Corporation (2011). doi:10.1155/2011/569784
131. Qu, B.Y., Suganthan, P.N.: Multi-objective evolutionary programming without non-domination sorting is up to twenty times faster. In: Proceedings of Congress on Evolutionary Computation, CEC09, pp. 2934–2939 (2009)
132. Li, H., Zhang, Q.: Multi-objective optimization problems with complicated Pareto sets, MOEA/D and NSGA-II. IEEE Trans. Evol. Comput. **13**(2), 284–302 (2009)
133. Akbari, R., Ziarati, R.H.K., Hassanizadeh, B.: A multi-objective artificial bee colony algorithm. Swarm Evol. Comput. **2**, 39–52 (2012)
134. Akbari, R., Mohammadi, A., Ziarati, K.: A novel bee swarm optimization algorithm for numerical function optimization, communications. Nonlinear Sci. Numer. Simul. **15**(9), 3142–3155 (2010)
135. Xinyi, L., Zunchao1, L., Liqiang, L.: An Artificial Bee Colony Algorithm for Multi-objective Optimization. In: IEEE Second International Conference on Intelligent System Design and Engineering Application, pp. 153–156 (2012)
136. Zhou, G., Wang, L., Xu, Y., Wang, S.: An effective artificial bee colony algorithm for multi-objective flexible job shop scheduling problem. In: Huang, D.-S., et al. (eds.) ICIC, pp. 1–8 (2012)
137. Wang, L., Zhou, G., Xu, Y., Liu, M.: An enhanced pareto-based artificial bee colony algorithm for the multi-objective flexible job-shop scheduling. Int. J. Adv. Manuf. Technol. **60**, 1111–1123 (2012)
138. Akay, B.: Synchronous and asynchronous pareto-based multi-objective artificial bee colony algorithms. J. Glob. Optim. **57**, 415–445 (2013)
139. Zhou, X., Shen, J., Li, Y.: Immune based chaotic artificial bee colony multiobjective optimization algorithm. In: Tan, Y., Shi, Y., Mo, H. (eds.) ICSI 2013, Part I. LNCS 7928, pp. 387–395 (2013)
140. Drechsler, N., Drechsler, R., Becker, B.: Multi-objective Optimization Based on Relation Favour. Lecture Notes in Computer Science. 1993: Evolutionary Multi-Criterion Optimization - EMO 2001, pp. 154–166. Springer, Berlin (2001)
141. Zitzler, E., Thiele, L.: Multi-objective evolutionary algorithms: a comparative case study and the strength Pareto approach. IEEE Trans. Evol. Comput. **3**(4), 257–271 (1999)
142. Ishibuchi, H., Nojima, Y.: Optimization of scalarizing functions through evolutionary multi-objective optimization. Lecture Notes in Computer Science **4403**, 51–67 (2007)
143. Julee, T., Park, S.M., Sim, K.B.: Electro encephalography signal grouping and feature classification using harmony Sarch for BCI. J. Appl. Math. **2013**(154539), 1–9 (2013)
144. Stastny, J., Covaka, P., Stancak, A.: EEG signal classification introduction to the problem. Radio Eng. **12**(3), 51–55 (2003)
145. Gonzalez, A.R.: EEG signal processing for BCI application: human computer system interaction background and applications. J. Adv. Intell. Soft Comput. **98**(1) (2012)
146. Teplan, M.: Fundamentals of EEG measurement. Meas. Sci. Rev. **2**(2), 1–11 (2002)
147. Garrett, D., Peterson, D.A., Anderson, C.W., Thaut, M.H.: Comparison of linear, non linear, and feature selection methods for EEG signal classification. IEEE Trans. Neural Syst. Rehabil. Eng. **11**(2) (2003)
148. Ahirwala, M.K., Kumar, A., Singh, G.K.: Adaptive Filtering of EEG/ERP Through Bounded Range Artificial Bee Colony (BR-ABC) algorithm. Journal of Digital Signal Processing **25**, 164–172 (2013)
149. Dries, J.E., Peterson, L.G.: Scaling ant colony optimization with hierarchical reinforcement learning partitioning. In: Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation, GECCO '08, pp. 25–32 (2008)

150. Bursa, M., Lhotska, L.: Modified ant colony clustering method in long term electrocardiogram processing. In: 29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society 2007 (EMBS- 2007), pp. 3249–3252 (2007)
151. Khushaba, R.N., Alsukker, A., Al-Ani, A., Al-jumaily, A.: Intelligent artificial ants based feature extraction from wavelet packet co-efficient for bio-medical signal classification. ISCCSP. Malta (2008)
152. Khushaba, R.N., Alsukker, A., Al-Ani, A., Al-jumaily, A.: A combined ant colony and differential evolution feature selection algorithm. In: Darigo, M., et al. (eds.) ANTS 2008. LNCS-5217, pp. 1–12 (2008)
153. Bursa, M., Lhotska, L.: Ant colony cooperative strategy in electrocardiogram and electroencephalogram data clustering. Stud. Comput. Intell. (SCI) **129**, 323–333 (2008)
154. Kennedy, J., Eberhart, R.C.: A discrete binary version of the particle swarm algorithm. In: IEEE International Conference on Systems, Man, and Cybernetics, pp. 4101–4108. IEEE Press, New York (1997)
155. Qiu, L., Li, Y., Yao, D.: A Feasibility Study of EEG Dipole Source Localization Using Particle Swarm Optimization. IEEE Congress on Evolutionary Computation, vol. 1, pp. 720–726 (2005)
156. Pulraj, M.P., Hema, C.R., Nagrajan, R., Yaacob, S., Adom, A.H.: EEG classification using radial basis PSO neural network for brain machine interfaces. In: The 5th Student Conference on Research and Development -SCOReD 2007. Malaysia (2007)
157. Nakamura, T., Tomita, Y., Ito, S., Mitsukura, Y.: A method of obtaining sense of touch by using EEG. 19th IEEE International Symposium on Robot and Human Interactive Communication Principe di Piemonte - Viareggio. Italy, Sept. 12–15 (2010)
158. Alp, Y., Arikan, O., Karakas, S.: Dipole source reconstruction of brain signals by using particle swarm optimization. In: IEEE 17th Conference on Signal Processing and Communication Applications, Antalya, pp. 9–12 (2009)
159. Satti, A.R., Coyle, D., Prasad, G.: Spatio-spectral & temporal parameter searching using class correlation analysis and particle swarm optimization for a brain computer interface. In: Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics. San Antonio (2009)
160. Lin, C.-J., Hsieh, M.-H.: Classification of mental task from EEG data using neural networks based on particle swarm optimization. J. Neurocomput. Elsevier (2009)
161. Ba-Karait, N.O.S., Shamsuddin, S.M., Sudirman, R.: Swarm negative selection algorithm for electroencephalogram signals classification. J. Comput. Sci. **5**(12), 995–1002 (2009)
162. Wei, Q., Wang, Y.: Binary multi-objective particle swarm optimization for channel selection in motor imagery based brain-computer interfaces. In: IEEE 4th International Conference on Biomedical Engineering and Informatics (BME I) (2011)
163. zbeyaz, A., Grsoy, M., oban, R.: Regularization and Kernel parameters optimization based on PSO algorithm in EEG signals classification with SVM. In: 2011 IEEE 19th Signal Processing and Communications Applications Conference (SIU 2011) (2011)
164. Kim, J.-Y., Park, S.-M., Ko, K.-E., Sim, K.-B.: A binary PSO-based optimal EEG channel selection method for a motor imagery based BCI system, ICHIT. Springer, Berlin (2012)
165. Sem a Arslan, Gulay T ezel, Ha kan Islk: EEG signals classification using a hybrid structure of ANN and PSO. Int. J. Future Comput. Commun. **1**(2) (2012)
166. Atyabi, A., Luerssen, M., Fitzgibbon, S.P., Powers, D.M.W.: Adapting subject-independent task-specific EEG feature masks using PSO. In: 2012 IEEE World Congress on Computational Intelligence (CEC), pp. 1–7. Brisbane (2012)
167. Shirvany, Y., Edelvik, F., Jakobsson, S., Hedstrm, A., Persson, M.: Application of particle swarm optimization in Epileptic spike EEG source localization. J. Appl. Soft Comput. **13**(5), 2515–2525 (2013)
168. Rakshit, P., Bhattacharyya, S., Konar, A., Khasnobish, A., Tibarewala, D.N., Janarthanan, R.: Artificial bee colony based feature selection for motor imagery EEG data. In: Bansal, J.C., et al. (eds.) Proceedings of Seventh International Conference on Bio-Inspired Computing: Theories and Applications (BIC-TA 2012), pp. 127–138 (2013)

169. Gandomi, A.H., Yang, X.S., Talatahari, S., Alavi, A.H.: Bat algorithm for constrained optimization tasks. Neural Comput. Appl. **22**(6), 1239–1255 (2013)
170. Dehuri, S., Cho, S.-B. (eds.): Knowledge Mining Using Intelligent Agents, Imperial College Press (2011)
171. Knowles, J., Corne, D.: Quantifying the effects of objective space dimension in evolutionary multi-objective optimization. In: Obayashi, S., Deb, K., Poloni, C., Hiroyasu, T., Murata, T. (eds.) Proceedings of Evolutionary Multi-Criterion Optimization, (EMO 2007). Lecture Notes in Computer Science, vol. 4403, pp. 757–771. Springer, Matshushima (2007)
172. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of IEEE International Conference on Neural Networks. Piscataway, pp. 1942–1948 (1995)
173. Zitzler, E., Thiele, L.: Multi-objective optimization using evolutionary algorithms - a comparative case study. Parallel Problem Solving from Nature - PPSNV. Lecture Notes in Computer Science 1498, pp. 292–301. Springer, Berlin (1998)
174. Knowles, J.D., Corne, D.W., Fleischer, M.: Bounded archiving using the Lebesgue measure, In: Proceedings of 2003 Congress on Evolutionary Computation, pp. 2490–2497. Canberra (2003)
175. Zitzler, E., Knzli, S.: Indicator-based selection in multi-objective search. Parallel Problem Solving from Nature - PPSN VIII. Lecture Notes in Computer Science 3242, pp. 832–842. Springer, Berlin (2004)
176. Emmerich, M., Beume, N., Naujoks, B.: An EMO algorithm using the hypervolume measure as selection criterion. Evolutionary Multi-Criterion Optimization -EMO 2005. Lecture Notes in Computer Science 3410, pp. 62–76. Springer, Berlin (2005)
177. Ibanez M.L., Stutzle, T.: The automatic design of multi-objective ant colony optimization algorithms. IEEE Trans. Evol. Comput. **16**(6) (2012)
178. Mariano, C.E., Morales, E.: A Multiple Objective Ant-Q Algorithm for the Design of Water Distribution Irrigation Networks, Technical Report HC-9904. Institute Mexicano de Tecnologa del Agua, Mexico (1999)
179. Stutzle, T.: MAXMIN Ant System for Quadratic Assignment Problems, Technical Report AIDA-97-04. FG Intellektik, FB Informatik, TU Darmstadt, Germany (1997)
180. Stutzle, T., Hoos, H.H.: MAXMIN ant system. Future Gener. Comput. Syst. J. **16**(8), 889–914 (2000)
181. Baran, B., Schaerer, M.: A multi-objective ant colony system for vehicle routing problem with time windows. In: Proceedings of the 21st IASTED International Conference, pp. 97–102 (2003)
182. Liao, C.J., Tseng, C.T., Luarn, P.: A discrete version of particle swarm optimization for flowshop scheduling problems. J. Comput. Oper. Res. **34**, 3099–3111 (2005)
183. Correa, E.S., Freitas, A.A., Johnson, C.G.: A new discrete particle swarm algorithm applied to attribute selection in a bioinformatics data set. In: Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation, pp. 35–42 (2006)
184. Wang, J.: A novel discrete particle swarm optimization based on estimation of distribution. In: International Proceedings on Intelligent Computing, pp. 791–802 (2007)
185. Jarboui, B., Damak, N., Siarry, P., Rebai, A.: A combinatorial particle swarm optimization for solving multi-mode resource-constrained project scheduling problems. J. Appl. Math. Comput. **195**, 299–308 (2007)
186. Zhen, Z., Wang, L., Huang, Z.: Probability-based binary particle swarm optimization algorithm and its application to WFGD control. Proceedings International Conference on Computer Science and Software Engineering **1**, 443–446 (2008)
187. Carlisle, A., Dozier, G.: Adapting particle swarm optimization to dynamic environments. In: Proceedings of International Conference on Artificial Intelligence, pp. 1958–1962 (2000)
188. Zhang, W., Liu, Y.: Adaptive particle swarm optimization for reactive power and voltage control in power systems. In: Proceedings of International Conference in Natural Computation. LNCS 3612, pp. 449–452 (2005)
189. Zhen, Z., Wang, Z., Liu, Y.: An adaptive particle swarm optimization for global optimization. In: Proceedings of Third International Conference on Natural Computation **4**, 8–12 (2007)

190. Chunxia, F., Youhong, W.: An adaptive simple particle swarm optimization algorithm. In: Proceedings of Chinese Control and Decision Conference International Conference on Computer Science and Software Engineering, pp. 3067–3072 (2008)
191. Zhan, Z.H., Zhang, J., Li, Y., Chung, H.S.H.: Adaptive particle swarm optimization. IEEE Trans. Syst. Man Cybern. **39**(6), 1362–1381 (2009)
192. Hongwu, L.: An adaptive chaotic particle swarm optimization. In: Proceedings of International Colloquium on Computing, Communication, Control, and Management **2**, 254–257 (2009)
193. Jian, L., Zhiming, L., Peng, C.: Solving constrained optimization via dual particle swarm optimization with stochastic ranking. In: Proceedings of International Conference on Computer Science and Software Engineering **1**, 1215–1218 (2008)
194. Li, J., Xiao, X.: Multi-swarm and multi-best particle swarm optimization algorithm. In: Proceedings of 7th World Congress on Intelligent Control and Automation, pp. 6281–6286 (2008)
195. Ling, C.A., Gen-ke, Y., Zhi-ming, W.: Hybrid discrete particle swarm optimization algorithm for capacitated vehicle routing problem. J. Zhejiang Univ. **7**(4), 607–614 (2005)
196. Pan, G., Duo, Q., Liu, X.: Performance of two improved particle swarm optimization in dynamic optimization environments. In: Proceedings of Sixth International Conference on Intelligent Systems Design and Applications, vol. 2, pp. 1024–1028 (2006)
197. Meissner, M., Schmuker, M., Schneider, G.: Optimized Particle Swarm Optimization (OPSP) and its Application to Artificial Neural Network Training. J. BMC Bioinform. **7** (2006)
198. Tian, D.P., Li, N.Q.: Fuzzy particle swarm optimization algorithm. In: Proceedings of International Joint Conference on Artificial Intelligence, pp. 263–267 (2009)
199. Lung, R. I., Dumitrescu, D.: Evolutionary swarm cooperative optimization in dynamic environments. Nat. Comput. (2009)
200. Mousa, A.A., El-Shorbagy, M.A., Abed-El-Wahed, W.F.: Local search based hybrid particle swarm optimization algorithm for multi-objective optimization. Swarm Evol. Comput. **3**, 1–14 (2012)
201. Dorigo, M., Maniezzo, V., Colorni, A.: The ant system : an autocatalytic optimizing process, Technical Report TR91-016, Politecnicodi Milano (1991)
202. Colorni, A., Dorigo, M., Maniezzo, V.: An investigation of some properties of an ant algorithm. In: Proceedings of the Parallel Problem Solving from Nature Conference (PPSN 92), pp. 509–520. Elsevier, Belgium (1992)
203. Bullnheimer, B., Hartl, R.F., Straub, C.: A new rank based version of the ant system-A computational study. Cent. Eur. J. Oper. Res. Econ. **7**, 25–38 (1997)
204. Cordon, O., Viana, I.F., Herrera, F., Moreno, L.: A new ACO model interesting evolutionary computation concepts : the best-worst ant system. In: Proceedings of ANTS 2000, pp. 22–29. IRIDIA (2000)

Comparison of Various Approaches in Multi-objective Particle Swarm Optimization (MOPSO): Empirical Study

Swagatika Devi, Alok Kumar Jagadev and Sachidananda Dehuri

Abstract This chapter presents a study of particle swarm optimization (PSO) method in multi-objective optimization problems. Many of these methods have focused on improving characteristics like convergence, diversity, and computational times by proposing effective 'archiving' and 'guide selection' techniques. What has still been lacking is an empirical study of these proposals in a common frame-work. In this chapter, an attempt to analyze these methods has been made; discussing their strengths and weaknesses. A multi-objective particle swarm optimization (MOPSO) algorithm, named dynamic multiple swarms in MOPSO is compared with other well known MOPSO techniques in which the number of swarms are adaptively adjusted throughout the search process via dynamic swarm strategy. The strategy allocates an appropriate number of swarms as required to support convergence and diversity criteria among the swarms. Additional novel designs include a PSO updating mechanism to better manage the communication within a swarm and among swarms and an objective space compression and expansion strategy to progressively exploit the objective space during the search process. Comparative study shows that the performance of the variant is competitive in comparison to the selected algorithms on standard benchmark problems. A dynamic MOPSO approach is compared and validated using several test functions and metrics taken from the standard literatures on evolutionary multi-objective optimization. Results indicate that the approach is highly competitive and that can be considered a viable alternative to solve multi-objective optimization problems.

S. Devi · A.K. Jagadev (✉)
Department of Computer Science and Engineering, Siksha 'O' Anusandhan University, Bhubaneswar 751030, India
e-mail: alok.jagadev@gmail.com

S. Devi
e-mail: sweetsweettalk@gmail.com

S. Dehuri
Department of Information & Communication Technology, Fakir Mohan University, Balasore 756019, India
e-mail: satchi.lapa@gmail.com

S. Dehuri et al. (eds.), *Multi-objective Swarm Intelligence*,
Studies in Computational Intelligence 592, DOI 10.1007/978-3-662-46309-3_3

Keywords Particle swarm optimization · MOPSO · Dynamic swarm strategy · Guide selection · Archiving

1 Introduction

Particle swarm optimization under the umbrella of swarm intelligence is a heuristic search technique, which is also considered as an evolutionary algorithm and is inspired by bird flocking and swarm behavior of birds, insects, and fish schools. It has been successfully used for single-objective optimization and shown to have fast convergence. The relative simplicity of PSO [1] and its population-based approach have made it a natural candidate to be extended for multi-objective optimization; such methods are usually referred as Multi-Objective Particle Swarm Optimization (MOPSO). One of the earliest proposals for MOPSO was made by Moore and Chapman in an unpublished manuscript in 1999. Since then there have been several promising MOPSO proposals flourished. In nature, certain bird species joined together in a flock to travel, to feed, and to collectively defend against any predators. Evidence indicates that the increase in feeding efficiency may be the key motivation of rendering the formation of mixed species flocks. The birds in different species collaborate and share information among each other if any food sources [2] are located. Different bird species may prefer different foods and acquire different foraging techniques. In addition, different species act as flock leaders [3–7] under various environments to lead and to influence the flocking behavior of a variety of bird species. The number of species in a flock may vary, depending upon the types of food sources available and the degree of competition among them. By analogy, the bird species join together in a flock to achieve certain foraging behaviors that will benefit each other, which is similar to the notion that multiple swarms in PSO explore the search space together to attain the objective of finding the optimal solutions, while different food preference in mixed species flocking corresponds to the tendency of multiple-swarm PSO in locating possible solutions in different regions in a fitness landscape. In addition, the fact that different species assume the leadership under various environments is analogous to the notion that multiple swarms in PSO select their global leaders that would lead and influence their movement toward the best solution found so far. The information shared within a species and among species is also closely portrayed in multi swarm PSO [8, 9] movement. Another feature, which we regard equally important, any MOPSO method is the way in which non-dominated solutions are retained over the iterations. Usual practice is to maintain an external archive, which is updated over iterations, to store the non-dominated solutions [10]. Often based on some criterion less crowded particles are retained. Also, sometimes relaxed forms of dominance have been employed to limit the archive size.

The rest of the chapter is organized as follows: Sect. 1 defines the general multi-objective problem domain. In Sects. 2 and 3, we describe the structure of general PSO and multi-objective particle swarm optimization employed in this study. Section 4 discusses different approaches borrowed from literature to conduct this study. Section 5

describes a particular variant of PSO i.e., DSMOPSO, its experimental settings, and is followed by results and discussions. Finally, the last section presents our conclusion and notes for future work.

2 General Multi-objective Problem

Multi-objective optimization is an area of multiple criteria decision making, which is concerned with mathematical optimization problems involving more than one objective function [11, 12] to be optimized simultaneously. Multi-objective optimization has been applied in many fields of science, including engineering, economics and logistics (see the section on applications for detailed examples) where optimal decisions need to be taken in the presence of trade-offs between two or more conflicting objectives. Minimizing cost while maximizing comfort while buying a car, and maximizing performance whilst minimizing fuel consumption and emission of pollutants of a vehicle are examples of multi-objective optimization problems involving two and three objectives, respectively. In practical problems, there can be more than three objectives. For a nontrivial multi-objective optimization problem, there does not exist a single solution that simultaneously optimizes each objective. In that case, the objective functions are said to be conflicting, and there exists a (possibly infinite number of) Pareto optimal solutions. A solution is called non dominated, Pareto optimal, Pareto efficient [13–15] or non inferior, if none of the objective functions can be improved in value without degrading some of the other objective values. Without additional subjective preference information, all Pareto optimal solutions are considered equally good. In mathematical terms, a multi-objective optimization problem can be formulated as $min(f_1(x), f_2(x), \ldots, f_k(x))$ s.t. $x \in X$, where the integer $k \geq 2$ is the number of objectives and the set X is the feasible set of decision vectors. The feasible set is typically defined by some constraint functions. In addition, the vector-valued objective function [16, 17] is often defined as $: X \to \mathbb{R}^k$, $f(x) = (f_1(x), f_2(x), \ldots, f_k(x))^T$. If some objective function is to be maximized, it is equivalent to minimize its negative. The image of X is denoted by $y \in \mathbb{R}^k$ an element $x^* \in \mathbb{R}^k$ is called a feasible solution or a feasible decision. A vector $z^* := f(x^*)\mathbb{R}^k$ for a feasible solution x^* is called an objective vector or an outcome. In multi-objective optimization, there does not typically exist a feasible solution [18, 19] that minimizes all objective functions simultaneously. Therefore, attention is paid to Pareto optimal solutions; that is, solutions that cannot be improved in any of the objectives without degrading at least one of the other objectives. In mathematical terms, a feasible solution $x^1 \in X$ is said to (Pareto) dominate another solution $x^2 \in X$, if

1. $f_1(x^1) \leq f_1(x^2) \, \forall \, i = 1, 2, \ldots, k$ and
2. $f_1(x^1) < f_1(x^2) \, \exists \, j = 1, 2, \ldots, k$.

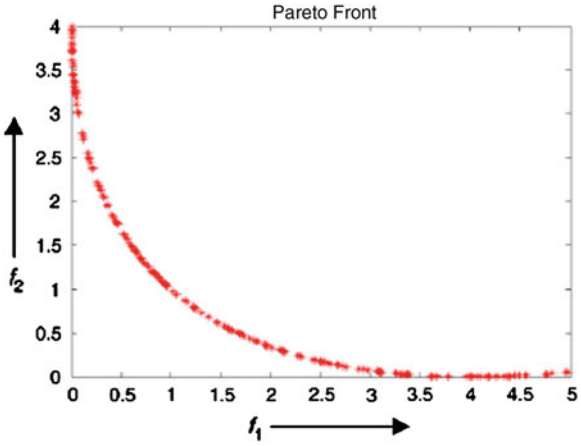


Fig. 1 Pareto front of simple multi-objective problem

A solution $x^1 \in X$ (and the corresponding outcome $f(x^*)$) is called Pareto optimal, if there does not exist another solution that dominates it. The set of Pareto optimal [20–22] outcomes is often called the Pareto front (in Fig. 1) or Pareto boundary.

The Pareto front [23–25] of a multi-objective optimization problem is bounded by a so-called nadir objective vector z^{nad} and an ideal objective vector z^{Ideal}, if these are finite. The nadir objective vector is defined in Eq. 1 where as the ideal objective vector is defined in Eq. 2.

$$z_i^{nad} = \sup_{x \in X in Pareto\, optimal} f_i(x) \, \forall \, i = 1, 2, \ldots, k \tag{1}$$

And the ideal objective vector as

$$z_i^{Ideal} = \inf_{x \in X} f_i(x) \, \forall \, i = 1, 2, \ldots, k \tag{2}$$

In other words, the components of a nadir and an ideal objective vector define upper and lower bounds for the objective function values of Pareto optimal solutions [26–31], respectively. In practice, the nadir objective vector can only be approximated as, typically, the whole Pareto optimal set is unknown. In addition, a utopian objective vector $z^{utopian}$ with $z_i^{utopian} = z_i^{ideal} - \varepsilon \, \forall \, i = 1, 2, \ldots, k$, where $\varepsilon > 0$ is small constant, is often defined because of numerical reasons.

3 Particle Swarm Optimization

PSO is an evolutionary computation technique motivated by the simulation of social behavior. Namely, each individual (agent) utilizes two important kinds of information in decision process. The first one is their own experience; that is, they have tried the

choices and know which state has been better so far, and they know how good it was. The second one is other agent's experiences; that is, they have knowledge of how the other agents around them have performed. Namely, they know which choices their neighbors have found are most positive so far and how positive the best pattern of choices was. In the PSO [32, 33] system, each agent makes his decision according to his own experiences and other agent's experiences. The system initially has a population of random solutions. Each potential solution, called a particle (agent), is given a random velocity and is flown through the problem space. The agents have memory and each agent keeps track of its previous (local) best position (called the P_{best}) and its corresponding fitness. There exist a number of P_{best} for the respective agents in the swarm and the agent with greatest fitness is called the global best (G_{best}) of the swarm. Each particle is treated as a point in an n-dimensional space. The ith particle is represented as $X_i = (x_{i1}, x_{i2}, \ldots, x_{in})$. The best previous position of the ith particle (P_{besti}) that gives the best fitness value is represented as $P_i = (p_{i1}, p_{i2}, \ldots, p_{in})$. The best particle among all the particles in the population is represented by $P_g = (p_{g1}, p_{g2}, \ldots, p_{gn})$. The velocity for particle i (i.e., the rate of the position change) is represented as $V_i = (v_{i1}, v_{i2}, \ldots, v_{in})$. The particles are manipulated according to the Eqs. 3 and 4 (the superscripts denote the iteration):

$$v_i^{k+1} = w v_i^k c_1 r_i (p_i - x_i^k) + c_2 r_2 (p_g - x_i^k) \tag{3}$$

$$x_i^{k+1} = x_i^k + v_i^{k+1} \tag{4}$$

where $i = 1, 2, \ldots, N$, and N is the size of the population; w is the inertia weight; c1 and c2 are two positive constants, called the cognitive and social parameter respec-

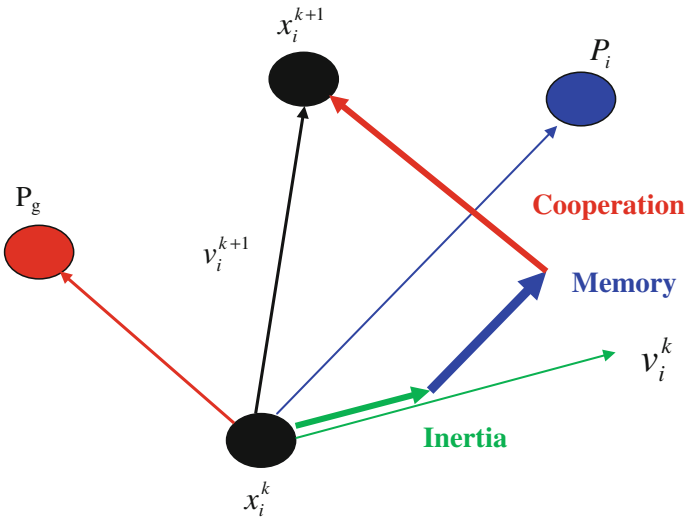


Fig. 2 Description of velocity and position updates in PSO

```
1: initializeSwarm()
2: locateLeader()
3: generation = 0
4: while generation < maxGenerations do
5:    for each particle do
6:       updatePosition()
7:       evaluation()
8:       updatePbest()
9:    end for
10:    updateLeader()
11:    generation ++
12: end while
```

Fig. 3 The pseudo code of the general PSO algorithm

tively; r1 and r2 are random numbers uniformly distributed within the range [0, 1]. Figure 2 shows the description of velocity and position updates of a particle for a two-dimensional parameter space. Figure 3 shows the pseudo code of the general PSO algorithm. We can make two main distinctions between PSO and an evolutionary algorithm [34, 35]:

1. Evolutionary algorithms rely on three mechanisms in their processing: parent representation, selection of individuals and the fine tuning of their parameters. In contrast, PSO only relies on two mechanisms, since PSO does not adopt an explicit selection function. The absence of a selection mechanism in PSO is compensated by the use of leaders [22, 36] to guide the search. However, there is no notion of offspring generation in PSO as with evolutionary algorithms.
2. A second difference between evolutionary algorithms and PSO [37–39] has to do with the way in which the individuals are manipulated. PSO uses an operator that sets the velocity of a particle to a particular direction. This can be seen as a directional mutation operator in which the direction is defined by both the particle's personal best and the global best (of the swarm). If the direction of the personal best is similar to the direction of the global best, the angle of potential directions will be small, whereas a larger angle will provide a larger range of exploration. In contrast, evolutionary algorithms use a mutation operator that can set an individual in any direction (although the relative probabilities for each direction may be different). In fact, the limitations exhibited by the directional mutation of PSO have led to the use of mutation operators similar to those adopted in evolutionary algorithms. Two are the key aspects by which we believe that PSO [40, 41] has become so popular:
    i. The main algorithm of PSO is relatively simple (since in its original version, it only adopts one operator for creating new solutions, unlike most evolutionary algorithms) and its implementation is, therefore, straightforward.
    ii. PSO has been found to be very effective in a wide variety of applications, being able to produce very good results at a very low computational cost.

In order to establish a common terminology, in the following we provide some definitions of several technical terms commonly used:
Swarm: Population of the algorithm.
Particle: Member (individual) of the swarm. Each particle [42] represents a potential solution to the problem being solved. The position of a particle is determined by the solution it currently represents.

- **pbest** (personal best): Personal best position of a given particle, so far. That is, the position of the particle that has provided the greatest success (measured in terms of a scalar value analogous to the fitness adopted in evolutionary algorithms).
- **lbest** (local best): Position of the best particle member of the neighborhood of a given particle.
- **gbest** (global best): Position of the best particle of the entire swarm.

Leader: Particle that is used to guide another particle towards better regions of the search space.
Velocity (vector): This vector drives the optimization process, that is, it determines the direction in which a particle needs to "fly" (move), in order to improve its current position. Inertia weight: Denoted by w, the inertia weight is employed to control the impact of the previous history of velocities on the current velocity of a given particle.
Learning factor: Represents the attraction that a particle has toward either its own success or that of its neighbors. Two are the learning factors used: c1 and c2. c1 is the cognitive learning factor and represents the attraction that a particle has toward its own success. c2 is the social learning factor and represents the attraction that a particle has toward the success of its neighbors. Both, c1 and c2, are usually defined as constants.
Neighborhood topology: Determines the set of particles that contribute to the calculation of the lbest value of a given particle.

4 PSO for Multiple Objectives

In a large number of design applications there are a number of competing quantitative measures that define the quality of a solution. For instance, designing aircrafts requires simultaneous optimization of fuel efficiency, payload, and weight. Similarly for bridge construction a good design characterized by low total mass and high stiffness. These objectives cannot be typically met by a single solution, so, by adjusting the various design parameters, the firm may seek to discover what possible combinations of these objectives are available, given a set of constraints. The relative simplicity of PSO [18, 43–49] and its population based approach have made it a natural candidate to be extended for multi-objective optimization. Many different strategies for solving multi-objective problems using PSO has been published since 2002. However, although most of these studies were generated in tandem, each of these studies implements MOPSO in a different fashion. Given the wealth of multi-objective evolutionary algorithms (MOEAs) [50–52] in the literature this may not

seem particularly surprising, however the PSO heuristics puts a number of constraints on MOPSO that MOEAs are not subject to. In PSO itself the swarm population is fixed in size, and its member cannot be replaced, only adjusted their p_{best} and g_{best}, which are themselves easy to define. However, in order to facilitate an MO approach to PSO a set of non-dominated solutions (the best individuals found so far using the search process) must replace the single global best individual in the standard uniobjective PSO case. In addition, there may be no single previous best individual for each member of the swarm. Choosing both which gbest and pbest to direct a swarm member's flight therefore is not trivial in MOPSO. In general, when solving a multi-objective problem using evolutionary or non-evolutionary techniques [53–55], the following three main goals need to be achieved: (i) maximize the number of elements of the Pareto optimal set found; (ii) minimize the distance of the Pareto front produced by the algorithm with respect to the true (global) Pareto front (assuming we know its location); (iii) maximize the spreads of solutions found, so that we can have a distributions of vectors as smooth and uniform as possible. Based on the population nature of PSO, it is desirable to produce several (different) non-dominated solutions with a single run. So, as with any other evolutionary algorithm, the three main issues to be considered when using PSO to multi-objective optimization [56–58] are: (i) how to select g_{best} particles in order to give preference to non-dominated solutions over those that are dominated? (ii) How to retain the non-dominated solutions found during the search process in order to report solutions that are non-dominated with respect to all the past populations and not only with respect to the current one? Also it is desirable that these solutions are well spread along the Pareto front [59–61]; (iii) how to maintain diversity in the swarm in order to avoid convergence to a single solution? As we could see in the previous section, when solving single-objective optimization problems, the gbest that each particle uses to update its position is completely determined once a neighborhood topology is established. However in the case of multi-objective optimizations [62–64] problems, each particle might have a set of different gbests from which just one can be selected in order to update its position. Such set of gbests is usually stored in a different place from the swarm that we will call external archive denoted as EX_ARCHIVE. This is a repository in which the non-dominated solutions found so far are stored. The solutions contained in the external archive are used as global bests when the positions of the particles of the swarm have to be updated. Furthermore, the contents of the external archive are also usually reported as the final output of the algorithm. The following algorithm describes how a general MOPSO [65–67] works.

First the swarm is initialized. Then a set of gbests is also initialized with the non-dominated particles from the swarm. As we mentioned before, the set of gbests is usually stored in an external archive, which we call EX_ARCHIVE [68].

Later on, some sort of quality measure is calculated for all the gbests [69] in order to select (usually) one gbest for each particle of the swarm. At each generation, for each particle, a leader is selected and the flight is performed. Most of the existing MOPSOs apply some sort of mutation operator after performing the flight. Then the

```
Begin
Parameters initialization for cell-based rank density estimation scheme, crowdedness
indicator, age indicator, and objective space compression and expansion strategy.

/*Initialization
Set swarm size
Randomly generate one swarm
Set Maximum iterations (tmax)
Set iteration t = 0

Fitness Evaluate for all particles
Cell-based_rank_density_estimation_scheme()
Identify_swarm_leaders
Update_swarm_local_best_archive
Archive_maintenance
t = 1
While t < tmax
    Objective_space_compression_expansion_strategy()
    Swarm_growing_strategy()
    Swarm_declining_strategy()

    For each swarm (%Exclude newly generated swarms)
       For each swarm member
                For each dimension
                       Flight()
                EndFor
      EndFor
    EndFor

    Fitness Evaluate for all particles
    Cell-based_rank_density_estimation_scheme()
    Identify_swarm_leaders
    Update_swarm_local_best_archive
    Archive_maintenance
    t = t + 1
EndWhile
Report nondominated solutions stored in archive
End
```

particle is evaluated and its corresponding pbest is updated. A new particle replaces its pbest particle usually when this particle is dominated or if both are incomparable (i.e. they are both non-dominated with respect to each other). After all the particles have been updated, the set of gbests is updated, too. Finally, the quality measure of the set of gbests is recalculated. This process is repeated for a certain number of iterations.

5 Approaches of MOPSO

In this section, we review the state-of-the-art literature on different approaches of multi-objective PSO algorithms. We will distinguish two fundamental categories of algorithms, based on the two approaches mentioned namely, approaches that exploit each objective function separately, and Pareto-based [70–72] schemes. The distinction is made mainly for presentation purposes, and it is not strict, since there are algorithms that combine features from both approaches.

5.1 Algorithms That Exploit Each Objective Function Separately

This category consists of approaches that either combine all objective functions to a single one or consider each objective function in turn for the evaluation of the swarm, in an attempt to exploit the efficiency of PSO in solving single-objective problems. This approach has the advantage of straightforward update of the swarm and best positions, with an external archive usually employed for storing non- dominated solutions. The main drawback of these methods is the lack of a priori information regarding the most proper manipulation of the distinct objective values, in order to converge to the actual Pareto front. In computational complexity and optimization the no free lunch theorem is a result that states that for certain types of mathematical problems, the computational cost of finding a solution, averaged over all problems in the class, is the same for any solution method. No solution therefore offers a 'short cut'. In computing, there are circumstances in which the outputs of all procedures solving a particular type of problem are statistically identical. In the "no free lunch" metaphor, each "restaurant" (problem-solving procedure) has a "menu" associating each "lunch plate" (problem) with a "price" (the performance of the procedure in solving the problem). The menus of restaurants are identical except in one regard—the prices are shuffled from one restaurant to the next. For an omnivore who is as likely to order each plate as any other, the average cost of lunch does not depend on the choice of restaurant. But a vegan who goes to lunch regularly with a carnivore who seeks economy pays a high average cost for lunch. To methodically reduce the average cost, one must use advance knowledge of (a) what one will order and (b) what the order will cost at various restaurants. That is, improvement of performance in problem-solving hinges on using prior information to match procedures to problems.

In formal terms, there is no free lunch when the probability distribution on problem instances is such that all problem solvers have identically distributed results. In the case of search, a problem instance is an objective function [73], and a result is a sequence of values obtained in evaluation of candidate solutions in the domain of the function. For typical interpretations of results, search is an optimization process [74]. There is no free lunch in search if and only if the distribution on objective functions is invariant under permutation of the space of candidate solutions. This condition does not hold precisely in practice, but an "(almost) no free lunch" theorem suggests that it holds approximately.

5.2 Objective Function Aggregation Approaches

These approaches aggregate, through a weighted combination, all objective functions in a single one, (Eq. 5)

$$F(x) = \sum_{i=1}^{k} w_i f_i(x) \tag{5}$$

where w_i are nonnegative weights such that $\sum_{i=1}^{k} w_i = 1$

And the optimization is performed on F(x), similarly to the single-objective case. Two of its variants are discussed below. The Weighted Aggregation is the most common approach for coping with MO problems. According to this approach, all the objectives are summed to a weighted combination $F = \sum_{i=1}^{k} w_i f_i(x)$, where $w_i, i = 1, 2, \ldots, k$, are non-negative weights. It is usually assumed that $\sum_{i=1}^{k} w_i = 1$. These weights can be either fixed or dynamically adapted during the optimization. If the weights are fixed then we are in the case of the Conventional Weighted Aggregation (CWA). Using this approach only a single Pareto Optimal point [75] can be obtained per optimization run and a priori knowledge of the search space is required in order to choose the appropriate weights [40]. Thus, the search has to be repeated several times to obtain a desired number of Pareto Optimal points. Yet, this is not suitable in most problems due to time limitations and heavy computational costs. Moreover, CWA is unable to detect solutions in concave regions of the Pareto Front [40]. Other Weighted Aggregation approaches have been proposed to alleviate the limitations of the CWA. For a two-objective MO problem, the weights can be modified during the optimization, according to Eq. 6,

$$w_1(t) = sign\left(sin\left({}^{2\pi t}/_{F}\right)\right), \qquad w_2(t) = 1 - w_1(t) \tag{6}$$

where t is the iteration's index and F is the weights' change frequency. This is the well-known Bang-Bang Weighted Aggregation (BWA) approach, according to which, the weights are changed abruptly due to the usage of the sign (−) function. Alternatively, the weights can be gradually modified according to Eq. 7,

$$w_1(t) = \left|sin\left({}^{2\pi t}/_{F}\right)\right|, \qquad w_2(t) = 1 - w_1(t) \tag{7}$$

This is called Dynamic Weighted Aggregation (DWA). The slow change of the weights forces the optimizer to keep moving on the Pareto Front, if it is convex [76, 77], performing better than in the BWA case. If the Pareto Front is concave then the performance using DWA and BWA is almost identical when Genetic Algorithms are used [40]. Function F1 (convex, uniform Pareto Front):

$$f_1 = 1/n \sum_{i=1}^{n} x_i^2, \qquad f_2 = 1/n \sum_{i=1}^{n} (x_i - 2)^2$$

Function F2 (convex, non-uniform Pareto Front):

$$f_1 = x_1, \quad g = 1 + \frac{g}{(n-1)} \sum_{i=2}^{n} x_i, \quad f_2 = g\left(1 - \sqrt{f_1/g}\right)$$

5.3 Non-Pareto, Vector Evaluated Approaches

This is a multi swarm approach based on the idea of Vector Evaluated Genetic Algorithm (VEGA). In VEPSO [71, 78], there is one swarm devoted to each objective function, and evaluated only for this objective function. However, in the swarm update (the algorithm employed the global variant of PSO), the best positions of one swarm are used for the velocity update of another swarm that corresponds to a different objective function.

Thus, if the problem consists of k objectives, then k swarms are used. If $v_i^{[s]}$ denotes the velocity of the ith particle in the sth swarm, $s = 1, \ldots, k$, then it is updated based on Eq. 8

$$v_{ij}^{[s]}(t+1) = wv_{ij}^{[s]}(t) + c_1 r_1 (p_{ij}^{[s]}(t) - x_{ij}^{[s]}(t)) + c_2 r_2 (p_{gj}^{[q]}(t) - x_{ij}^{[s]}(t)) \quad (8)$$

where $p_g^{[q]}$ is the best position of the qth swarm (which is evaluated with the qth objective function). In this way, the information regarding the promising regions for one objective is inoculated to a swarm that already possesses information for a different objective. In this implementation, each swarm is assigned to a processor and the number of swarms is not necessarily equal to the number of objective functions. The communication among swarms is performed through an island migration scheme, similar to the ring topology used in PSO's ring neighborhood topology.

5.4 Pareto Dominance Approach

These approaches use the concept of Pareto dominance to determine the best positions (leaders) [79] that will guide the swarm during search. In MOPSO, the non-dominated solutions detected by the particles are stored in a repository. Also, the search space is divided in hyper cubes. Each hypercube is assigned a fitness value that is inversely proportional to the number of particles it contains. Then, the classical roulette wheel selection is used to select a hypercube and a leader from it. Thus, the velocity update (Eq. 9) for the ith particle becomes

$$v_{ij}(t+1) = wv_{ij}(t) + c_1 r_1 (p_{ij}(t) - x_{ij}(t)) + c_2 r_2 (R_h(t) - x_{ij}(t)) \quad (9)$$

where p_i is its best position and R_h is the selected leader from the repository. The best position pi is updated at each iteration, based on the domination relation between the existing best position of the particle and its new position. Also, the repository has limited size and, if it is full, new solutions are inserted based on the retention criterion, that is, giving priority to solutions located in less crowded areas of the objective space. More specifically, leaders are generated through a multilevel sieve procedure that ranks individuals. Initially, all non-dominated particles [80–82] are assigned a rank of 1 and removed from swarm. The non-dominated solutions from the remaining particles are assigned a rank of 2, and the procedure continues until all particles have been assigned a rank. If at most half of the swarm has been assigned a rank of 1, then all particles with rank smaller than the average rank are assigned to the set of leaders. Otherwise, only particles with a rank of 1 are assigned to the set of leaders. For the rest of the particles, a leader is selected and used for updating their position. The selection of leader is based on the computation of the crowding radius for each leader and a roulette wheel selection mechanism that uses these values. Leaders with higher crowding [9] radius have higher selection probability and, therefore, promote the uniform spread of solutions on the Pareto front. Special care is taken in constrained problems, where ranking [83] takes into consideration both the objective and constraint values.

5.5 Weighted Sum Approach

Under this category we consider approaches that combine (or .aggregate.) all the objectives of the problem into a single one. In other words, the multi-objective problem is transformed into a single-objective one. Generally there are three types of aggregating functions: (1) a conventional linear aggregating function (where weights are fixed during the run), (2) a dynamic aggregating function (where weights are gradually modified during the run) and (3) the bang bang weighted aggregation approach (were weights are abruptly modified during the run). In decision theory, the weighted sum model (WSM) is the best known and simplest multi criteria decision analysis (MCDA)/multi-criteria decision making method for evaluating a number of alternatives in terms of a number of decision criteria. It is very important to state here that it is applicable only when all the data are expressed in exactly the same unit. If this is not the case, then the final result is equivalent to "adding apples and oranges."

In general, suppose that a given MCDA problem is defined on m alternatives and n decision criteria. Furthermore, let us assume that all the criteria are benefit criteria, that is, the higher the values are, the better it is. Next suppose that w_j denotes the relative weight of importance of the criterion C_j and a_{ij} is the performance value of alternative A_i when it is evaluated in terms of criterion C_j. Then, the total (i.e., when all the criteria are considered simultaneously) importance of alternative A_i, denoted as $A_i^{WSM-score}$, is defined in Eq. 10:

Table 1 Decision matrix

	C_1	C_2	C_3	C_4
Alts.	0.20	0.15	0.40	0.25
A_1	25	20	15	30
A_2	10	30	20	30
A_3	30	10	30	10

$$A_i^{WSM-score} = \sum_{j=1}^{n} w_j a_{ij}, \text{ for } i = 1, 2, 3, \ldots, m. \tag{10}$$

For the maximization case, the best alternative is the one that yields the maximum total performance value [84–86]. For a simple numerical example suppose that a decision problem of this type is defined on three alternatives A_1, A_2, A_3 each described in terms of four criteria C_1, C_2, C_3 and C_4. Furthermore, let the numerical data for this problem be as in the following decision matrix (Table 1).

For instance, the relative weight of the first criterion is equal to 0.20, the relative weight for the second criterion is 0.15 and so on. Similarly, the value of the first alternative (i.e., A_1) in terms of the first criterion is equal to 25, the value of the same alternative in terms of the second criterion is equal to 20 and so on. When the previous formula is applied on these numerical data the WSM scores for the three alternatives are:

$$A_1^{WSM-score} = 25 \times 0.20 + 20 \times 0.15 + 15 \times 0.40 + 30 \times 0.25 = 21.50$$

Similarly, one gets:

$$A_2^{WSM-score} = 22.0, \text{ and } A_2^{WSM-score} = 22.0$$

Thus, the best alternative (in the maximization case) is alternative A_2 (because it has the maximum WSM score which is equal to 22.00). Furthermore, these numerical results imply the following ranking of these three alternatives: $A_2 = A_2 > A_1$.

5.6 Lexicographic Approach

In this method, the user is asked to rank the objectives in order of importance. The optimum solution is then obtained by minimizing the objective functions separately, starting with the most important one and proceeds according to the assigned order of importance of the objectives. Lexicographic ordering [7] tends to be useful only when few objective functions [84, 86, 87] are used (two or three), and it may be sensitive to the ordering of the objectives. Another way to tackle multiple objectives is by lexicographic ordering, a technique that requires the decision-maker to establish a

priority for each objective [37]. Then, two solutions are compared with respect to the most important objective. If this results in a tie, the algorithm proceeds to compare the solutions but now with respect to the second most important objective. This tie-breaking process is repeated until no objectives are left to be examined, in which case, both solutions have the same objective values. Formally, we can define the ith problem as

$$\begin{aligned} &\textit{minimize } f_i(x) \\ &\text{subject to:} \\ &\quad g_j(x) \leq 0, j = 1, 2, \ldots, m \\ &\quad f_l(x) = f_{l*}, l = 1, 2, \ldots, i-1 \end{aligned}$$

The lexicographic approach is usually useful when dealing with few objectives (two or three). It should also be noted that sometimes its performance is tightly subject to the ordering given by the set priorities.

5.7 Subpopulation Approach

These approaches involve the use of several subpopulations as single-objective [88] optimizers. Then, the subpopulations somehow exchange information or recombine among themselves aiming to produce tradeoffs among the different solutions previously generated for the objectives that were separately optimized.

5.8 Pareto Based Approach

These approaches use leader selection techniques based on Pareto dominance [50, 89]. The basic idea of all the approaches considered here is to select as leaders to the particles that are non-dominated [2, 4–6, 90, 91] with respect to the swarm. Note however, that several variations of the leader selection scheme are possible since most authors adopt additional information to select leaders (e.g., information provided by a density estimator) in order to avoid a random selection of a leader from the current set of non dominated solutions. The selection of the gbest for a particle in the swarm is based on the structure defined by the dominated tree. First, a composite point of the tree is located based on dominance relations, and then the closest member (in objective function space) of the composite point is chosen as the leader. On the other hand, a set of personal best particles found (non-dominated) is also maintained for each swarm member, and the selection is performed uniformly. This approach also uses a turbulence operator that is basically a mutation operator that acts on the velocity value used by the PSO algorithm [1, 92]. This proposal uses two external archives: one for storing the leaders currently used for performing the flight and another for storing the final solutions. The density estimator factor is used to filter out the list of leaders whenever the maximum limit imposed on such list

is exceeded. In a multi-objective optimization problem we seek to simultaneously extremise D objectives: $y_i = f(x_i)$, where $i = 1, \ldots, D$ and where each objective depends upon a vector x of K parameters or decision variables. The parameters may also be subject to the J constraints: $e_j(x) \geq 0$ for $j = 1, \ldots, J$. Without loss of generality it is assumed that these objectives are to be minimized, as such the problem can be stated as:

$$\begin{aligned} &\text{minimize } y = f(x) \cong (f_1(x), f_2(x), \ldots, f_D(x)) \\ &\text{subject to,} \\ &\qquad e(x) \cong (e_1(x), e_2(x), \ldots, e_j(x)) \geq 0 \end{aligned}$$

A decision vector u is said to strictly dominate another v (denoted u $\leq$ v) if $f_i(u) \leq f_i(v) \, \forall \, i = 1, 2, \ldots, D$ and $f_i(u) < f_i(v)$ for some i; less stringently u weakly dominates v (denoted u $\leq$ v) if $f_i(u) \leq f_i(v) \, \forall \, i$. A set of decision vectors is said to be a non-dominated set if no member of the set is dominated by any other member. The true Pareto front [93, 94], P, is the non-dominated set of solutions which are not dominated by any feasible solution.

6 A Variant of MOPSO

A multiple-swarm multi-objective particle swarm optimization (PSO) algorithm, named dynamic multiple swarms in multi-objective PSO, is discussed in which the number of swarms is adaptively adjusted throughout the search process via dynamic swarm strategy. The strategy allocates an appropriate number of swarms as required to support convergence and diversity [8, 95] criteria among the swarms.

The main objective of this is to develop a multiple-swarm MOPSO [96, 97] that eliminates the need to estimate an initial number of swarms to improve the computational efficiency without compromising the performance of the algorithm. Once the swarm template (x_{new}) [98] is generated, the template is perturbed to generate a swarm of particles. In order for perturbation to happen, the perturbation region centered around x_{new} needs to be defined. In addition to x_{new}, a particle from every swarm is randomly selected. For example, Fig. 4 shows eight randomly selected particles from eight different swarms in addition to x_{new}. The black circle in Fig. 4 represents the coordinate of x_{new}, i.e., (x_{new}, ja, x_{new}, jb). Since there are more than two corners around the coordinate of xnew (i.e., represented by "x" symbol and labeled as Z_1 to Z_6 in Fig. 4), a corner is randomly selected. In Fig. 4, the selected corner is Z_2 and denoted as v_{cell}. The distance between the center and Z_2, i.e., Δd, is computed to form the perturbation region of x_{new}. The proposed algorithm, DSMOPSO, involves two key strategies: a swarm [99] growing strategy to allocate more swarms if necessary and justified, and a swarm declining strategy to eliminate swarms that fail to contribute in search of a Pareto front. Additional designs are included to support the aforementioned two strategies. These designs include the following:

1. cell-based rank density estimation scheme [100–102] to keep track of the rank and density values of the particles;

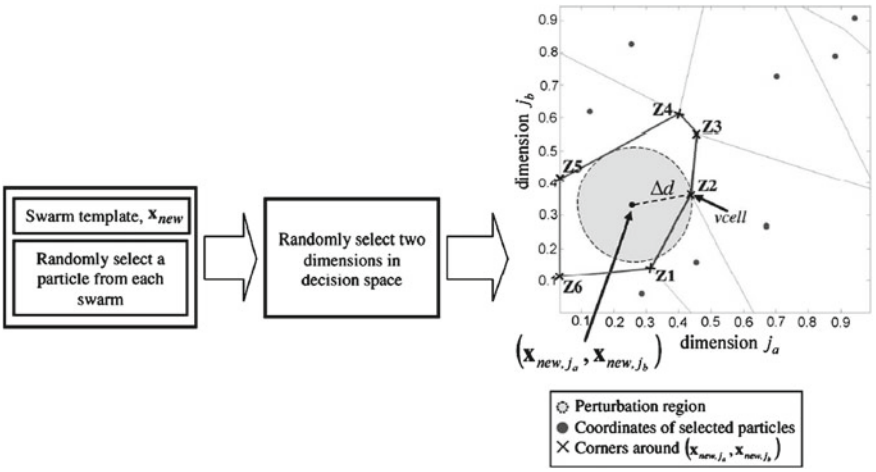


Fig. 4 Generation of swarm template x_{new}

2. Objective space compression and expansion strategy to adjust the size of the objective space whenever needed to progressively search for high-precision true Pareto front;
3. PSO updating equation modified to exploit its usefulness and to accommodate the multiple-swarm concept.
4. swarm local best archive updated based on the progression of its swarm representatives, the swarm leaders. Figure 5 shows the generic steps of DSMOPSO [103].

In DSMOPSO [103–105], adding and removing the swarms throughout the search process will directly affect the swarm population distribution. Instead of applying the Pareto ranking method to update the Pareto rank of the particles and applying niching strategy [102, 106] to estimate the density of the particles when the swarm population progresses at every iteration.

1. If the swarm local best archive is empty or the reinitialized parameter is triggered, record rank values of all swarm leaders, including their corresponding positions and the positions of their respective swarm members.
2. If the swarm local best archive is nonempty, the rank values of the swarm leaders in the current iteration are compared with those recorded in the swarm local best archive. Any of the current swarm leaders [107, 108] that have lesser rank values are identified; their rank values, their positions, and the positions of their corresponding swarm members will replace the recorded ones. If the rank values of a current swarm and its recorded swarm leader have the same rank value, then pure Pareto ranking method is applied to both of the swarm leaders. If the current swarm dominates the recorded swarm leader, then the current one will replace the recorded one. If both do not dominate each other, one of them is randomly chosen to update the swarm local best archive. The new velocity and position are given in Eq. 11,

```
Begin
Parameters initialization for cell-based rank density estimation scheme, crowdedness
indicator, age indicator, and objective space compression and expansion strategy.

/*Initialization
Set swarm size
Randomly generate one swarm
Set Maximum iterations (tmax)
Set iteration t = 0

Fitness Evaluate for all particles
Cell-based_rank_density_estimation_scheme()
Identify_swarm_leaders
Update_swarm_local_best_archive
Archive_maintenance
t = 1
While t < tmax
    Objective_space_compression_expansion_strategy()
    Swarm_growing_strategy()
    Swarm_declining_strategy()

    For each swarm (%Exclude newly generated swarms)
      For each swarm member
              For each dimension
                    Flight()
              EndFor
      EndFor
    EndFor

    Fitness Evaluate for all particles
    Cell-based_rank_density_estimation_scheme()
    Identify_swarm_leaders
    Update_swarm_local_best_archive
    Archive_maintenance
    t = t+1
EndWhile
Report nondominated solutions stored in archive
End
```

Fig. 5 Pseudo code of DSMOPSO

$$
\begin{aligned}
v_{i,j}^{n}(t+1) &= w \times v_{i,j}^{n}(t) + c_1 \times r_1 \times \left(P_{i,j}^{n}(t) - x_{i,j}^{n}(t)\right) \\
&\quad + c_2 \times r_2 \times \left(P_{g,j}(t) - x_{i,j}^{n}(t)\right) + c_3 \times (1 - r_2) \\
&\quad \times \left[r_3 \times \left(P_{L,j}^{n}(t) - x_{i,j}^{n}(t)\right) + (1 - r_3) \times \left(P_{L,j}^{\tilde{n}}(t) - x_{i,j}^{n}(t)\right)\right] \\
x_{i,j}^{n}(t+1) &= x_{i,j}^{n}(t) + v_{i,j}^{n}(t+1)
\end{aligned} \tag{11}
$$

where $v_{i,j}^{n}(t)$ is the jth dimensional velocity of swarm member i of swarm n in iteration t; $x_{i,j}^{n}(t)$ is the jth dimensional position of swarm member i of swarm n in iteration t; $P_{i,j}^{n}(t)$ denotes the jth dimensional personal best (p_{best}) position of the swarm members i of swarm n in iteration t; $P_{g,j}^{n}(t)$ is the jth dimensional global best (g_{best}) selected from archive in iteration t; $P_{L,j}^{n}(t)$ is the jth dimensional local best position of swarm leader of swarm n in iteration t; $P_{L,j}^{\tilde{n}}(t)$ (with a superscript $\tilde{n}$) is the jth dimensional local best position of any swarm leaders other than their own swarm leader in iteration t; r_1, r_2, and r_3 are random numbers within [0, 1] that

are regenerated every time they occur; w is the inertial weight, varied between 0.1 and 0.5 at every iteration; and c_1, c_2, and c_3 are the acceleration constants. Each of the acceleration constants is randomly varied between 1.5 and 2 to provide different emphasis on the components. As shown in Fig. 6a, the gray cells have a rank value [89, 109] of one. The swarm leaders G and H are located in a gray cell. After the objective space is compressed, the location of gray cells is updated, and only swarm leader G now resides in a gray cell, while the rank value of swarm leader H is two (Fig. 6b). This will encourage swarm leader H to move toward the true Pareto front [110]. When the objective space compression strategy is applied several times at early iterations, there is a possibility that the objective space is overly compressed and can cause the boundaries of the objective to not cover the true Pareto front (Fig. 7a, b).

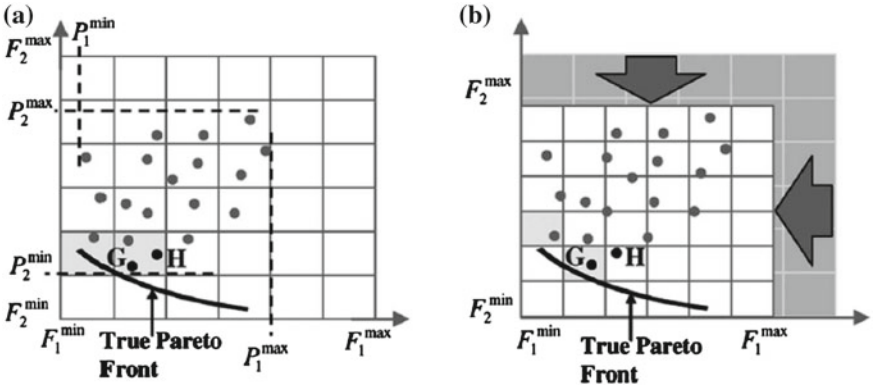


Fig. 6 Illustration of objective space compression strategy (*arrows* in (**b**) signify the objective space is compressed)

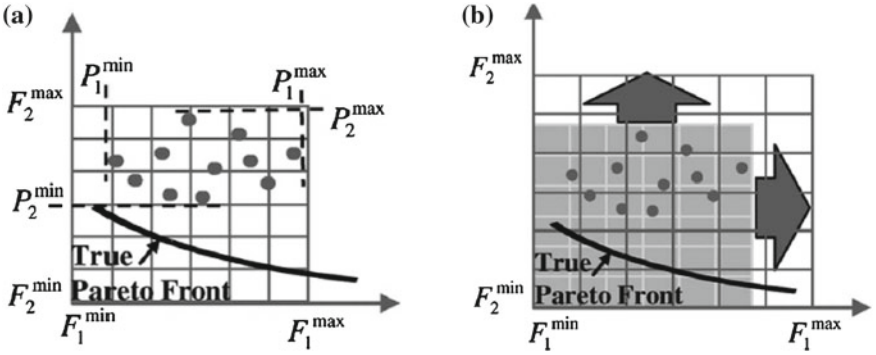


Fig. 7 Illustration of objective space expansion strategy (*arrows* in (**b**) signify the objective space that is expanded)

7 Comparative Study

To evaluate the performance and computational cost of DSMOPSO with selected MOPSOs [111–113], standard benchmark test problems are used: ZDT1, ZDT2, ZDT3, ZDT4, and ZDT6 and the parameters are set to some appropriate values as in Table 2. ZDT1 has convex Pareto fronts, while ZDT2 has non convex Pareto fronts. ZDT1 and ZDT2 challenge the algorithms' ability to find and produce a quality spread of Pareto front. ZDT3 possesses a non convex [114] and disconnected Pareto front. It exploits the algorithms' ability to search for all of the disconnected regions

Table 2 Parameter configurations for selected MOPSOs

	Parameters setting for MOPSO
cMOPSO [44]	No. of swarms = 4; Internal iterations; stmax = 5; Population size = 40; Archive size = Not fixed
	No. of iterations = *150
MOPSO [95]	50 divisions adaptive grid; Mutation probability = 0.5; Population size = 100; Archive size = 100
	No. of iterations = 300
DMOPSO [45]	*Test Function ZDT1*
	No. of swarms = 4; Swarm size = 5; Ki = 50, i = 1,2; Archive size = Not fixed; stmax = 5; unp = 3
	lnp = 1; $\alpha = \beta = 0.5$; rud = 0.7; rld = 0.02; ppv = 10; S = 0.02; and Ka = 10
	Test Function ZDT2
	No. of swarms = 2; Swarm size = 20; Ki = 50, i = 1,2; Archive size = Not fixed; stmax = 5; unp = 3
	lnp = 1; $\alpha = \beta = 0.5$; rud = 0.7; rld = 0.02; ppv = 10; S = 0.02; and Ka = 40
	Test Function ZDT3
	No. of swarms = 3; Swarm size = 6; Ki = 50, i = 1,2; Archive size = Not fixed; stmax = 5; unp = 3
	lnp = 1; $\alpha = \beta = 0.5$; rud = 0.7; rld = 0.02; ppv = 10; S = 0.02; and Ka = 15
	Test Function ZDT4
	No. of swarms = 2; Swarm size = 20; Ki = 50, i = 1,2; Archive size = Not fixed; stmax = 5; unp = 5
	lnp = 1; $\alpha = \beta = 0.5$; rud = 0.9; rld = 0.02; ppv = 10; S = 0.02; and Ka = 40
	Test Function ZDT6
	No. of swarms = 4; Swarm size = 5; Ki = 50, i = 1,2; Archive size = Not fixed; stmax = 5; unp = 3
	lnp = 1; $\alpha = \beta = 0.5$; rud = 0.9; rld = 0.02; ppv = 10; S = 0.02; and Ka = 40
	Test Function DTLZ2
	No. of swarms = 4; Swarm size = 5; Ki = 50, i =1,2; Archive size = Not fixed; stmax = 5; unp = 3
	lnp = 1; $\alpha = \beta = 0.5$; rud = 0.7; rld = 0.02; ppv = 10; S = 0.02; and Ka = 10
*DSMOPSO	Swarm size = 6; Ki = 6, i =1,2; Ath = 3; ppv = 3; = 0.1; Archive size = 100

and to maintain a uniform spread on those regions. The difficulty of ZDT4 is finding the global Pareto front. Finally, ZDT6 [115, 116] has a non convex Pareto front.

7.1 Performance Evaluation

Figure 8 shows the box plots of the hyper volume indicator, i.e., the IH values, found by all chosen MOPSOs for all test problems. The figure shows that DSMOPSO, DMOPSO, and MOPSO achieve high IH values for all test problems. Higher IH values indicate that the solutions found by an algorithm are able to dominate a larger region in the objective space. In Fig. 4, the IH values of MOPSOs are normalized for each test problem. As a result, the highest IH value achievable will equal one. Comparing the IH values of DSMOPSO with DMOPSO and MOPSO for ZDT1, ZDT3, and DTLZ2, DSMOPSO is slightly lower, which is also confirmed by the tested result computed using the Wilcoxon rank-sum test in Table 3. However, the results in Table 3 show that DSMOPSO is better than DMOPSO for ZDT2 and ZDT6 and performs better than MOPSO for ZDT2. Only for ZDT4, DSMOPSO shares the same victory with DMOPSO and MOPSO. For the case of DTLZ2, although the IH values of DSMOPSO are lower than the rest of MOPSOs, the difference in IH values is very small. In addition, the results in Table 2 indicate that there is no significant difference between the solutions found by DSMOPSO and the rest of the MOPSOs. Overall, both box plots and results in Table 3 clearly show that the performance of DSMOPSO is significantly better than cMOPSO.

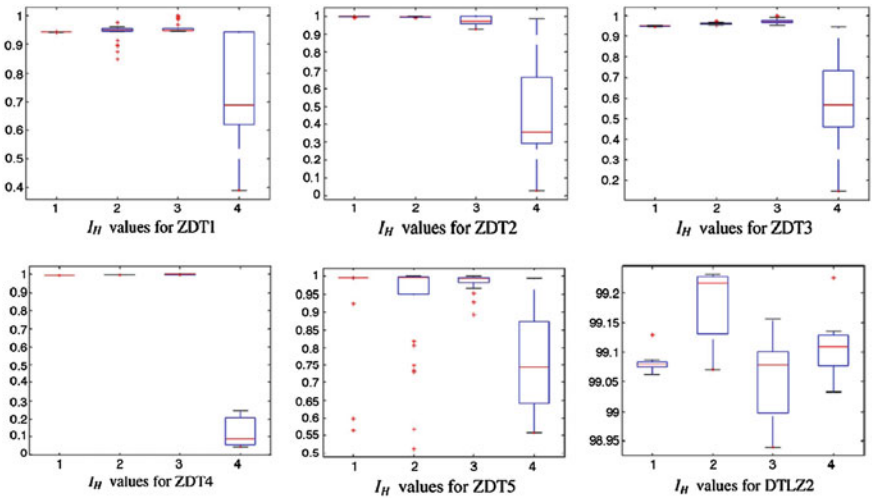


Fig. 8 Box plot of hypervolume indicator (IH values) for all test functions represented (in order): DSMOPSO, DMOPSO, MOPSO, and cMOPSO

Table 3 Distribution of IH values tested using wilcoxon rank-sum test

Test functions	I_H (DSMOPSO) AND		
	I_H (DMOPSO)	I_H (MOPSO)	I_H (cMOPSO)
ZDT1	(−3.6283, 2.9e-4)	(−6.0537, 1.9e-9)	(3.9776, 6.9e-5)
ZDT2	(2.8620, 0.04)	(2.8620, 0.042)	(4.6455, 3.4e-6)
ZDT3	(−6.5569, 5.5e-11)	(−6.6318, 3.3e-11)	(6.5421, 6.1e-11)
ZDT4	(3.8575, >0.05) no difference	(0.290, >0.05) no difference	(4.6455, 3.4e-6)
ZDT6	(−2.7837, 0.005)	(0.8087, >0.05) no difference	(5.5674, 2.6e-6)
DTLZ2	(0.5127, >0.05) no difference	(0.5681, >0.05) no difference	(0.5589, >0.05) no difference

In each cell, the results are presented as (Z-VALUE, P-VALUE) for each pair of DSMOPSO and a selected MOPSO

The comparative study shows that DSMOPSO [117–119] occasionally exhibits slower search progression, which may render larger computational cost than some selected MOPSOs. This may be due to two possible reasons. The growth rate for the number of swarms is not high enough and is lacking of a good strategy to enhance the communication among and within swarms. The characteristics observed from the dynamic variation of the number of swarms with respect to iterations show the need to effectively trigger the objective space compression and expansion routine.

7.2 Conclusion and Future Work

In this chapter, a systematic empirical study is conducted on various approaches in MOPSO, analysing their convergence and diversity. One variant of PSO i.e., PSO with dynamic multiple swarms is experimented and compared with other well known variants. Three main ingredients are developed in this algorithm. First, the swarm growing and declining strategies are designed to dynamically allocate new swarms in the needed area and to remove swarms with little or no contribution to the search process. These two strategies promote diversity by placing new swarms in the unexplored areas and by eliminating swarms that reside in crowded regions. Second, PSO updating rule is modified to incorporate the knowledge sharing among swarms and encourage convergence and diversity at the same time. Third, the objective space compression and expansion strategy is suggested to allow adjustment of the size of the objective space to ensure that the swarms orderly and progressively find the true Pareto front.

The main features of the study could be summarized as follows.

(a) The experimented approach has been effectively applied to solve the MOP, with no limitation in handing higher-dimensional problems.

(b) The algorithm was able to find a well distributed Pareto optimal curve in the objective space.
(c) The algorithm keeps track of all the feasible solutions found during the optimization and therefore does not have any restrictions on the number of the Pareto optimal solutions found.

One aspect that we would like to explore in the future is the use of a crowding operator to improve the distribution of non dominated solutions along the Pareto front. This would improve the capabilities of the algorithm to distribute uniformly the non dominated vectors found. We are also considering the possibility of extending this algorithm so that it can deal with dynamic functions. Finally, it is desirable to study in more detail the parameters fine tuning required by different algorithms, as to provide a more solid basis to define them.

References

1. Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C.M., da Fonseca, V.G.: Performance assessment of multiobjective optimizers: an analysis and review. IEEE Trans. Evol. Comput. **7**(2), 117–132 (2003)
2. Conover, W.J.: Practical Nonparametric Statistics, 3rd edn, pp. 272–286. Wiley, New York (1999)
3. Van Veldhuizen, D.A., Lamount, G.B.: Multiobjective evolutionary algorithm research: a history and analysis. Department of Electrical and Computer Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH, Technical Report TR-98-03 (1998)
4. Fonseca, C.M., Fleming, P.J.: Multiobjective optimization and multiple constraint handling with evolutionary algorithms—Part I: a unified formulation. IEEE Trans. Syst., Man, Cybern. A, Syst., Hum. **28**(1), 26–37 (1998)
5. Fonseca, C.M., Fleming, P.J.: Multiobjective optimization and multiple constraint handling with evolutionary algorithms—Part II: application example. IEEE Trans. Syst., Man, Cybern. A, Syst., Hum. **28**(1), 38–47 (1998)
6. Venkatraman, S., Yen, G.G.: A generic framework for constrained optimization using genetic algorithms. IEEE Trans. Evol. Comput. **9**(4), 424–435 (2005)
7. de Lima, P., Yen, G.G.: Multiple objective evolutionary algorithm for temporal linguistic rule extraction. ISA Trans. **44**(2), 315–327 (2005)
8. Zitzler, E.: Evolutionary algorithms for multiobjective optimization: methods and applications. Ph.D. Dissertation, Swiss Federal Institute Technology, Zurich (1999)
9. Goldstein, M.L., Yen, G.G.: Using evolutionary algorithms for defining the sampling policy of complex n-partite networks. IEEE Trans. Knowl. Data Eng. **17**(6), 762–773 (2005)
10. Buche, D., Muller, S., Koumoutsakos, P.: Self-adaptation for multi-objective evolutionary algorithms. In: Fonseca, M.C., Fleming, J.P., Zitzler, E., Deb, K., Thiele, L. (eds.) Second International Conference on Evolutionary Multi-Criterion Optimization, EMO 2003. Lecture Notes in Computer Science, vol. 2632, pp. 267–281. Springer, Faro April 2003
11. Chow, Chi-kin., Tsui, Hung-tat .: Autonomous agent response learning by a multi-species particle swarm optimization. In: Congress on Evolutionary Computation (CEC'2004), Portland, June 2004, vol. 1, pp. 778–785. IEEE Service Center
12. Srinivasan, D., Seow, H.T.: Particle swarm inspired evolutionary algorithm (PS-EA) for multiobjective optimization problem. In: Congress on Evolutionary Computation (CEC'2003), Canberra, December 2003, vol. 3, pp. 2292–2297. IEEE Press

13. Salazar-Lechuga, M., Rowe, J.: Particle swarm optimization and fitness sharing to solve multi-objective optimization problems. In: Congress on Evolutionary Computation (CEC'2005), Edinburgh, September 2005, pp. 1204–1211. IEEE Press
14. Schaffer, D.J.: Multiple objective optimization with vector evaluated genetic algorithms. Ph.D. Thesis, Vanderbilt University (1984)
15. Schaffer, D.J.: Multiple objective optimization with vector evaluated genetic algorithms. In: Proceedings of the First International Conference on Genetic Algorithms and their Applications, pp. 93–100. Lawrence Erlbaum (1985)
16. Shi, Y., Eberhart, R.: Empirical study of particle swarm optimization. In: Congress on Evolutionary Computation (CEC'1999), Piscataway, NJ, (1999), pp. 1945–1950. IEEE Press
17. Clerc, M., Kennedy, J.: The particle swarm-explosion, stability, and convergence in a multi-dimensional complex space. IEEE Trans. Evol. Comput. **6**(1), 58–73 (2002)
18. Goldberg, E.D., Richardson, J.: Genetic algorithms with sharing for multimodal function optimization. In: Proceedings of the Second International Conference on Genetic Algorithms, pp. 41–49. Lawrence Erlbaum Associates (1987)
19. Tan, K.C., Lee, T.H., Khor, E.F.: Evolutionary algorithms with dynamic population size and local exploration for multiobjective optimization. IEEE Trans. Evol. Comput. **5**(6), 565–588 (2001)
20. Mostaghim, S., Teich, J.: Covering paretooptimal fronts by subswarms in multi-objective particle swarm optimization. In: Congress on Evolutionary Computation (CEC'2004), Portland, June 2004, vol. 2, pp. 1404–1411. IEEE Service Center
21. Baumgartner, U., Magele, Ch., Renhart, W.: Pareto optimality and particle swarm optimization. IEEE Trans. Magn. **40**(2), 1172–1175 (2004)
22. Das, I., Dennis, J.: A closer look at drawbacks of minimizing weighted sums of objectives for pareto set generation in multicriteria optimization problems. Struct. Optim. **14**(1), 63–69 (1997)
23. Knowles, J.D., Corne, D.W.: Approximating the nondominated front using the Pareto archived evolution strategy. Evol. Comput. **8**(2), 149–172 (2000)
24. Alvarez-Benitez, J.E., Everson, R.M., Fieldsend, J.E.: A MOPSO algorithm based exclusively on Pareto dominance concepts. In: Third International Conference on Evolutionary MultiCriterion Optimization, Guanajuato, Mexico, 2005, pp. 459–473
25. Zitzler, E., Thiele, L.: Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. IEEE Trans. Evol. Comput. **3**(4), 257–271 (1999)
26. Parsopoulos, E.K., Tasoulis, K.D., Vrahatis, N.M.: Multiobjective optimization using parallel vector evaluated particle swarm optimization. In: Proceedings of the IASTED International Conference on Artificial Intelligence and Applications (AIA2004), Innsbruck, Austria, February 2004, vol. 2, pp. 823–828. ACTA Press
27. Parsopoulos, E.K., Vrahatis, N.M.: Particle swarm optimization method in multiobjective problems. In: Proceedings of the 2002 ACM Symposium on Applied Computing (SAC'2002), Madrid, Spain (2002), pp. 603–607. ACM Press
28. Raquel R.C., Prospero C. Naval, Jr.: An effective use of crowding distance in multiobjective particle swarm optimization. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2005), Washington, DC, USA, June 2005, pp. 257–264. ACM Press
29. Ray, T., Kang, T., Chye, K.S.: An evolutionary algorithm for constrained optimization. In: Whitley, D., Goldberg, D., Cantu-Paz, E., Spector, L., Parmee, I., Beyer, Hans-Georg.(eds.) Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2000), pp. 771–777. Morgan Kaufmann, San Francisco (2000)
30. Ray, T., Liew, K.M.: A swarm metaphor for multiobjective design optimization. Eng. Optim. **34**(2), 141–153 (2002)
31. Rudolph, G.: On a multi-objective evolutionary algorithm and its convergence to the pareto set. In: Proceedings of the 5th IEEE Conference on Evolutionary Computation, Piscataway, New Jersey (1998), pp. 511–516. IEEE Press
32. Ozcan E., Mohan, K.C.: Particle swarm optimization: Surfing the waves. In: Congress on Evolutionary Computation (CEC'1999), Washington D.C. (1999), pp. 1939–1944. IEEE Press

33. Coello Coello, C.A.: A comprehensive survey of evolutionary-based multiobjective optimization techniques. Knowl. Inf. Syst. Int. J. **1**(3), 269–308 (1999)
34. Ozcan E., Mohan, K.C.: Analysis of a simple particle swarm optimization system. In: Intelligent Engineering Systems Through Artificial Neural Networks, pp. 253–258 (1998)
35. Coello C.A.C., Lechuga, S.M.: MOPSO: a proposal for multiple objective particle swarm optimization. In: Congress on Evolutionary Computation (CEC'2002), Piscataway, NJ, May 2002, vol. 2, pp. 1051–1056. IEEE Service Center
36. Nunes de Castro, L., Timmis, J.: An Introduction to Artificial Immune Systems: A New Computational Intelligence Paradigm. Springer, New York (2002)
37. Kennedy, J., Eberhart, R.C.: Swarm Intelligence. Morgan kaufmann Publishers Inc., San Francisco (2001)
38. Laumanns, M., Thiele, L., Deb, K., Zitzler, E.: Combining convergence and diversity in evolutionary multi-objective optimization. Evol. Comput. **10**, 263–282 (2002)
39. Mostaghim, S., Teich, J.: Strategies for finding good local guides in multi-objective particle swarm optimization (MOPSO). In: Proceedings of the 2003 IEEE Swarm Intelligence Symposium, Indianapolis, IN, April 2003, pp. 26–33. IEEE Service Center
40. Mostaghim, S., Teich, J.: Strategies for finding good local guides in multi-objective particle swarm optimization (MOPSO). In: Proceedings of the 2003 IEEE Swarm Intelligence Symposium, SIS'03, April 2003
41. Mostaghim, S., Teich, J.: The role of ε-dominance in multi objective particle swarm optimization methods. In: Congress on Evolutionary Computation (CEC'2003), Canberra, Australia, December 2003, vol. 3, pp. 1764–1771. IEEE Press
42. Zitzler, E., Deb, K., Thiele, L.: Comparison of multiobjective evolutionary algorithms: empirical results. Evol. Comput. **8**(2), 173–195 (2000)
43. Pulido T.G., Coello, C.A.C.: Using clustering techniques to improve the performance of a particle swarm optimizer. In: Proceedings of Genetic and Evolutionary Computation Conference, Seattle, WA, pp. 225–237 (2004)
44. Leong, W.F., Yen, G.G.: PSO-based multiobjective optimization with dynamic population size and adaptive local archives. IEEE Trans. Syst., Man, Cybern. B, Cybern. **38**(5), 1270–1293 (2008)
45. Becerra, R.L., Coello Coello, C.A., Hernndez-Daz, A.G., Caballero, R., Molina, J.: Alternative technique to solve hard multi-objective optimization problems. In: Proceedings of Genetic and Evolutionary Computation Conference, London, U.K, pp. 757–764 (2007)
46. Arabas, J., Michalewicz, Z., Mulawka, J.: GAVaPS-A genetic algorithm with varying population size. In: Proceedings of Congress on Evolutionary Computation, Orlando, FL, pp. 73–78 (1994)
47. Blackwell, T., Branke, J.: Multiswarms, exclusion, and anticonvergence in dynamic environments. IEEE Trans. Evol. Comput. **10**(4), 459–472 (2006)
48. Zhuang, N., Benten, M., Cheung, P.: Improved variable ordering of BBDS with novel genetic algorithm. In: Proceedings of IEEE International Symposium on Circuits and Systems, Atlanta, GA, pp. 414–417 (1996)
49. Grefenstette, J.: Optimization of control parameters for genetic algorithms. IEEE Trans. Syst., Man, Cybern. **SMC–16**(1), 122–128 (1986)
50. Fieldsend E.J., Singh, S.: A multi-objective algorithm based upon particle swarm optimisation, an efficient data structure and turbulence. In: The 2002 U.K. Workshop on Computational Intelligence, pp. 34–44 (2002)
51. Hu, X., Eberhart, R.: Multiobjective optimization using dynamic neighborhood particle swarm optimization. In: Proceedings of the 2002 Congress on Evolutionary Computation, (2002), IEEE Press
52. Pulido, T.G., Coello, C.A.C.: Using clustering techniques to improve the performance of a multi-objective particle swarm optimizer. In: Genetic and EvolutionaryComputation—GECCO 2004. LNCS, vol. 3102, pp. 225–237 (2004)
53. Okabe, T., Jin, Y., Olhofer, M., Sendhoff, B.: On test functions for evolutionary multi-objective optimization. In: Parallel Problem Solving from Nature—PPSN VIII. vol. 3242, pp. 792–802 (2004)

54. Pulido, T.G., Coello, C.A.C.: Using Clustering Techniques to improve the performance of a particle swarm optimizer. In: Genetic and Evolutionary Computation-GECCO 2004, Proceedings of the Genetic and Evolutionary Computation Conference—Part I. Lecture Notes in Computer Science, vol. 3102, pp. 225–237. Springer, June 2004
55. Reyes-Sierra, M., Coello, C.: Multi-objective particle swarm optimizers: a survey of the State-of-the-Art. Int. J. Comput. Intell. Res. **2**(3), 287–308 (2006)
56. Padhye, N.: Topology optimization of compliant mechanism using multi-objective particle swarm optimization. In: Proceedings of the 2008 GECCO, ACM, pp. 1831–1834 (2008)
57. Parsopoulos, K., Vrahatis, M.: Particle swarm optimization method in multi-objective optimization problems. In: Proceedings of the 2002 ACM Symposium on Applied Computing (SAC 2002), pp. 603–607 (2002)
58. Raquel, R.C., Naval, C.P.: An effective use of crowding distance in multiobjective particle swarm optimization. In: GECOO, pp. 257–264 (2005)
59. Mostaghim, S., Teich, J.: Quad-trees: a data structure for storing pareto-sets in mulit-objective evolutionary algorithms with elitism. In: Evolutionary Computation Based Multi-Criteria Optimization: Theoretical Advances and Applications, pp. 81–104 (2005)
60. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: improving the strength pareto evolutionary algorithm. Swiss Federal Institute of Technology (ETH), Zurich, Technical report TIK-Report 103, May 2001
61. Mostaghim, S., Teich, J.: Strategies for finding good local guides in multi- objective particle swarm optimization (MOPSO). In: Proceedings of the IEEE Swarm Intelligence Symposium, 2003. SIS'03, pp. 26–33 (2003)
62. Padhye, N., Branke, J., Mostaghim, S.: A comprehensive comparison of mopso methods: study of convergence and diversity-survey of state of the art. Report under review, submitted to CEC (2009)
63. Wierzbicki, A.P.: The use of reference objectives in multiobjective optimization. Fandel, G., Gal, T. (eds.) Multiple Objective Decision Making, Theory and Application, vol. 6(2) pp. 468–486 (1980)
64. Zitzler, E.: Evolutionary algorithms for multiobjective optimization: methods and applications. Shaker (1999)
65. Papadimitriou, C.H., Yannakakis, M.: On the approximability of trade-offs and optimal access of web sources (extended abstract). In: IEEE Symposium on Foundations of Computer Science (2000)
66. Reyes-Sierra, M., Coello, C.A.C.: Multi-objective particle swarm optimizers: a survey of the state-of-the art. Int. J. Comput. Intell. Res. **2**(3), 287–308 (2006)
67. Schott, J.R.: Fault tolerant design using single and multicriteria genetic algorithm optimization. Master's Thesis, Department of Aeronautics and Astonautics, Massachusetts Institute of Technology, Cambridge, MA (1995)
68. Moore, J., Chapman, R.: Application of particle swarm to multiobjective optimization. Department of Computer Science and Software Engineering, Auburn University,Technical report (1999)
69. Li, X.: A non-dominated sorting particle Swarm optimizer for multiobjective optimization. In: Genetic and Evolutionary Computation—GECCO 2003. LNCS, vol. 2723,pp. 37–48 (2003)
70. Everson, R.M., Alvarez-Benitez1 J.E., Fieldsend, J.E.: A mopso algorithm based exclusively on pareto dominance concepts. In: EMO, vol. 3410, pp. 459–473 (2005)
71. Kennedy, James, Eberhart, Russell C.: Swarm Intelligence. Morgan Kaufmann Publishers, San Francisco (2001)
72. Knowles, J., Thiele, L., Zitzler, E.: A tutorial on the performance assessment of stochastic multiobjective optimizers. Technical Report 214, Computer Engineering and Networks Laboratory (TIK), ETH Zurich (2006)
73. Wolpert, D.H., Macready, W.G.: No free lunch theorems for optimization. IEEE Trans. Evol. Comput. **1**, 67 (1997)
74. Fieldsend, J., Everson, R., Singh, S.: Using unconstrained elite archives for Multi-objective optimisation. IEEE Trans. Evol. Comput. **7**, 305–323 (2003)

75. Veldhuizen, D.V., Lamont, G.: Multiobjective Evolutionary Algorithms Research: A History and Analysis. Technical Report TR-98-03, Department of Electrical and Computer Engineering, Graduate School of Enginering, Air Force Institute Technology, Wright-Patterson AFB, OH (1998)
76. Deb, K., Thiele, L., Laumanns, M., Zitzler, E.: Scalable multi-objective optimization test problems. In: Congress on Evolutionary Computation (CEC2002). vol. 1, pp. 825–830 (2002)
77. Fieldsend, J.: Multi-objective particle swarm optimization methods. Technical Report No. 419, Department of Computer Science, University of Exeter (2004)
78. Huband, S., Hingston, P., Barone, L., While, L.: A review of multiobjective test problems and a scalable test problem toolkit. IEEE Trans. Evol. Comput. **10**(5), 477–506 (2006)
79. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Fourth IEEEInternational Conference on Neural Networks, pp. 1942–1948 (1995)
80. Coello, C.A.C., Van Veldhuizen, A.D., ALamont, B.G.: Evolutionary Algorithms for Solving Multi-objective Problems. Kluwer Academic Publishers, New York, May 2002. ISBN 0-3064-6762-3
81. Deb, K.: Multi-Objective Optimization using Evolutionary Algorithms. Wiley, Chichester, U.K (2001). ISBN 0-471-87339-X
82. Deb, K., Thiele, L., Laumanns, M., Zitzler, E.: Scalable Multi-objective Optimization Test Problems. In: Congress on Evolutionary Computation(CEC'2002), Piscataway, NJ, May 2002, vol. 1, pp. 825–830. IEEE Service Center
83. Lu, H., Yen, G.G.: Rank-density-based multiobjective genetic algorithm and benchmark test function study. IEEE Trans. Evol. Comput. **7**(4), 325–343 (2003)
84. Pulido, T.G., Coello, C.A.C.: The micro genetic algorithm 2: towardsonline adaptation in evolutionary multiobjective optimization. In:Fonseca, M.C., Fleming, J.P., Zitzler, E., Deb, K., Thiele, L.(eds.) Evolutionary Multi-Criterion Optimization. SecondInternational Conference, EMO 2003. Lecture Notes in ComputerScience, vol. 2632 pp. 252–266. Springer, Faro, April 2003
85. Tripathi, K.P., Bandyopadhyay, S., Pal. K.S.: Multi-objective particle swarm optimization with time variant inertia and acceleration coefficients. Inf. Sci. **177**(22), 5033–5049 (2007)
86. Pulido, T.G., Coello, C.A.C.: Using clustering techniques to improve the performance of a particle swarm optimizer. In: Deb, K., et al. (ed.) Proceedings of the Genetic and Evolutionary Computation Conference (GECCO2004). Lecture Notes in Computer Science, vol. 3102, pp. 225–237. Springer, Seattle, June 2004
87. Tan, C.H., Goh, C.K., Tan, K.C., Tay, A.: A cooperative coevolutionary algorithm for multi-objective particle swarm optimization. In: 2007 IEEE Congress on Evolutionary Computation (CEC2007), Singapore, September 2007, pp 3180–3186. IEEE Press
88. Clerc, M.: The swarm and the queen: towards a deterministic and adaptative particle swarm optimization. In: Proceedings of the Congress on Evolutionary Computation, (1999), pp. 1951–1957. IEEE Press
89. Mostaghim, S., Teich, J., Tyagi, A.: Comparision of data structures for storing pareto-sets in moeas. In: IEEE Proceedings World Congress on Computational Intelligence, pp. 843–849 (2002)
90. Van Veldhuizen, A.D., Lamount, B.G.: Multiobjective evolutionary algorithm research: a history and analysis. Department of Electrical and Computer Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH, Technical Report TR-98- 03 (1998)
91. Fieldsend, E.J., Singh S.: A multiobjective algorithm based upon particle swarm optimisation, an efficient data structure and turbulence. In: Proceedings of the 2002 U.K. Workshop on Computational Intelligence, pp. 37–44. Birmingham, U.K, September 2002
92. Engelbrecht, P.A.: Fundamentals of Computational Swarm Intelligence.Wiley (2005)
93. Coello, C., Pulido, G., Lechunga, M.: Handling multiple objectives with particle swarm optimization. IEEE Trans. Evol. Comput. **8**(3), 256–279 (2004)
94. Hanne, T.: On the convergence of multiobjective evolutionary algorithms. Eur. J. Oper. Res. **117**, 553–564 (1999)

95. Coello, C.A.C., Pulido, T.G., Lechuga, S.M.: Handling multiple objectives with particle swarm optimization. IEEE Trans. Evol. Comput **8**(3), 256–279 (2004)
96. Zitzler, E., Deb, K., Thiele, L.: Comparison of multiobjective evolutionary algorithms: empirical results. Evol. Comput. **8**(2), 173–195 (2000)
97. Eberhart, C.R., Shi, Y.: Comparison between genetic algorithms and particle swarm optimization. In: Porto, W.V., Saravanan, N., Waagen, D., Eibe, E.A. (eds.) Proceedings of the Seventh Annual Conference on Evolutionary Programming, pp. 611–619. Springer, March 1998
98. Engelbrecht, Andries P. (ed.): Computational Intelligence: An Introduction. Wiley, England (2002)
99. Sierra, R.M., Coello, C.A.C.: Improving PSO-based multiobjective optimization using crowding, mutation and ε-dominance. In: Evolutionary Multi-Criterion Optimization Conference, Guanajuato, Mexico, pp. 505–519 (2005)
100. Mostaghim, S., Teich, J.: Strategies for finding good local guides in multi-objective particle swarm optimization. In: Proceedings of IEEE Swarm Intelligence Symposium, Indianapolis, IN, pp. 26–33 (2003)
101. van den Bergh, F.: An Analysis of Particle Swarm Optimizers. Ph.D. Thesis, Faculty of natural and agricultural science, University of Pretoria (2001)
102. Villalobos-Arias, A.M., Pulido, TG., Coello, C.A.C.: A proposal to use stripes to maintain diversity in a multi-objective particle swarm optimizer. In: Proceedings of Swarm Intelligence Symposium, Pasadena, CA, pp. 22–29 (2005)
103. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. **6**(2), 182–197 (2002)
104. Fieldsand, J.E., Singh, S.: A multi-objective algorithm based upon particle swarm optimization, an efficient data structure and turbulence. In: Proceedings of U.K. Workshop on Computational Intelligence, Birmingham, U.K.,pp. 37–44 (2002)
105. Zhang, L.B., Zhou, C.G., Liu, X.H., Ma, Z.Q., Ma, M., Liang, Y.C.: Solving multi objective problems using particle swarm optimization. In: Proceedings of Congress on Evolutionary Computation, Canberra, Australia, pp. 2400–2405 (2003)
106. Eberhart, C.R., Dobbins, R., Simpson, K.P.: Computational Intelligence PC Tools. Morgan Kaufmann Publishers (1996)
107. Yen, G.G., Lu, H.: Dynamic multiobjective evolutionary algorithm: adaptive cell-based rank and density estimation. IEEE Trans. Evol. Comput. **7**(3), 253–274 (2003)
108. Kennedy, J., Eberhart, R.C.: Particle swarm optimization. In: Proceedings of IEEE International Conference on Neural Networks, Perth, pp. 1942–1948 (1995)
109. Deb, K., Goldberg, E.D.: An investigation of niche and species formation in genetic function optimization. In: Schaffer, D.J. (ed.) Proceedings of the Third International Conference on Genetic Algorithms, pp. 42–50. George Mason University, Morgan Kaufmann Publishers, San Mateo, June 1989
110. Okabe, T., Jin, Y., Sendhoff, B., Olhofer, M.: Voronoi-based estimation of distribution algorithm for multi-objective optimization. In: Proceedings of Congress on Evolutionary Computation, Portland, OR, pp. 1594–1601 (2004)
111. Sierra, R.M., Coello, C.A.C.: Multi-objective particle swarm optimizers: a survey of the state-of-the-art. Int. J. Comput. Intell. Res. **2**(3), 287–308 (2006)
112. Sierra RM., Coello, C.A.C.: Improving PSO-based multi-objective optimization using crowding, mutation and ε-dominance. In: Evolutionary Multi-Criterion Optimization (EMO 2005). LNCS, vol. 3410, pp. 505–519 (2005)
113. Sierra R.M., Coello, C.A.C.: Improving PSO-based multi-objective optimization using crowding, mutation and ε-dominance. In: Third International Conference on Evolutionary Multi-Criterion Optimization, EMO 2005. LNCS vol. 3410, pp. 505–519. Springer (2005)
114. Shi, Y., Eberhart, R.: Parameter selection in particle swarm optimization. In: Evolutionary Programming VII: Proceedings of the Seventh Annual Conference on Evolutionary Programming, pp. 591–600. Springer, New York (1998)
115. Carlos, C.A.C., Pulido, P.T., Lechuga, S.M.: Handling multiple objectives with particle swarm optimization. IEEE Trans. Evol. Comput. **8**(3), 256–279 (2004)

116. Coello, C.A.C., Van Veldhuizen, A.D., Lamont, B.G.: Evolutionary Algorithms for Solving Multi-Objective Problems. Kluwer Academic Publishers, New York (2002)
117. Li, X.: Better spread and convergence: particle swarm multiobjective optimization using the maximin fitness. In: GECCO, pp. 117–128 (2004)
118. Moore, J., Chapman, R.: Application of particle swarm to multiobjective optimization. Department of Computer Science and Software Engineering, Auburn University (1999)
119. Mostaghim, S., Teich. J.: The role of ε-dominance in multi objective particle swarm optimization methods. In: The Proceedings of the 2003 Congress on Evolutionary Computation, pp. 1764–1771 (2003)

Binary Ant Colony Optimization for Subset Problems

Nadia Abd-Alsabour

Abstract Many optimization problems involve selecting the best subset of solution components. Besides, many other optimization problems can be modelled as a subset problem. This chapter focuses on developing a new framework in ant colony optimization (ACO) for optimization problems that require selection rather than ordering with an application to feature selection for regression problems as a representative for subset problems. This is addressed through three steps that are: explaining the main guidelines of developing an ant algorithm, demonstrating different solution representations for subset problems using ACO algorithms, and proposing a binary ant algorithm for feature selection for regression problems.

Keywords Ant colony optimization · Binary ant colony optimization (BACO) · Subset problems · Feature selection

1 Introduction

ACO is based on the foraging behavior of ants i.e., how ants can find the shortest path between food sources and their nest (clearly a shortest path problem). Thus, representing the optimization problem by a path in the construction graph is considered of crucial importance in much of the ACO literature [1].

ACO algorithms have been applied to many discrete optimization problems. Many of them are ordering or sequencing problems where the problem to be solved is modeled as a search for a minimum cost path in a graph. The artificial ants walk on the graph representing the given problem and each path corresponds to a potential

N. Abd-Alsabour (✉)
Cairo University Cairo, Cairo, Egypt
e-mail: nadia.abdalsabour@cu.edu.eg

S. Dehuri et al. (eds.), *Multi-objective Swarm Intelligence*,
Studies in Computational Intelligence 592, DOI 10.1007/978-3-662-46309-3_4

solution to the given problem. The artificial ants deposit pheromone on the path in a quantity proportional to the quality of the solution represented by that path. However, there are many problems that show a marked divergence from classic shortest path problems. Subset problems require selecting rather than ordering or sequencing [2]. Consequently, construction graph for these problems can not be helpful and may offer little assistance in determining the appropriate pheromone representation [1]. This raises the need for developing suitable ACO algorithms for tackling these optimization problems that use suitable problem representations while in the same time follow the main concepts of ACO metaheuristic. This chapter focus on developing a BACO algorithm that best suits optimization problems that require selecting solution components while in the same time integrates that into the ant colony system (ACS) framework. Moreover, the chapter explains the main guidelines for developing a new ant algorithm. Besides, it explains the different solution representations for subset problems introducing the need for developing BACO for this kind of optimization problems.

The rest of this chapter is organized as follows. Section 2 introduces the basics of ACO and the main guidelines for developing an ant algorithm. Section 3 addresses the fundamentals of feature selection. The sect. 4 briefly describes different solution approaches for subset problems using ACO algorithms. The sect. 5 explains the proposed algorithm. Section 6 details the experiments carried out and presents the obtained results. The discussion of the obtained results is given in Sect. 7. Section 8 concludes this paper and finally Sect. 9 highlights future work in this area.

2 Ant Colony Optimization

The behavior of ants has inspired the development of artificial distributed problem solving systems. In terms of biological systems, each ant has its own agenda and follows very simple rules; more complex global-level patterns emerge solely through the ants' interactions with each other and their environment (without supervision or central control). There are two types of interactions between ants in a colony: *direct interactions* (involve tactile, visual, or chemical contact) and *indirect interactions* (initiated by individuals that exhibit some behavior that modifies the environment which in turn stimulates a change in the behavior of other individuals). Indirect interactions can solve difficult problems although they may be simple [3]. The ant colony (rather than individual ants) can be seen as an intelligent entity for its great level of self-organization and the complexity of the tasks it performs. Natural ant colony systems inspired many researchers in computer science to develop new solutions for optimization problems [4].

Some guidelines of developing an ant algorithm for an optimization problem can be summarized by the following:

- An appropriate problem representation which allows the artificial ants to build solutions. The most important point when developing an ant algorithm is to map

the considered optimization problem to a representation that can be used by the artificial ants to build solutions [5, 6]. The traditional approach when developing most of ACO algorithms was to represent the given optimization problem as a path finding problem using the construction graph where the nodes represent solution components, edges represent transitions or connections between the nodes, and (the most important point here) the solution for the given optimization problem is represented by one path (this was not accurate in the previous work that deal with subset problems as a path finding problem as explained in Sect. 4). That is why the optimization problems that have been solved using ACO algorithms are called graph based shortest path problems.

- Two factors are used in guiding the search process, these are:

  1. Pheromone values that are numerical values that the artificial ants put on the edges, the nodes, or both of them. The most important point here is the modification (update) of the pheromone values that should be proportional to the solution quality represented by the path representing the candidate solution for the given optimization problem, and
  2. The heuristic information (that should be problem specific) associated to each component, transition, or both of them (depending on the used problem representation). The main role of the heuristic information is to avoid constructing tours of too bad quality by biasing ants so that they build reasonably good tours from the very beginning. If no obvious heuristic information exists, using an ACO algorithm incorporating local search may be enough to achieve good results [6]. The heuristic information must be used in guiding the search process rather building solutions as in some of the previous ACO algorithms for feature selection. Using heuristic information becomes crucial for good performance when local search is not applicable as explained below.

- If possible, use an efficient local search algorithm (optional): it is used to implement centralized actions which can not be performed by single ants. It can observe the path found by each ant in the colony and select one or a few ants (those that built the best solutions). Once ants have completed their solutions construction, a local search routine can be performed. Such a coupling of solution construction with local search is a promising approach. Because ACO's solution construction uses a different neighborhood than local search, the probability that local search improves a solution constructed by an ant is quite high. On the other hand, local search alone suffers from the problem of finding good starting solutions (provided by the artificial ants) [6]. Although using local search enhances the performance of ACO and have been used with most of ACO for different optimization problems, there exist problems for which local search is of limited effectiveness such as very strongly constraint problems. These are problems for which local search efficient polynomial neighborhoods contain few solutions or none at all and local search is of very limited use [7].

- An appropriate definition of the pheromone update rule that specifies how to modify the pheromone trails [8–10].

3 Feature Selection

For a given problem, the number of potential features can be large. This is because:

- The collected data are not solely for one particular task as data mining. For example, it can be collected for a general task as book keeping or required by a law.
- Even if we know that features are designed for a particular task, relevant features are often unknown as we can know that only after studying the collected data.
- A task may require data from different sources. If data from each source is large, the join of them could be large [11].

Not all of these features are likely to be necessary for accurate prediction and including them may lead to a worse predictor than if they were removed. Such features may slow down and mislead the learning step and do not contribute to the prediction process. Hence, feature selection from the original set of features is highly desirable in many situations in order to reduce the number of features, improve the predictions accuracy, and to simplify the learned representation. The main goal of feature selection is to find a subset of features with predictive performance comparable to the full set of features [12–14] i.e., the results of feature selection should be data having potential for good solutions.

Feature selection can be used in many applications from choosing the most important social-economic parameters for determining whoever a person that can return a bank loan to dealing with a chemical process and selecting the best set of ingredients [15]. It is used in many applications in order to simplify their datasets by eliminating the redundant and irrelevant features without affecting the prediction accuracy. Some of these applications are: face recognition [16–21], face detection [20, 22, 23], bioinformatics [24–26], web page classification [27, 28], text categorization [29, 30], speaker recognition [31], and customer relationship management [32].

Feature selection problems have three main characteristics that need to be addressed in order for them to be solved using an ACO algorithm. These are:

1. As a subset problem, the order between the solution components (features) in a partial solution is not important. Therefore, the next feature to be selected (to the solution under construction) does not depend on the last feature added to the solution under construction [6].
2. There is no a priori heuristic information that can be used in guiding the search process besides the pheromone. The use of heuristic information can guide the artificial ants towards the most promising solutions.
3. Solutions for feature selection problems do not have fixed length i.e., different ants may have solutions of different lengths [33].

The proposed algorithm addressed these characteristics as follows:

1. The first characteristic is addressed by not representing feature selection by a path in a graph. Yet, a set of binary bits is associated with each ant to represent its selection. For example: if we have a dataset consists of four features and the binary set is 0110, this means that the first and the fourth features in the given dataset will not be selected and the second and the third features will be selected.
2. The second characteristic is addressed by artificially defining a possible one that is the proportion of the ants that chose a particular feature as heuristic information that represents the desirability of that feature. So, the proposed algorithm does not need prior knowledge of the features. This heuristic information can be used with any used dataset regardless of its domain.
3. The third characteristic is addressed by not fixing the length of the solution of each ant to a specific length out of the original set of features. This allows each ant to construct its solution without any prior restriction. The only criterion used in evaluating the constructed solution by each ant is the prediction accuracy. Abd-Alsabour et al. [33] proved that not fixing the length of the selected feature subsets (to a specific length out of the original set of features) in the beginning of the algorithm gives better results than fixing.

While most of literature deals with feature selection for classification, this paper focus on feature selection for regression problems. Both classification and regression problems are the two major types of supervised learning (prediction) [34] where the goal is to predict (for a given record) the value of the output (target) feature based on the values of the input features [1, 35]. The learning process is supervised i.e., the presence of the output feature guides the learning process (the data are labeled).

The main differences between classification and regression are:

1. The types of the target feature: if it is categorical i.e., is chosen from a fixed set of possibilities such as predicting which one of three specific treatments a breast cancer patient should receive or predicting whether a credit application is low, medium, or high risky, then the prediction is called classification. If it is real-valued such as predicting the profit a retail store might earn from a given customer over a ten-year period or predicting the number of people who might access a particular website in a given month, then the prediction is called regression [32, 35, 36],
2. In regression, the records are classified according to some predicted future behavior or estimated future value (the result of regression is a prediction of a future behavior) while in classification there is no explicit order in the classes (often descriptive labels rather than numbers are used to denote the classes), and
3. Evaluating the performance of the system is different in both of them (different fitness function). These are the primary reasons for treating regression problems as a separate task from classification problems [37].

4 Solution Representations for Subset Problems Using Ant Colony Optimization

Much of the ACO literature considers the development of a construction graph for the problem to be solved as essential to the application of ACO as it is a graph based shortest path problem that real ants solve when traveling from the nest to a food source [1].

According to Leguizam'on and Michalewicz [38], there is no real concept of path in subset problems. According to Dorigo et al. [39], a construction graph could be obtained by representing solution components as vertices on which pheromone is deposited although it seems less natural for subset problems. Hence, even if the pheromone is associated with the items (an intuitive choice used in several ACO algorithms for these problems), considering these problems in terms of a graph is quite artificial [1]. According to Solnon and Bridge [2], the order in which objects are selected is not significant. Therefore, it is meaningless to model them as path finding problems [2].

It is evident that ACO has been applied to numerous problems that are not naturally described in terms of graphs. One example is in the application of ACO to the maximum clique problem where Solnon and Bridge [2] associate pheromone with edges between all pairs of nodes in the clique under construction. Thus, the pheromone values associated with a solution do not correspond to a single path [1]. Another example is in Lee et al. [40] where they adopted the graph based ant system to solve feature selection where candidate solutions can be represented in directed graphs. It is particularly successful in solving combinatorial optimization problems such as constructing paths based on direct graphs with specific starting points. In their algorithm, ACODISTAI, each traversed path by an ant in a cycle represents a candidate solution to the feature selection. The selected features are represented as a combination of arcs where ants have traversed through the graph. Every ant must visit every node in the graph at most once. Every ant starts at a specific node (first feature) and ends at a specific node (last feature) checking every node in between with a given sequence. Every node has two arcs connected to its next visiting node, each representing either selection or exclusion of the feature it is assigned to. Therefore, combining traversed arcs together gives a full representation of a candidate solution of the feature selection to classify the given dataset. This is illustrated in Fig. 1 reproduced from [40, p. 468]. Another example is in Bello et al. [41] where a feature subset problem is viewed as a network in which nodes represent features and all nodes are connected by bidirectional links. Pheromones are associated with nodes. Ants perform a forward selection in which each ant expands its subset step-by-step by adding new features. Finally, Jensen and Shen [42] represented feature selection as a graph where the nodes represent features and the edges between the nodes denote the choice of the next feature. The search for the optimal feature subset is then an ant traversal through the graph where a minimum number of nodes are visited that satisfies the traversal stopping criterion. Figure 2, reproduced from [42, p. 19], illustrates this process. In this figure, the ant is currently at node

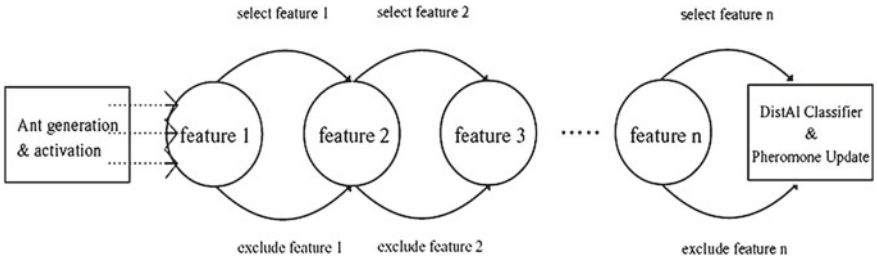


Fig. 1 Feature subset selection using an ACO algorithm with DistAl [40, p. 468]

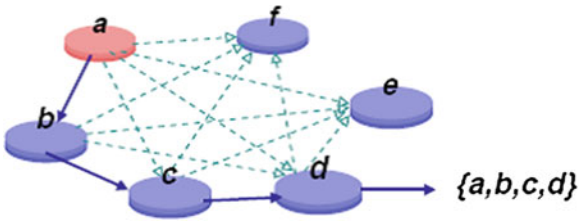


Fig. 2 ACO problem representation for FS [42, p. 19]

a and has a choice of which feature to add next to its path (dotted line). It chooses feature b next based on the transition rule, then c and then d. Upon arrival at d, the current subset a, b, c, d is determined to satisfy the traversal stopping criterion. Similarly, Aghdam et al. [43] follow similar strategy to Jensen and Shen [42] but the pheromone is associated with the nodes of the graph instead of with the edges of the graph. The nodes represent features and the edges between them denote the choice of the next feature. Each ant starts with a random feature. i.e., each ant starts path construction at a different feature. From these initial positions, they traverse edges probabilistically until a traversal stopping criterion is satisfied.

It should be noted that many optimization problems can be modeled in multiple ways. A problem might be modeled as a subset selection problem or as a kind of ordering problems. An example is the TSP that can be modeled as a constraint satisfaction problem (as a subset selection problem) i.e., variables represent cites and values are integers that represent the order in which the cities are visited.

The last point in this issue is about where the pheromone should be related with (suitable pheromone representation). According to Montgomery [1], two main pheromones strategies may be considered for solving subset selection problems using ACO algorithms: (1) Associate a pheromone trail τ_i with each object i so that τ_i represents the desirability of selecting object i, and (2) Associate pheromone trail τ_{ij} with each pair of different objects so that τ_{ij} representing the desirability that objects i and j belong to the same subset. According to Leguizam'on and Michalewicz [38], in ordering problems, the pheromone is laid on paths while for subset problems no paths exist connecting the items. Therefore, the idea of "the more pheromone trail on a particular path, the more profitable is that path" is adapted to "the more pheromone trail on a particular item, the more profitable that item is" i.e., the pheromone is

put on items not paths. A value should be assigned to each element without any considerations about possible connections between them (ordering is not important any longer) i.e., items only should be considered instead of connections between them [38].

The previous discussion shows the need to an alternative problem representation (rather than using the construction graph) that more suits there characteristics and in the same time follows the general characteristics of ACO metaheuristic.

5 Binary ACO

The proposed algorithm here is a binary ant colony optimization that uses binary representation rather than representing the given optimization problem as a path finding problem where a set of binary bits (of a length equivalent to the number of the features in the given dataset) is associated with each ant to represent its feature selection. If the nth bit is a 1 then that feature in the given dataset is selected, otherwise the feature is not selected. At the start of the algorithm, the bits are randomly initialized to zeros and ones. This means that this algorithm is not a constructive algorithm as ACO algorithms.

The main steps of this algorithm are as follows:

- *Initialization*: in this phase, the parameters of this proposed algorithm are set.
- *Construct a solution by each ant*: at each construction step, each ant selects the next feature using the move probability that is calculated based on the pheromone values and the heuristic information. After each construction step, the local pheromone update is performed by all ants to the traversed feature. Each feature can be selected at most once. An ant's solution represents its own feature subset. The length of each ant's feature subset does not need to be equal to other ants' solutions.
- *Build the predictor*: each ant passes its feature subset to the predictor and receives its prediction accuracy.
- *Update the best solution and the best ant*: the update is based on the RMSE (the fitness value).
- *Update the pheromone values* (global pheromone update): this is performed by the best ant only to its subset at the end of each iteration.
- *Termination*: if a predefined maximum number of iterations is reached, the algorithm halts and outputs the best solution encountered.

This proposed algorithm works as follows:

At each construction step, each ant selects a feature with the probability computed according to Eq. 1 i.e., each ant checks all the possible features one by one in a specific order checking if they are chosen or not.

$$P_i = \tau_i \cdot \mu_i \tag{1}$$

where, τ_i is the pheromone value associated with feature i, and μ_i = the proportion of the ants that selected this feature and acts as heuristic information that represents the desirability of feature i.

After each construction step, the local pheromone update is performed by all ants according to Eq. 2:

$$\tau_i = (1 - \varphi) \cdot \tau_i + \varphi \cdot \tau_0 \tag{2}$$

where, φ is a parameter and τ_0 is the initial pheromone. This equation is applied by each ant to the last traversed feature i.e., while building a solution, ants choose features and change their pheromone level by applying the local pheromone update making them less attractive and therefore favoring the exploration of features not yet chosen.

At the end of each iteration, the global pheromone update is performed according to Eq. 3:

$$\tau_i = (1 - \rho) \cdot \tau_i + (\rho \cdot 1/L_{best})^{\beta} \tag{3}$$

where,

L_{best} is the number of features selected by the best ant,
ρ is the evaporation rate, and
β is a parameter.

This equation is used by the best ant only at the end of each iteration and is applied to all features that it has chosen i.e., features that belong to the best solution are features that receive the greater amount of pheromone.

6 Computational Experiments

In order to test this algorithm, a support vector machine regression (SVR) algorithm is used. Although many prediction techniques have been developed in literature, Support vector machines (SVM) are considered one of the most effective ones not only in performing classification but also in performing regression [35]. The support vector machines for regression are a robust technique [44]. SVMs provide excellent generalization capabilities, fast, and robust to high input space dimension. It not only has a solid theoretical foundation, but also performs classification more accurately than most other algorithms in many applications. For example, it has been shown by several researchers that SVM is the most accurate algorithm for text classification and it is widely used in web page classification and bioinformatics applications. Although not all machine learning algorithms require the phase of feature selection, feature selection is important in building SVM (there are two cases for building predictors: by all the features of a given data set or by only a subset of the features of the given dataset) [45, 46].

In general, the radial basis function (RBF) kernel is a reasonable first choice. This kernel nonlinearly maps samples into a higher dimensional space so it can handle

the case when the relation between class labels and attributes is nonlinear. It is a general purpose kernel that is used when there is no prior knowledge about the data is available [47]. In the proposed algorithm, the radial basis function (RBF) kernel is used with the used SVR.

There are often more than two classes in a data set. Therefore, binary SVMs are usually not enough to solve the whole problem. In order to solve a multi-class classification problem, the whole problem is divided into a number of binary classification problem i.e., for the problem of N-class classification, train an SVM that learns to classify class one from all other classes, then another SVM that classifies class two from all other classes (so for N classes, we have N SVMs) and the final chosen SVM is the one that makes the strongest prediction. Usually, there are two approaches: "one against all" and "one against one" [48, 49]. In the proposed algorithm, "one against all" approach is used.

6.1 Fitness Function

The accuracy of a predictor refers to how well a given predictor can guess the value of the predicted feature for new or previously unseen data. Accuracy should be estimated using one or more test sets that are independent of the training set [34]. To asses performance in predicting a continuous response feature, we examine differences between observed score (y) and predicted score (y'). The difference between these values ($y - y'$) is called a *residual* which is an error in prediction. When the observed and the predicted responses are close, we have good prediction. In calculating goodness/badness of fit in predicting a continuous response, the squared residual is used. The Mean Squared Error (MSE) is the average of squared residuals for a given dataset [50]. The RMSE is just the square root of the MSE. It is probably the most easily interpreted static and it is usually best to report the RMSE rather than MSE since RMSE is measured in the same units as the data rather than the squared units. When comparing regression algorithms, the RMSE goes down. In this chapter, RMSE is used as a fitness function where each ant evaluates its solution based on its RMSE.

6.2 Cross Validation (CV)

In order to assess the adequacy of predictors, we can use part of the available data to develop predictors and another part to evaluate them. This is because they provide an honest appraisal of predictors' performance. It is not sufficient to show that a predictor works well with the data on which it was developed (the training set). We should show that it works well with other data (testing set) [50]. Moreover, the outcome (the feature subset) of many feature selection algorithms (for instance almost most of the algorithms that are based upon the wrapper methodology) is strongly dependent on

the training set size i.e., if the training set is small, the size of the reduced subset will be small as well due to the elimination of irrelevant and redundant features [51]. The question here is what the best size of each part (training and testing) should be. Besides, in some situations, data is scarce and we do not want to *waste* data on testing. The k-fold cross validation technique is designed to give an accurate estimation of the generalization error without wasting too much data. In k-fold cross validation the original set is partitioned into k subsets (folds). For each fold, a predictor is tested on this fold and is trained on the other folds. In the proposed algorithm, 10-fold cross validation is used.

6.3 Datasets

This proposed algorithm is tested using drug datasets [52] that are usually known as hard-modeled. This is because they have large number of features and a small number of samples. Getting good prediction results with these datasets can provide large financial and time savings in pharmaceutical research and development [53]. The details of the used datasets are shown in Table 1.

6.4 Method

In these experiments, the following two systems were developed:

- SVR: that uses the entire set of features (without the phase of feature selection), and
- SVR-FS: that uses a subset of features selected by this proposed algorithm.

In these two systems, 10-fold cross validation was used. The number of ants was set to the number of the features in the given dataset. The initial pheromone was set to 1. The number of iterations is 20 iterations. ρ was set to 0.2. β was set to 0.2. φ was set to 0.2.

Table 1 The details of the used drug datasets

Dataset name	No. of instances	No. of features
Benzo	195	32
Phen	22	110
Selwood	31	53
Pah	80	112
Qsbr_rw1	41	50

Table 2 The RMSE of SVR with and without the use of feature selection

Dataset name	No. of original features	Avg. no. of selected features	SVR (without FS)	SVR-FS (with FS)
Benzo	32	16	0.2074	0.1923
Phen	110	48.2	0.1806	0.139
Selwood	53	17.7	0.2057	0.163
Pah	112	29.9	0.1249	0.1083
Qsbr_rw1	50	13.3	0.2133	0.1492

6.5 Datasets

Table 2 shows the results of these two systems using the above mentioned datasets. The results of this algorithm represent the average of ten independent runs. These systems are implemented using the R language [54] and the WEKA machine learning tool [55]. All the experiments were run on a PC with a 2 GHz CPU and 2 GB RAM.

The previous results show that SVR-FS with this algorithm for performing feature selection outperforms SVR that uses all the features in all of the used datasets i.e., the RMSE of SVR (the last column) is smaller than that of SVR with all features (the fourth column). The number of features selected by this algorithm (the third column) is significantly smaller than the total number of the features in the original datasets (the second column) in all of the used datasets.

6.6 Comparisons with Other Algorithms

Amasyali and Ersoy [53] conducted a recent comparative study between the best performance regression algorithms both single regression algorithms and ensemble (more than one predictor are built and then are combined to produce a better prediction accuracy) regression algorithms with drug datasets. Table 3 compares some of their results with the results of this proposed algorithm (the last column).

The previous feature selection regression algorithms used in this comparison are as follows:

- RF-M5P: Ensemble based (Rotation Forest, RF) M5 Model Trees (M5P),
- PLS: Partial Least Squares,
- RF-NN: Ensemble based (Rotation Forest, RF) Neural Network, and
- RF-PLS: Ensemble based (Rotation Forest, RF) Partial Least Squares.

This comparison shows that this proposed algorithm is comparative with other regression algorithms although all of them (except the second one that is PLS) are based on ensemble approaches that are generally known that they perform better than single regression algorithms [53]. This shows that this proposed algorithm is a

Table 3 Comparisons with other regression algorithms

Dataset name	The best performing algorithm		SVR-FS (with this proposed algorithm)
	The algorithm name	RMSE	RMSE
Benzo	RF-M5P	0.21	0.1923
Phen	PLS	0.14	0.139
Selwood	RF-NN	0.21	0.163
Pah	RF-PLS	0.1083	0.10
Qsbr_rw1	–	–	0.1492

simple regression algorithm that does not require extra computation since it does not use ensample approaches for performing the regression.

7 Discussion

The proposed algorithm uses binary representation since considering representing subset problems in terms of a graph is quite artificial [1] (as explained in Sect. 4) and the most important point when developing an ant algorithm is to map the considered optimization problem to a representation that can be used by the artificial ants to build solutions [6]. Other previous ACO algorithms for feature selection mean by dealing with feature selection as a binary problem that the move probability is zero or one. In this case, these algorithms can't be called ant algorithms since ant algorithms are guided search techniques i.e., two factors (the pheromone values and the heuristic information) are used in guiding the search process and the move probability depends on the output (value) of the equation used to compute it. The heuristic information was used in the proposed algorithm in order to guide the search process besides the pheromone values. It should be noted that the heuristic information should be used in *guiding* the search rather in *building* the solutions. The used heuristic information is related to a given feature as it is used in computing the move probability to choose that feature. The obtained results are promising in terms of the solution quality and the number of selected features in all of the used the datasets.

8 Conclusion

In this paper, a new ACO metaheuristic for subset problems with an application to feature selection for regression problems was proposed. It uses binary representation and integrates this into an ACS framework introducing a new research direction in ant algorithms.

The main characteristics/advantages of this algorithm are as follows:

- The move probability is not zero or one as in Shen et al. [56] or uses only the weight (pheromone) associated with each feature as in Izrailev and Agrafiotis [57]. Yet, it uses two factors to compute it. These two factors are the old pheromone value associated with a feature and the heuristic information (since ant algorithms are guided search techniques). This heuristic information indicates how often a particular solution component has been chosen by different ants. Therefore the value of the move probability depends on the result of the equation used to compute the move probability.
- It uses the local pheromone update. Using the local pheromone update leads to decreasing the pheromone values on the selected features that encourages subsequent ants to choose other solution components and subsequently produce diverse solutions.
- The global pheromone update is performed by the best ant only. This is intended to make the search more directed i.e., those solution components belong to the best solution will receive reinforcement.
- The length of the solution of each ant is not fixed to a given length out of the length of the original set of features as in Gunturi et al. [58] and in Shamsipur et al. [59] allowing each ant to construct its solution without any prior restriction. The only criterion used in evaluating the constructed solution by each ant is its RMSE (the fitness value).
- It is a simple algorithm. This is because:
  1. it uses a single SVR only rather than ensemble of predictors as in Palanisamy and Kanmani [60],
  2. it uses a single ant colony only rather than more than one as in Shamsipur et al. [61],
  3. it uses a single ACO algorithm only rather than more than one as in Vieira et al. [62], and
  4. it does not use local search algorithm as in most of ACO algorithms.

9 Future Work

As a direction for future work in this area, there are a number of ways in which the proposed algorithm can be improved. In the proposed algorithm, the best ant only performs the global pheromone update. There may be other possibilities such as the use of the best two or more ants (*Elite strategy*).

Also, an interesting subject of ongoing research in this area is to establish the class of optimization problems that require binary representation and can be solved by the proposed algorithm.

References

1. Montgomery, E.J.: Solution biases and pheromone representation selection in ant colony optimization. Ph.D Thesis, Bond University, Australia (2005)
2. Solnon, C., Bridge, D.: An ant colony optimization meta-heuristic for subset selection problems. In: Nedjah, N., Mourelle, L.M. (eds.) Systems Engineering Using Particle Swarm Optimization, pp. 3–25. Nova Science Publishers, New York (2006)
3. Mirzayans, T., Parimi, N., Pilarski, P., Backhouse, C., Wyard-Scott, L., Musilek, P.: A swarm-based system for object recognition. Neural Netw. World **15**, 243–255 (2005)
4. Piatrik, T., Chandramouli, K., Izquierdo, E.: Image classification using biologically inspired systems. In: Proceedings of the 2nd International Mobile Multimedia Communications Conference MobiMedia'06, pp. 18–20 (2006)
5. Dorigo, M., Bonabeou, E., Theraulaz, G.: Inspiration for optimization from social insect behavior. Nature **406**, 39–42 (2000)
6. Dorigo, M., Stutzle, T.: Ant Colony Optimization. MIT Press, Cambridge (2004)
7. Maniezzo, V., Milandri, M.: An ant-based framework for very strongly constrained problems. In: Dorigo, M., et al. (eds.) Ants Algorithms. LNCS, pp. 222–227. Springer, Berlin (2002)
8. Cordon, O., Herrera, F., Stutzle, T.: A review on the ant colony optimization metaheuristic: basis, models and new trends. Mathw. Soft Comput. **9**(3), 141–175 (2002)
9. Galea, M.: Applying swarm intelligence to rule induction. M.Sc. Thesis, Division of Informatics, University of Edinburgh (2002)
10. Bonabeau, E., Dorigo, M., Theraulaz, G.: Swarm Intelligence: From Natural to Artificial Systems. Oxford University Press, New York (1999)
11. Liu, H., Motoda, H.: Feature Selection for Knowledge Discovery and Data Mining. Kluwer Academic Publishers, Boston (1998)
12. Klosgen, W., Zytkow, J.M.: Handbook of Data Mining and Knowledge Discovery. Oxford University Press, New York (2002)
13. Cios, K.J., Pedrycz, W., Wiswiniarski, R.: Data Mining Methods for Knowledge Discovery. Kluwer Academic Publishers, Boston (1998)
14. Weiss, S., Indurkhya, N.: Predictive Data Mining: A Practical Guide. Morgan Kaufmann Publishers, San Francisco (1998)
15. Rokach, L., Maimon, O.: Data Mining with Decision Trees. World Scientific Publishing, Singapore (2008)
16. Gong, S., McKenna, S., Psarrou, A.: Dynamic Vision: From Images to Face Recognition. Imperial College Press, London (1999)
17. Jahne, B., Haussecker, H., Geissler, P.: Handbook of Computer Vision and Applications. Academic Press, San Diego (1999)
18. Sonka, M., Hlavac, V., Boyle, R.: Image Processing, Analysis, and Machine Vision. Chapman & Hall Computing, New York (1993)
19. Bradski, G., Kaehler, A.: Learning OpenCV. O'Reilly, California (2008)
20. Jain, A., Flynn, P., Ross, A.: Handbook of Biometrics. Springer, New York (2008)
21. Whelan, P., Molloy, D.: Machine Vision Algorithms in Java: Techniques and Implementation. Springer, New York (2001)
22. Le, D., Satoh, S.: An efficient feature selection method for object detection. Pattern Recognition and Data Mining. LNCS, pp. 461–468. Springer, Berlin (2005)
23. Serre, T., Heisele, B., Mukherjee, S., Poggio, T.: Feature Selection for Face Detection. Massachusetts Institute of Technology, Cambridge (2000)
24. Lesk, A.: Introduction to Bioinformatics. Oxford University Press, New York (2002)
25. Silva, P., Hashimoto, R., Kim, S., Barrera, J.: Feature selection algorithms to find strong genes. Pattern Recognit. Lett. **26**, 1444–1453 (2005)
26. Baxevanis, A., Quellette, B.F.F.: Bioinformatics: A Practical Guide to the Analysis of Genes and Proteins. Wiley, New York (2001)
27. Riboni, D.: Feature selection for web page classification. In: Proceedings of the Workshop EURASIA-ICT 2002, pp. 473–477 (2002)

28. Sun, A., Lim, E., Ng, W.: Web classification using support vector machine. In: Proceedings of the WIDM'02, November, McLean, Virginia (2002)
29. Yang, Y., Pedersen, J.: A comparative study on feature selection in text categorization. In: Proceedings of the 14th International Conference on Machine Learning (ICML 1997), pp. 412–420 (1997)
30. Chen, J., Huang, H., Tian, S., Qua, Y.: Feature selection for text classification with naive bayes. Expert Syst. Appl. **36**, 5432–5435 (2009)
31. Erta, F.: Feature selection and classification techniques for speaker recognition. J. Eng. Sci. **7**(1), 47–54 (2001)
32. Ye, N.: The Handbook of Data Mining. Lawrence Erlbaum Associates, Mahwah (2003)
33. Abd-Alsabour, N., Randall, M., Lewis, A.: Investigating the effect of fixing the subset length using ant colony optimization algorithms for feature subset selection problems. In: Proceedings of the PDCAT, China (2012)
34. Han, J., Kamber, M.: Data Mining Concepts and Techniques, 1st edn. Morgan Kaufmann Publishers, San Francisco (2001)
35. Skillicorn, D.: Understanding Datasets: Datamining with Matrix Decomposition. Chapman & Hall/CRC, London (2007)
36. Hand, D., Mannila, H., Smyth, P.: Principles of Data Mining. Massachusetts Institute of Technology, Cambridge (2001)
37. Berry, M., Linoff, G.: Data Mining Techniques for Marketing, Sales, and Customer Relationship Management, 2nd edn. Wiley, New York (2004)
38. Leguizam'on, G., Michalewicz, Z.: A new version of ant system for subset problems. In: Angeline, P.J., Michalewicz, Z., Schoenauer, M., Yao, X., Zalzala, A. (eds.) Proceedings of Congress on Evolutionary Computation (CEC99), Washington DC, July 6–9. IEEE Press, (1999)
39. Dorigo, M., Birattari, M., Stutzle, T.: Ant colony optimization- artificial ants as a computational intelligence technique. IEEE Comput. Intell. Mag. 28–39 (2006)
40. Lee, K., Joo, J., Yang, J., Honavar, V.: Experimental comparison of feature subset selection using GA and ACO algorithm. In: Li, X., Zaiane, O.R., Li, Z. (eds.) ADMA 2006. LNAI, pp. 465–472. Springer, Berlin (2006)
41. Bello, R., Nowe, A., Caballero, Y., Gomez, Y., Vrancx, P.: A model based on ant colony system and rough set theory to feature selection. In: Proceedings of the GECCO'05. Washington (2005)
42. Jensen, R., Shen, Q.: Finding rough set reducts with ant colony optimization. In: Proceedings of the Workshop on Computational Intelligence, UK. 15–22 (2003)
43. Aghdam, M., Tanha, J., Naghsh-Nilchi, A., Basiri, M.: Combination of ant colony optimization and Bayesian classification for feature selection in a bioinformatics dataset. J. Comput. Sci. Syst. Biol. **2**(3), 186–199 (2009)
44. Trafalis, T.B., Ince, H.: Support vector machine for regression and applications to financial forecasting. In: Proceedings of the IEEE-INNS-ENNS International Joint Conference, vol. 6, pp. 348–353. IEEE Press (2000)
45. Durbha, S.S., King, R.L., Younan, N.H.: Support vector machines regression for retrieval of leaf area index from multi-angle imaging spectroradiometer. Remote Sens. Environ. **107**, 348–361 (2007)
46. Lui, B.: Web Data Mining. Springer, Berlin (2010)
47. Hsu, C.W., Chang, C.C., Lin, C.J.: A practical guide to support vector classification. Technical report, Department of Computer Science, National Taiwan University, Taipei, Available at http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf, (2003), viewed 25th March 2013
48. Wang, L., Fu, X.: Data Mining with Computational Intelligence. Springer, Berlin (2005)
49. Marsland, S.: Machine Learning: An Algorithmic Perspective, CRC Press-Taylor & Francis Group, (2009)
50. Miller, T.: Data and Text Mining- A Business Applications Approach. Pearson/ Prentice Hall, Upper Saddle River (2005)
51. Novat, A.: On the role of feature selection in machine learning. Ph.D Thesis, Hebrew University, Israel (2006)

52. http://www.cs.waikato.ac.nz/ml, viewed 25th March 2013
53. Amasyali, M., Ersoy, O.: A comparative review of regression ensembles on drug design datasets. Turkish J. Electr. Eng. Comput. Sci. 1–17 (2013)
54. R: A Language and Environment for Statistical Computing 2006 [http://www.R-project.org]. R Foundation for Statistical Computing, Vienna, Austria
55. http://www.cs.waikato.ac.nz/ml/weka, viewed 25th March 2011
56. Shen, Q., Jiang, J.H., Tao, J.C., Shen, G.L., Yu, R.Q.: Modified ant colony optimization algorithm for variable selection in QSAR modeling: QSAR studies of cyclooxygenase inhibitors. J. Chem. Inf. Model. **45**, 1024–1029 (2005)
57. Izrailev, S., Agrafiotis, D.: Variable selection for QSAR by artificial ant colony systems. SAR QSAR Environ. Res. **13**, 417–423 (2002)
58. Gunturi, S., Narayanan, R., Khandelwal, A.: In silico ADME modeling 2: computational models to predict human serum albumin binding affinity using ant colony systems. Biorgan. Med. Chem. **14**, 4118–4129 (2006)
59. Shamsipur, M., Zare-Shahabadi, V., Hemmateenejad, B., Akhond, M.: Ant colony optimization: a powerful tool for wavelength selection. J. Chemom. **20**, 146–157 (2006)
60. Palanisamy, S., Kanmani, S.: Artificial bee colony approach for optimizing feature selection. Int. J. Comput. Sci. Issues **9**(3), 432–438 (2012)
61. Shamsipur, M., Zare-Shahabadi, V., Hemmateenejad, B., Akhond, M.: An efficient variable selection method based on the use of external memory in ant colony optimization: application to QSAR/QSPR. Anal. Chim. Acta **646**, 39–46 (2009)
62. Vieira, S.M., Sousa, J.M.C., Runkler, T.A.: Two cooperative ant colonies for feature selection using fuzzy models. Expert Syst. Appl. **37**, 2714–2723 (2010)

Ant Colony for Locality Foraging in Image Enhancement

Gabriele Simone, Davide Gadia, Ivar Farup and Alessandro Rizzi

Abstract This chapter presents a spatial color algorithm called Termite Retinex and the problem of filtering locality in this family of algorithms for image enhancement, inspired by the human vision system. The algorithm we present is a recent implementation of Retinex with a colony of agents, which uses swarm intelligence to explore the image, determining in this way the locality of its filtering. This results in an unsupervised detail enhancement, dynamic range stretching, color correction, and high dynamic range tone rendering. In the chapter we describe the characteristic of glocality (glocal = global + local) of image exploration, and after a description of the Retinex spatial color algorithm family, we present the Termite approach and discuss results.

Keywords Retinex · Spatial color algorithm · Artificial termites · Image enhancement

1 Introduction

The term "image enhancement" refers to a multitude of algorithms and approaches. They differ in the way they realize the enhancement, and as a consequence, different methods can be considered in order to measure their success. For example, one of the subgoals of image enhancement is the adjustment of visual contrast, but no agreed

G. Simone (✉) · D. Gadia · A. Rizzi
Department of Computer Science, Università degli Studi di Milano, Milano, Italy
e-mail: gabriele.simone@hig.no

D. Gadia
e-mail: davide.gadia@unimi.it

A. Rizzi
e-mail: alessandro.rizzi@unimi.it

G. Simone · I. Farup
The Norwegian Colour and Visual Computing Laboratory, Faculty of Computer Science and Media Technology, Gjøvik University College, Gjøvik, Norway
e-mail: ivar.farup@hig.no

S. Dehuri et al. (eds.), *Multi-objective Swarm Intelligence*,
Studies in Computational Intelligence 592, DOI 10.1007/978-3-662-46309-3_5

measure of visual contrast in digital images exists. In particular, an important part in image enhancement is played by the correction of color. Many image enhancement algorithms mimic the Human Visual System (HVS) since it accomplish powerful adjustment automatisms, like lightness constancy and color constancy. Lightness constancy enables a stable perception of the scene regardless of changes in the mean luminance intensity, while color constancy enables a stable perception of the scene regardless of changes in the color of the illuminant [1]. The difference among taking inspiration from HVS or trying to mimic it is relevant [2]. Any model of vision needs a careful calibration of input and output information up to single pixel level in order to be used for the simulation of human vision. To ease these constraints, HVS could be considered just as inspiration for image enhancement, with the goal to meet observers preferences rather than to match observer perceptual response [1, 3, 4]. This is the approach we have followed in the design and implementation of the Termite Retinex (TR) [5] method presented in this chapter, which uses swarm intelligence techniques to tune the locality of Retinex filtering. Termite Retinex belongs to the MI-Retinex [6] family of implementations of the Retinex theory, proposed by Land and McCann [7, 8] as a model of color sensation of HVS.

All these algorithms belong to a family of computational models inspired by the HVS, called Spatial Color Algorithms (SCAs) [2]. The visual evidence at the basis of SCAs is that color sensation does not depend on the point-wise value of the stimulus, but on the spatial arrangement of the stimuli in the scene. Same point-wise values of radiance can originate a completely different visual response according to the image content [9]. This mechanism is also at the basis of several visual illusions [10–12]. To implement this important principle, SCAs perform spatial comparisons among the pixels in the input image. SCAs share a common structure of two phases. In the first phase each pixel is computed according to the spatial distribution of the other pixels in the image (all, or some of them, as prescribed by each implementation). To form the output image, the matrix of values, spatially computed in the first phase, is scaled in the second phase onto the available output dynamic range according to the model goal. SCAs has been proven to be able to obtain high preferences scores in terms of user preferences/pleasantness [3, 4].

In this chapter we present a brief overview of Retinex theory and models in Sect. 2. In Sect. 3 we present Termite Retinex, and in Sect. 4 we analyze the effect of swarm intelligence techniques in the exploitation of image locality. Finally, in Sect. 5 we present the properties of Termite Retinex and afterwards in Sect. 6 conclusions are drawn.

2 Retinex Theory

The term Retinex has been coined by Land and McCann [7, 8, 13] referring to the roles that both retina and cortex play in human vision. Land and McCann report [8]:

> Sensations of color show a strong correlation with reflectance, even though the amount of visible light reaching the eye depends on the product of reflectance and illumination. The visual system must achieve this remarkable result by a scheme that does not measure flux. Such a scheme is described as the basis of retinex theory. This theory assumes that there are three independent cone systems, each starting with a set of receptors peaking, respectively, in the long-, middle-, and short-wavelength regions of the visible spectrum. Each system forms a separate image of the world in terms of lightness that shows a strong correlation with reflectance within its particular band of wavelengths. These images are not mixed, but rather are compared to generate color sensations. The problem then becomes how the lightness of areas in these separate images can be independent of flux.

The term "lightness", for Land and McCann, is associated to the brightness sensation in each channel. Edges among adjacent areas of an image, and lightness ratio between two areas, play a fundamental role in the final appearance at each point.

Formally, Retinex is based on computing the relative channel lightness (L) at a point i as the mean value of the relative channel lightnesses (l) computed along N random paths from point j to the point i (Fig. 1a):

$$\frac{L^i = \sum_{h=1}^{N} l_h^{i,j}}{N} \tag{1}$$

where,

$$l_h^{i,j} = \sum_{x=j}^{i} \delta log\left(\frac{I_{x+1\in path}}{I_{x\in path}}\right), \tag{2}$$

where I_x is the lightness intensity of the pixel x, and $I_x + 1$ is the lightness intensity of the pixel $x + 1$ and h is indicating the path. An example of Random paths is shown in Fig. 1. The reset mechanism δ forces the chain of ratios to restart from the unitary value, considering the lightness value found at the reset point a new local reference white [6]:

$$\delta = \begin{cases} 1 & \text{if } \left| log\left(\frac{I_{x+1\in path}}{I_{x\in path}}\right)\right| > T \\ 0 & \text{if } \left| log\left(\frac{I_{x+1\in path}}{I_{x\in path}}\right)\right| \leq T \end{cases} \tag{3}$$

where T is a defined threshold.

The three color channel R, G, and B are processed independently and thus the lightness is represented by the triplet (L_R, L_G, L_B) of lightness values in the three chromatic channels.

The original formulation of Retinex does not provide a description of how to generate the random paths. This is a critical point: in Retinex, appearance is calculated using ratios, and the ratios are applied to samples in the image. Therefore, changing the method to create the random paths (i.e., their structure), we change the way locality is considered and introduced in the Retinex elaboration.

There is a large literature available on the different approaches applied for the exploration of locality in the implementation of Retinex models. Here we present

a brief overview of the main approaches and implementations, for an exhaustive overview and analysis we suggest the reader to consult [6, 14].

We can identify two different approaches in Retinex implementations: sampling and integrating.

In the sampling approach, Retinex elaboration in each pixel is applied using a subset of samples in its neighborhood, selected along paths starting from the considered pixel. The Retinex implementations based on sampling follow strictly the original model by Land and McCann, and they differ mainly in the method and geometry of the paths (e.g. spiral [15], or double spiral [16]). In other implementations [10, 15, 17], path exploration has been addressed by means of multilevel techniques, in which a multi-resolution pyramid of the input image is created, and Retinex ratio- reset operation, applied to the neighborhood of the pixel, is iterated from the coarsest to the finest level.

An important distinction among the sampling implementations regards the way pixel computing is scheduled across the path: the implementations closer to the original model update each pixel along the path, while the implementations in the MI-Retinex [6] family of algorithms apply the update of the pixel only at the end of each path. In Brownian Retinex [11], proposed by Marini and Rizzi, paths generation is based on an approximation of Brownian paths (Fig. 1b), inspired by the results of neuro-physiological research about human cortical vision areas, where the distribution of receptive fields centroids mimics a Brownian path, as demonstrated in many experiments by Zeki [18]. Brownian motion has been also investigated in [19] by Montagna and Finlayson, where it is proposed an efficient algorithm for the generation of pseudo-Brownian paths (Fig. 1c), applied to the original Retinex approach.

The sampling approach has been investigated in [20], where the authors have proved the redundancy of the MI-Retinex path-wise approach, demonstrating that, along a path, only the pixel with the highest value is relevant for the lightness calculation. As a consequence, they have proposed a simplified reformulation of MI-Retinex models and a novel implementation, called RSR (Random Spray Retinex) [21] where paths are replaced by two-dimensional point sampling (called sprays) across the image (Fig. 2), and the ratio-reset operation is replaced by a single ratio between the input pixel and the maximum value found in the spray. STRESS (Spatio-Temporal Retinex-inspired Envelope with Stochastic Sampling) [22] uses the same spray approach of RSR, but with an alternative technique based on the consideration for each pixel not only of the local maximum (reference white), but also of the local minimum (reference black), in each chromatic channel. RSR approach has been used also in the design of the RACE algorithm [23], which combines RSR with the ACE computational model [12], another member of the SCA family.

The integrating approach follows a modification proposed by Land [24] in one of his latest work, in which he has proposed to compute the lightness in a pixel using the average of a surround of the pixel, weighted using the square distance from the center. This approach is considerably different from the original Retinex formulation since it uses no more the reset mechanism. This absence, together with the average on a fixed region, makes it a Gray World type of algorithm, and not a

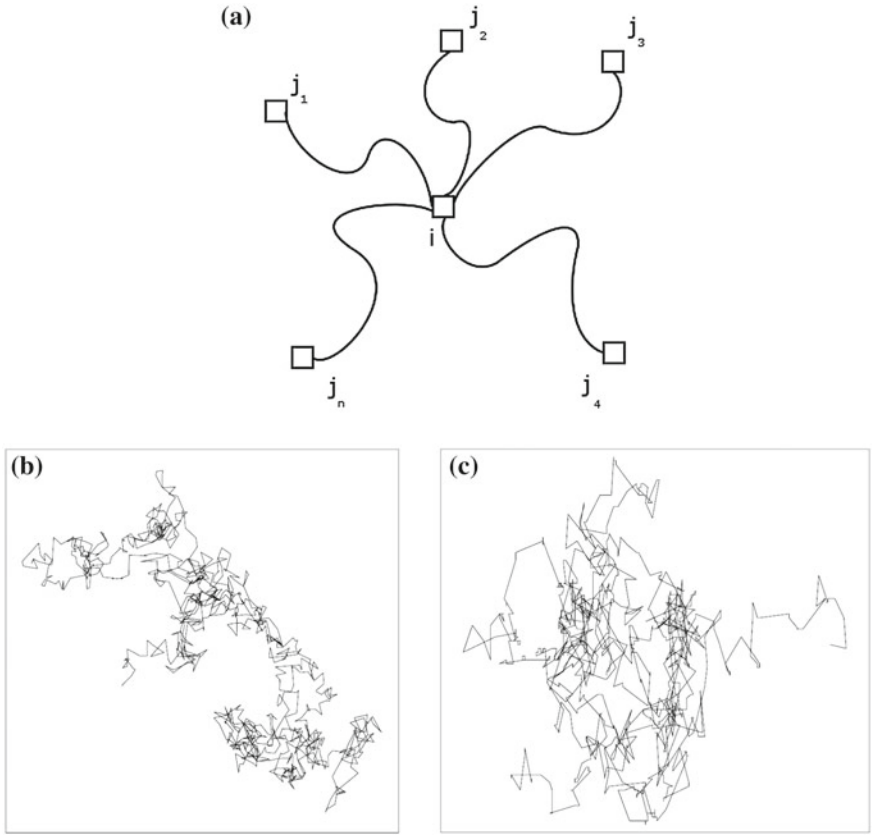


Fig. 1 Representation of N random paths. Traditional Brownian motion investigation proposed by Marini and Rizzi on the *bottom-left*, Pseudo-Brownian motion investigation proposed by Montagna and Finlayson. Figure from [19]. **a** Random paths. **b** Brownian paths. **c** Pseudo-Brownian paths

White Patch one like the original Retinex. In this model, locality is considered on the basis of the choice of the size of the surround, and of its sampling. Using a small surround leads to a significant increase in local contrast but induces halo artifacts along high contrast edges and an overall effect of desaturation in final color rendition. On the contrary, adopting a larger surround reduces the artifacts, but provides less increase in local contrast. Jobson et al. [25, 26] have taken inspiration from this center/surround approach to develop the Multi Scale Retinex (MSR), in which they have computed the weighted average of the surround by convolving the image with a normalized kernel function. To overcome halo problems and color desaturation, they have refined the algorithm by introducing a multilevel approach, and a color restoration stage, respectively. Different modifications to MSR have been proposed, like e.g. Ramponi et al. [27].

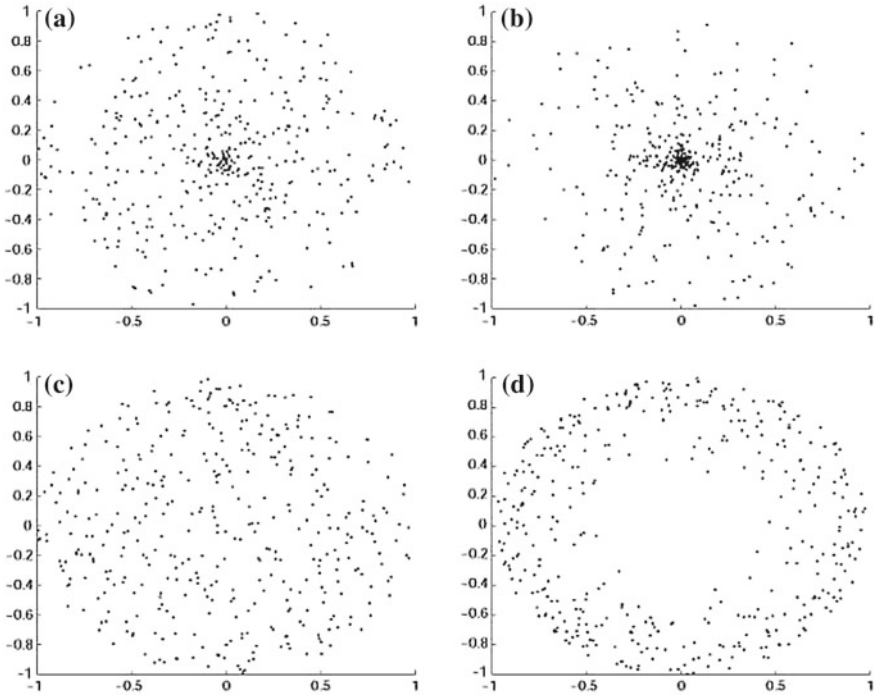


Fig. 2 Four examples of random sprays used by RSR. The density of a spray depends on three parameters: the radius of the spray, the radial density function, and the number of pixels per spray. Figure from [21]

The Retinex differential mathematical approaches belong to the second family, since their locality depends on the kernel they use to implement their variational approach, without the use of path sampling (e.g. [28, 29]).

3 Termite Retinex

"Termite Retinex", in short TR, is a path based MI-Retinex implementation that takes into account a swarm intelligence model. TR is directly derived from the Ant Colony Optimization (ACO) model proposed by Dorigo et al. [30, 31] for the Traveling Salesman Problem, which we briefly recall in order to be able to introduce TR.

The Traveling Salesman Problem (TSP) is probably the most famous NP-hard problem in combinatorial optimization and theoretical computer science. Consider a salesman who must visit n cities labeled $v_0, v_1, \ldots, v_n$. The salesman starts in city v_0, his home, and he wants to find an ordered tour, in which he can visit all

the other cities only once and come back home, traveling as little total distance as possible [32]. In other words:

Definition 1 Given a set of n cities and a pairwise distance function d(r, u), is there a tour of length $\leq D$?

In the original Ant Colony System [30], when cities are on a plane, and a path (edge) exists between each pair of cities (i.e., the TSP graph is completely connected), an artificial ant k in city r chooses the city s to move to among those which do not belong to its working memory M_k by applying the following probabilistic formula:

$$p_k(r,s) = \begin{cases} \frac{(\tau_{r,s})^{\alpha}(\eta_{r,s})^{\beta}}{\sum_{u \notin M_k}(\tau_{r,s})^{\alpha}(\eta_{r,s})^{\beta}} & \text{if } s \notin M_k \\ 0 & \text{otherwise} \end{cases}, \tag{4}$$

where $\tau_{r,u}$ is the amount of pheromone trail on edge (r, u), $\eta_{r,u}$ is a heuristic function called visibility, which is the inverse of the distance between cities r and u and, α and β are parameters that allow a user to control the importance of the trail versus the visibility. The memory M_k is the taboo list of the kth ant, which contains the cities that it has already visited. City s is inserted in the list when the ant transits from city r to city s. The choice criteria of the parameters α and β can differ widely according to the problem for which the ACO is used. A guideline on how to choose the values of the different parameters for the TSP problem can be found in [31].

Essentially, from the origin of the ACO model to its consecutive works, three ideas from natural ant behavior are transferred to the artificial ant colony:

1. The preference for paths with a high pheromone level;
2. The higher rate of growth of the amount of pheromone on shorter paths;
3. The trail mediated communication among ants.

In TR a convex combination of Eq. 4 is derived with a different leading principle, where an artificial termite k in pixel r chooses to move to the pixel s among those that belong to the 8-neighborhood N_8 and that do not currently belong to its working memory M_k by applying the following probabilistic formula:

$$p_k(r,s) = \begin{cases} \frac{(\theta_s)^{\alpha}(c_{r,s})^{\beta}}{\sum_{u \notin M_k \text{ and } u \in N_8}(\theta_u)^{\alpha}(c_{r,s})^{\beta}} & \text{if } s \notin M_k \text{ and } s \in N_8 \\ 0 & \text{otherwise} \end{cases}, \tag{5}$$

where θ_u is the amount of poison on pixel u, $c_{r,u}$ is the bilateral distance between pixels r and u and, α and β are parameters weighting the importance of the poison versus the closeness, which is directly related to the brightness of the pixel. In the case all the surrounding pixels have the same probability, one pixel is drawn randomly with uniform probability. In our model the memory M_k is the taboo list of the kth termite, which contains the coordinates of the pixels that the termite has already visited. This list is updated inserting the coordinates of pixel s when the termite transits from pixel r to pixel s. The poison is the inverse of the amount of pheromone, once a termite has transited on pixel u, the quantity of poison on pixel u is updated as follows:

$$\tau_u = \tau_u + Q \tag{6a}$$

$$\theta_u = \frac{1}{\tau_u} \tag{6b}$$

where Q is a chosen amount of poison with $0 < Q \leq 1$. As in Retinex model, each pixel is treated independently, after processing a pixel, the amount of poison at all pixels must be reset to the initial value of 1 before starting processing a new pixel.

Therefore, artificial termites are essentially governed by three principles:

1. The preference for pixels with a low poison level. Divergence is required, in order to explore different areas of the image: the poison acts as a repulsion "force", inverse of the attraction force pheromone in ACO;
2. The growth of the amount of poison on visited pixels. The higher the quantity of the poison added on a pixel, the stronger the divergence behavior of the termites. Furthermore the poison is a feature that prevents the paths from a complete randomness since it affects the glocality, it can be used as a tuning parameter in order to obtain different spatial effects;
3. The length of the path, which also affects the glocality of the filtering. In particular, a termite should never travel across the whole set of pixels in the image, because it would lead the filtering to the global white asymptote and thus a to a global white normalization of the image content.

Furthermore, the following constraints are required to be a novel tool for exploring glocality:

1. A termite can choose to move only to one of the 8-neighboring pixels, jumps forbidden. By exploring larger neighborhoods and allowing the termite to "jump" might lead to not discovering the proper local reference white;
2. The choice of a pixel is based on its distance and intensity value: the visibility η is substituted with the bilateral distance c as defined below, that we will refer to as "closeness". The use of the bilateral distance is known to be a suitable tool for edge-preserving [33] and in TR is justified by the fact that the presence of halos are reduced in the final image with respect to the simple distance based only in intensity values (Fig. 3).

The bilateral distance $c_{r,u}$ is defined as follows:

$$c_{r,u} = \frac{d_e + d_v}{\sqrt{2}} \tag{7a}$$

$$d_e = \sqrt{(x_r - x_u)^2 + (y_r - y_u)^2} \tag{7b}$$

$$d_v = |I(x_r, y_r) - I(x_u, y_u)| \tag{7c}$$

where d_e and d_v are the distance in coordinates and in intensity values respectively, I is the image channel and (x, y) are the coordinates of the pixels. When a termite has

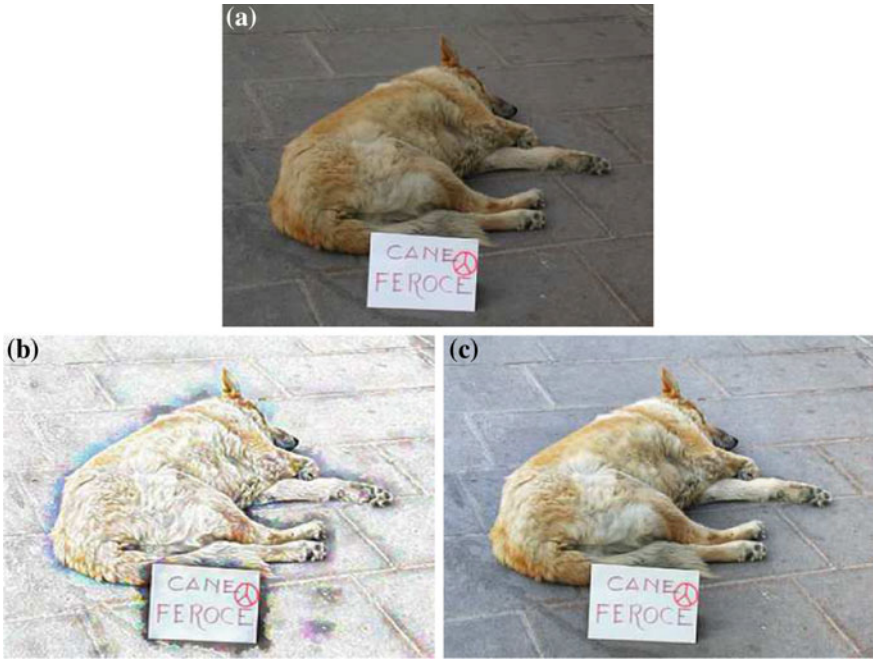


Fig. 3 Halos appearing using only the simple distance on intensity values on the *bottom-left*, using bilateral distance on the *bottom-right*. **a** Original. **b** Only intensity values. **c** With bilateral distance

to choose among eight neighboring pixels with same intensity values, the preference goes to one of the four diagonal pixels with respect to the vertical and horizontal ones. This phenomenon of preference for the diagonal pixel gradually disappears with a high number of termites and with the participation of the poison.

At the current stage, the reader may wonder why the proposed method is not called "Ant Retinex". Two reasons motivate the choice of "Termite Retinex". Our artificial termites attempt an eager exploration in search of the local reference white, in analogy with biological worker termites, also known as "white ants", which undertake the labors of foraging. As well as the behavior of biological termites change with the evolving of the nest structure, our artificial termites change their behavior with the exploring of the image content.

4 Termite Retinex Locality

TR glocality is controlled by the following five parameters:

- α and β: they determine the trade off the quantity of poison found on the pixel and between the brightness of a pixel to choose; In the case of high values of β,

a pixel is chosen, even if already visited by another termite. On the contrary high values of α force a termite to choose another direction;
- k (number of termites): it determines the size of the swarm, specifically higher is the number of termites, higher is the chance to find the proper local reference white;
- N_s (length of the path): it determines how far a termite should travel, specifically how wide is the area of the image to be explored by the swarm;
- Q (quantity of poison): it determines the strength of the divergence behavior of the swarm.

In TR, the values of the parameters, do not change during pixel recomputation, they are kept constant for the whole processing of the image.

Essentially, α and β determine the behavior of a single termite, k, N_s, and Q influence the glocality and the final visual sensation of the image.

Figure 4 shows the behavior of the termites with two different configurations of α and β, specifically $\alpha = 0.9$ and $\beta = 0.1$ in Fig. 4a, and $\alpha = 0.1$ and $\beta = 0.9$ in Fig. 4b. The higher the value of α and the lower the value of β, the wider the area explored by the termites. Viceversa, the lower the value of α and the higher the value of β, the smaller the area of the image explored, making the termites to forage the

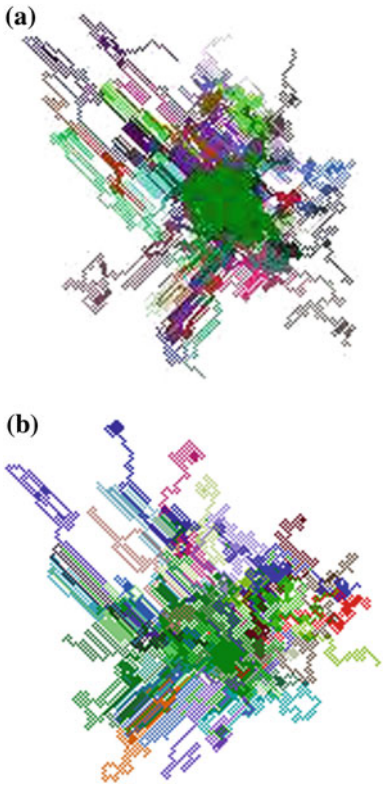


Fig. 4 Termite investigation with two different configurations of α and β and quantity of poison Q = 1. The path of each termite is distinguished by a different color. A high value of α and a low value of β make the termite swarm to explore a wide area of the image as shown on the *top*, while a low value of α and a high value of β make the termite swarm to explore a smaller area of the image as shown on the *bottom*. **a** $\alpha = 0.1$, $\beta = 0.9$. **b** $\alpha = 0.9$, $\beta = 0.1$

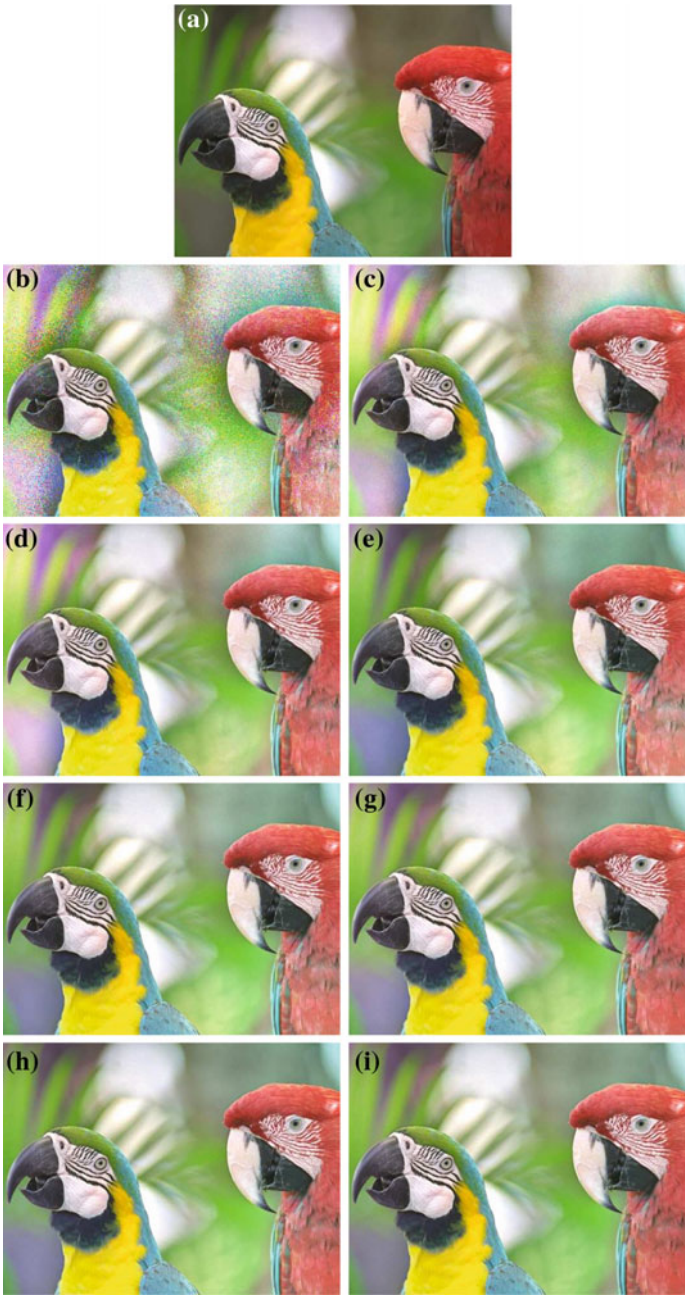


Fig. 5 Termite Retinex filtering with $N_s = 500$, $\alpha = 0.1$, $\beta = 0.9$, and $Q = 1$, with increasing number of termites k. A few number of termites may lead to chromatic noise while a too large number of termites may lead to a low contrast enhancement and loss of details. **a** Original. **b** $k = 1$. **c** $k = 10$. **d** $k = 50$. **e** $k = 100$. **f** $k = 250$. **g** $k = 500$. **h** $k = 750$. **i** $k = 1{,}000$

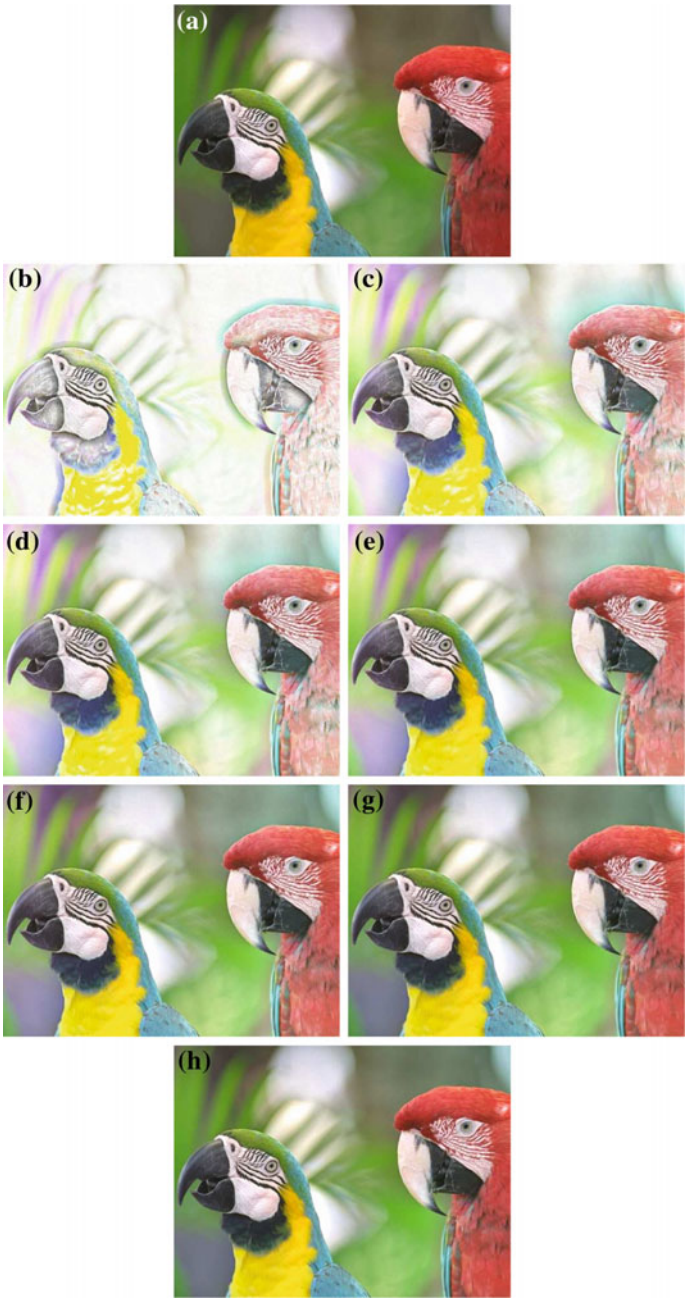


Fig. 6 Termite Retinex filtering with $k = 500$, $\alpha = 0.1$, $\beta = 0.9$, and $Q = 1$, with increasing length of the path N_s. A too short path may lead to edge overenhancing while a too long path may lead in a loss of the local effect. **a** Original. **b** $N_s = 10$. **c** $N_s = 50$. **d** $N_s = 100$. **e** $N_s = 250$. **f** $N_s = 500$. **g** $N_s = 750$. **h** $N_s = 1{,}000$

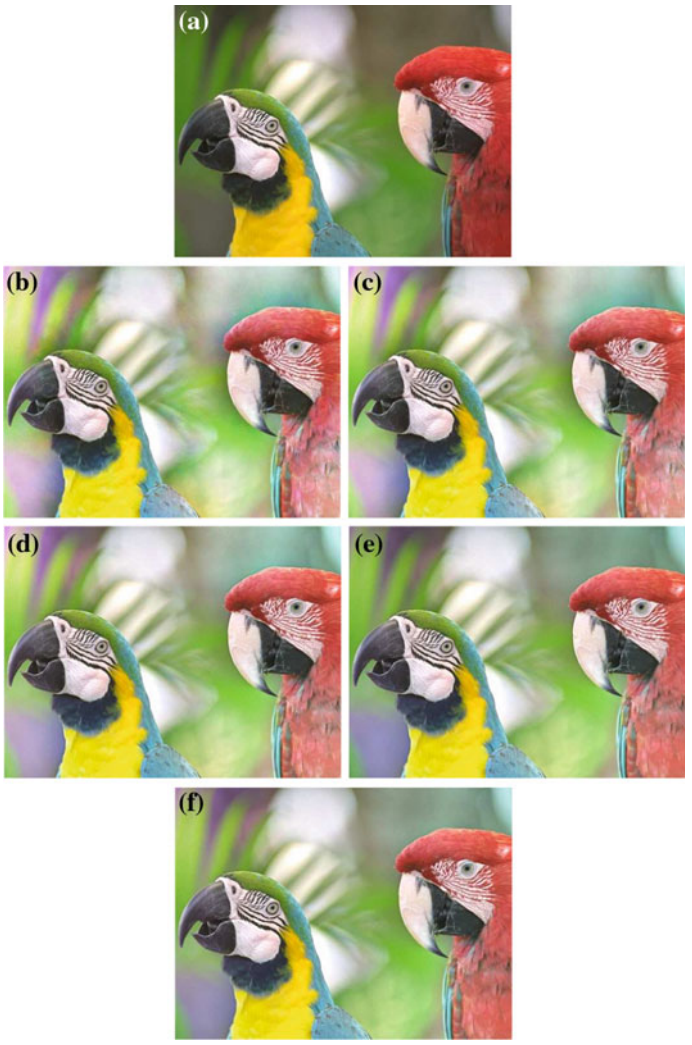


Fig. 7 Termite Retinex filtering with $k = 500$, $\alpha = 0.1$, $\beta = 0.9$, and $N_s = 500$, with increasing quantity of poison Q. Very low quantities of poison may lead to the presence of halos. **a** Original. **b** $Q = 10^{-4}$. **c** $Q = 10^{-3}$. **d** $Q = 10^{-2}$. **e** $Q = 10^{-1}$. **f** $Q = 1$

same local reference white. The particular case, with $\alpha = 0$ and $\beta = 0$, yields to complete random paths.

Figure 5 shows an example of filtering with $\alpha = 0.1$ and $\beta = 0.9$, constant path length and quantity of poison, but varying number of termites. As it can be observed in Fig. 5c, d, too few termites, due to insufficient image sampling, can lead to a high chromatic noise. On the other side a large number of termites leads to low contrast enhancement and loss of details. In the proposed example this can be particularly

seen in the breast of the red parrot, where the details of the feathers are lost in Fig. 5h, i with respect to Fig. 5e–g.

Figure 6 shows the previous example of filtering, but this time, varying the length of the path N_s and keeping constant all the other parameters. Too short paths lead to edge overenhancing and the presence of halos as it can be observed in Fig. 6b–d. On the contrary, too long paths result in a loss of the local effect and in not very bright images as it can be seen in Fig. 6g, h with respect to Fig. 6e, f. The specific case with the longest path $N_s = n$ with $n = width \times height$, or in other words making the termites visiting all pixels, yields to global white normalization. This normalization, known also as "White Patch" [34], performs a global enhancement of the three color channel separately dividing each pixel for the maximum value of the channel. It is important to recall that differently from Von Kries classic White Patch, Retinex normalization to the channel maxima is local for each pixel.

As the last parameter to discuss, we present in Fig. 7 an example of filtering varying only the quantity of poison Q. Very low quantities of poison lead to the presence of halos as it can be observed in Fig. 7b–d with respect to Fig. 7e, f. This is due by the fact that lowering the quantity of poison, reduces the divergence of the swarm making the termites foraging the local reference white in a smaller area. This may lead also the termites to discover the same local maxima as reference white.

Previous studies has shown that $\alpha = 0.1$ and $\beta = 0.9$, $k \approx \min(height, width)$, $N_s \approx 70\,\%$ of the length of the diagonal, and $Q = 1$ are in line with observer preference [5].

5 Termite Retinex Properties

5.1 Computational Complexity

The computational complexity of TR is given by:

$$O(k \cdot N_s \cdot n) \tag{8}$$

where k is the number of termites, N_s is the length of the path, and n is the number of pixels in the image. TR has the same computational complexity of other SCAs, such as RSR [21] or STRESS [22] which have a computational complexity of $O(NMn)$, where N is the number of iterations, M is the number of samples and n is the number of pixels in the image.

In the last years, Retinex algorithms have been implemented using different programming languages and environments (like e.g. Matlab [17, 19] and C++ [21, 22]), and on different architectures and devices (like e.g. on GPU [35] and in digital cameras [36]). In all these cases, the way the algorithm is implemented effects relevantly its overall performance.

We give some remarks for an efficient TR implementation:

- Each color channel is treated independently like in the original model, thus each channel can be processed in parallel.
- As well, each pixel in each color channel is recomputed independently, thus all pixels of the image can be processed in parallel.
- Each termite is an autonomous worker, so each termite foraging can be parallelized.
- Each termite allocates its taboo list. The higher the number of termites and the length of the path, higher the memory consumption.
- The higher the length of the path, the higher becomes the computational time.

5.2 Dynamic Range Stretching

An important property of TR in line with other SCAs is the ability of performing dynamic range stretching. TR, as many other Retinex implementations and SCAs, performs dynamic range stretching processing all the pixel-value population but also taking into account the local spatial relationships of the image content, which control appearances and thus the final visual sensation of the image [6].

Figure 8 presents an example of dynamic range stretching on a grey scale image. As it can be observed, the original image presents a reduced dynamic range, with

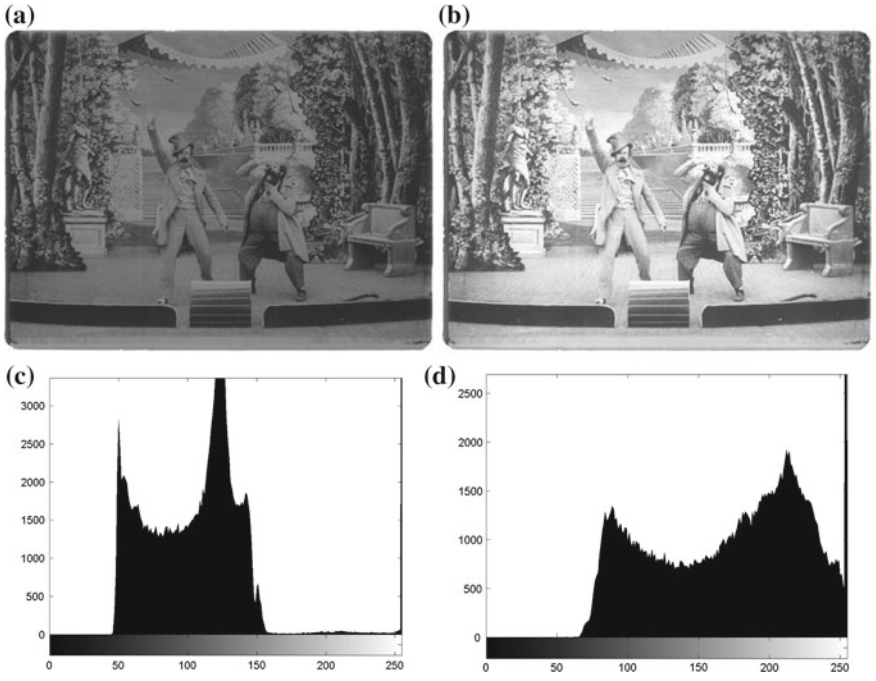


Fig. 8 An example of unsupervised dynamic range stretching on grey scale image. **a** Original. **b** Termite Retinex. **c** Original histogram. **d** Termite Retinex histogram

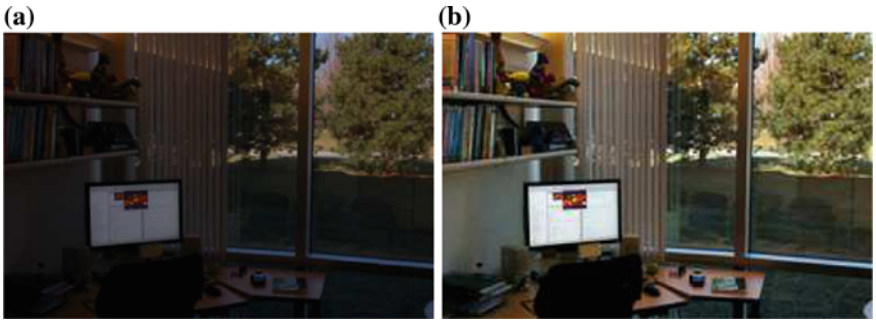


Fig. 9 An example of Termite Retinex unsupervised dynamic range stretching on color image. **a** Original. **b** Termite Retinex

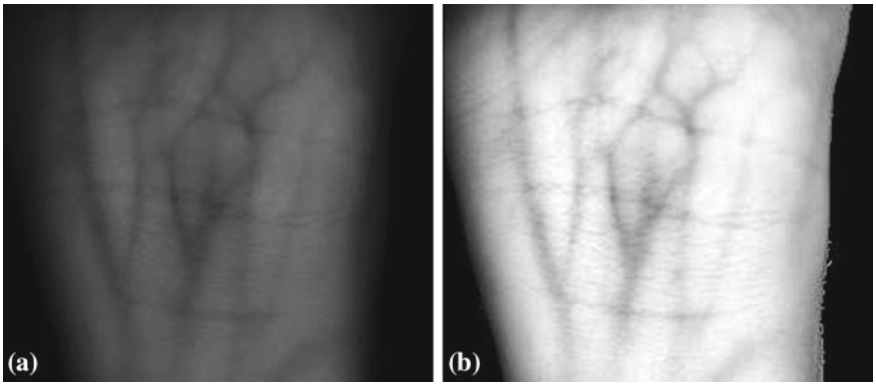


Fig. 10 An example of Termite Retinex unsupervised enhancement of a medical image. Original image provided with courtesy by Daniel Hartung [37]. **a** Original. **b** Termite Retinex

the most of the pixel-value population in the range of neutral grey and lower. TR filtering stretches and enhances the used dynamic range, and the resulting pixel-value population is centered in higher ranges.

Likewise, TR is able to perform dynamic range stretching processing on color images (Fig. 9) and on medical images (Fig. 10).

5.3 Color Constancy and Color Correction

Color Constancy is a complex issue largely debated in the imaging community [1]. In digital imaging literature, there are two different approaches to color constancy: Computer Vision CC (in some works referred as Computational CC, but we prefer to refer it as CV CC since both CCs have computational aspects), in short CV CC and Human Vision CC, in short HV CC. They have distinct goals in modeling this

Fig. 11 An example of Termite Retinex unsupervised color correction on Lena portrait. **a** Original. **b** Termite Retinex

phenomenon and thus different kind of outcomes are expected, and different measures of performance are required.

Computer Vision CC has the goal of separating illuminant from reflectance or alternatively estimating the physical reflectance of objects in different illuminants, or alternatively estimating the illuminant spectral or colorimetric component. This is a well-known ill-posed problem [38], thus these algorithms need constraints or assumptions on the scene content, illumination or geometry. In any case, CV CC aims to cancel totally the interaction between reflectance and illumination.

In Human Vision CC, the illuminant component is not totally canceled, it generates appearances that are close to reflectance, but with significant departures. These departures serve as important signatures of the underlying visual mechanisms [6]. For many years these two variables, reflectance and appearance, have been treated as an unique correlated feature. This has been proven to be an incorrect assumption [1, 39]. HV CC aims at computing appearance and algorithms belonging to the Retinex family, and thus TR, share this approach attempting to mimic the response of human vision.

Figure 11 shows the TR effect of unsupervised color cast reduction on the classic Lena portrait.

5.4 HDR Tone Rendering

As every SCA algorithm, TR can be used as a tone renderer for High Dynamic Range (HDR) images [6]. HDR images are formed by values that span over a range much wider than the range allowed by the visualization devices. For these images a tone rendering operation is necessary to display all the values preserving as much as

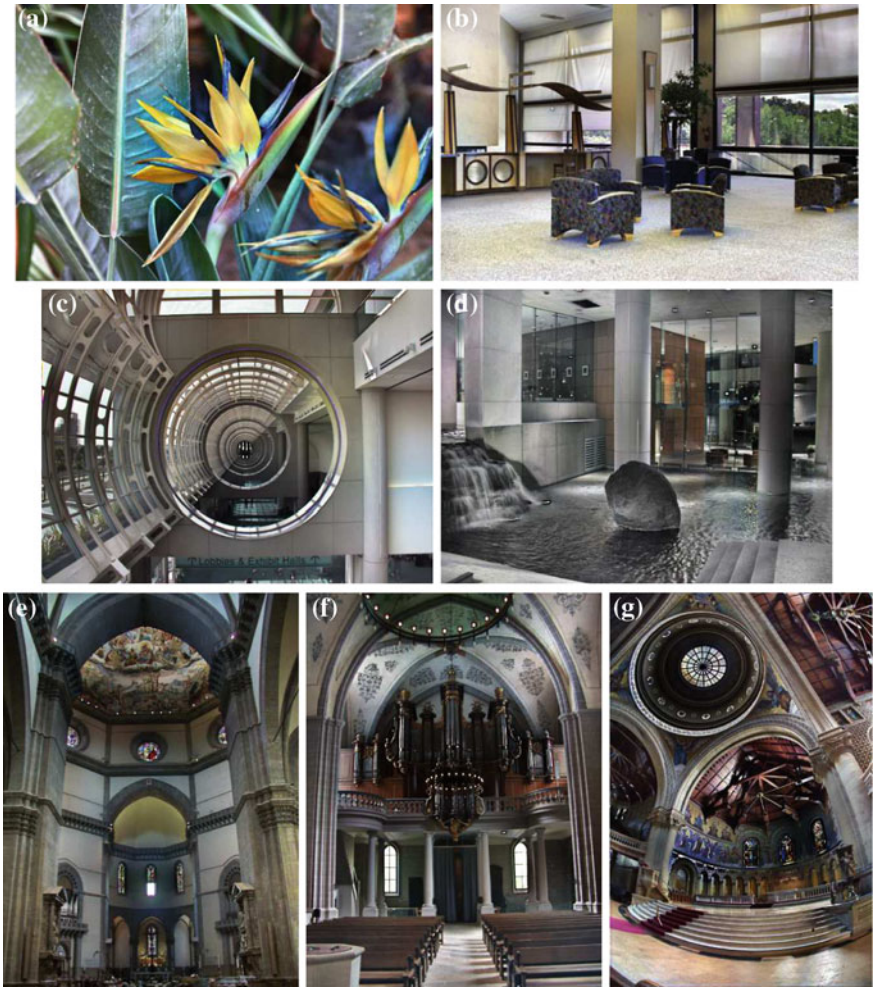


Fig. 12 Examples of unsupervised tone rendering of HDR images. **a** Paradise flower. **b** Chairs. **c** Convention center. **d** Vancouver building. **e** Duomo. **f** StFrancis. **g** Memorial. *Source*: web

possible their original appearance. Figure 12 shows some examples of the application of TR as a tone renderer for HDR images from the web.

6 Conclusions

In this chapter, we have presented an example of swarm intelligence applied for color and contrast enhancement of digital images, inspired by the Human Vision

System. In particular, we have presented a Retinex implementation, called Termite Retinex, in which the glocality (glocal = global + local) of image exploration is exploited through a colony of agents. In Termite Retinex, the idea of paths (the basis of the sampling approach of the original Retinex model) is reconsidered from the point of view of the Ant Colony Optimization model: the local filtering of Retinex is investigated through a set of "termites" moving inside the image. By tuning a small set of parameters, we can control the overall behavior of the swarm and therefore the final effect of the Retinex filtering. We have presented a detailed explanation of the swarm intelligence model used in Termite Retinex, and a discussion of its parameters. To demonstrate the capabilities of Termite Retinex, we have shown different examples and applications like unsupervised image enhancement, dynamic range stretching, color correction, and high dynamic range tone rendering.

References

1. McCann, J.J., Parraman, C., Rizzi, A.: Reflectance, illumination, and appearance in colorconstancy. Front. Psychol. **5**(5), (2014)
2. Rizzi, A., McCann, J.J.: On the behavior of spatial models of color. In: IS&T/SPIE Electronic Imaging, vol. 6493, p. 649303. San Jose, California, USA, January 2007
3. Parraman, C., Rizzi, A.: Searching user preferences in printing: a proposal for an automatic-solution. In: Printing Technology SpB06, St. Petersburg, Russia, June 2006
4. Parraman, C., Rizzi, A.: User preferences in color enhancement for unsupervised printing-methods. In: SPIE, vol. 6493, pp. 64930U–64930U-11 (2007)
5. Simone, G., Audino, G., Farup, I., Albregtsen, F., Rizzi, A.: Termite Retinex: a new implementation based on a colony of intelligent agents. J. Electron. Imaging **23**(1), 013006-1-13 (2014)
6. McCann, J.J., Rizzi, A.: The Art and Science of HDR Imaging. Wiley, New York (2011). ISBN: 978-0-470-66622-7
7. Land, E.H.: The Retinex. Am. Sci. **52**, 64–247 (1964)
8. Land, E.H., McCann, J.J.: Lightness and Retinex theory. J. Opt. Soc. Am. **61**(1), 1–11 (1971)
9. McCann, J., McKee, S., Taylor, T.: Quantitative studies in Retinex theory. A comparison between theoretical predictions and observer responses to the color Mondrian experiments. Vis. Res. **16**(5), 445–458 (1976)
10. McCann, J.J.: Lessons learned from Mondrians applied to real images and color gamuts. IS&T Rep. **14**(6), 1–7 (1999)
11. Marini, D., Rizzi, A.: A computational approach to color adaptation effects. Image Vis. Comput. **18**, 1005–1014 (2000)
12. Rizzi, A., Gatta, C., Marini, D.: A new algorithm for unsupervised global and local colorcorrection. Pattern Recognit. Lett. **24**, 1663–1677 (2003)
13. Land, E.H.: The Retinex theory of color vision. Sci. Am. **237**, 108–128 (1977)
14. McCann, J.J.: Retinex at 40. J. Electron. Imaging **13**(1), 6–7 (2004)
15. Frankle, J.J., McCann, J.J.: Method and apparatus for lightness imaging (1983)
16. Cooper, T.J., Baqai, F.A.: Analysis and extensions of the Frankle-Mccann Retinex algorithm. J. Electron. Imaging **13**(1), 85–92 (2004)
17. Funt, B., Ciurea, F., McCann, J.J.: Retinex in MATLAB. J. Electron. Imaging **13**(1), 48–57 (2004)
18. Zeki, S.: A Vision of the Brain. Blackwell Scientific Publications, Oxford (1993)
19. Montagna, R., Finlayson, G.D.: Constrained pseudo-Brownian motion and its application to image enhancement. J. Opt. Soc. Am. A **28**(8), 1677–1688 (2011)

20. Provenzi, E., Carli, L.D., Rizzi, A.: Mathematical definition and analysis of the Retinexalgorithm. J. Opt. Soc. Am. A **22**(12), 2613–2621 (2005)
21. Provenzi, E., Fierro, M., Rizzi, A., Carli, L.D., Gadia, D., Marini, D.: Random spray Retinex: a new Retinex implementation to investigate the local properties of the model. IEEE Trans. Image Process. **16**(1), 162–171 (2007)
22. Kolas, ø., Farup, I., Rizzi, A.: Spatio-temporal Retinex-inspired envelope with stochastic sampling: a framework for spatial color algorithms. J. Imaging Sci. Technol. **55**(4):1–10 (2011)
23. Provenzi, E., Gatta, C., Fierro, M., Rizzi, A.: A spatially variant white-patch and gray-world method for color image enhancement driven by local contrast. IEEE Trans. Pattern Anal. Mach. Intell. **30**(10), 1757–1770 (2008)
24. Land, E.H.: An alternative technique for the computation of the designator in the Retinex theory of color vision. PNAS **83**(10), 3078–3080 (1986)
25. Jobson, D.J., Rahman, Z., Woodell, G.A.: Properties and performance of a center/surround Retinex. IEEE Trans. Image Process. **6**(3), 451–462 (1997)
26. Jobson, D.J., Rahman, Z.U., Woodell, G.A.: A multiscale Retinex for bridging the gap between color images and the human observation of scenes. IEEE Trans. Image Process. **6**(7), 965–976 (1997)
27. Ramponi, G., Tenze, L., Carrato, S., Marsi, S.: Nonlinear contrast enhancement based on the Retinex approach. In: Proceeding of the Image Processing: Algorithms and Systems II, Santa Clara, CA, USA January 2003, vol. 5014, pp. 169–177 (2003)
28. Kimmel, R., Elad, M., Shaked, D., Keshet, R., Sobel, I.: A variational framework for Retinex. Int. J. Comput. Vis. **52**(1), 7–23 (2003)
29. Bertalmo, M., Cowan, J.D.: Implementing the Retinex algorithm with Wilsoncowan equations. J. Physiol.-Paris **103**(1–2), 69–72 (2009) (Neuromathematics of Vision)
30. Dorigo, M., Maniezzo, V., Colorni, A.: The ant system: optimization by a colony of cooperating agents. IEEE Trans. Syst. Man Cybern.-Part B **26**(1), 29–41 (1996)
31. Dorigo, M., Gambardella, L.: Ant colonies for the traveling salesman problem. BioSystems **43**, 73–81 (1997)
32. Kleinberg, J., Tardos, E.: Algorithm Design. Addison-Wesley Longman Publishing Co. Inc., Boston (2005)
33. Tomasi, C., Manduchi, R.: Bilateral filtering for gray and color images. In: Proceedings of the Sixth International Conference on Computer Vision, ICCV 98, Bombay, January 1998, pp. 839–846. IEEE Computer Society
34. von Kries, J.: Sources of color science. Chromatic Adaptation, pp. 109–119. MIT Press, Cambridge (1970)
35. Wang, Y.-K., Huang, W.-B.: Acceleration of the Retinex algorithm for image restoration by GPGPU/CUDA. In: Parallel Processing for Imaging Applications, number SPIE 7872, San Francisco, CA, USA January 2011
36. Sobol, R.: Improving the Retinex algorithm for rendering wide dynamic range photographs. J. Electron. Imaging **13**(1), 65–74 (2004)
37. Hartung, D.: Vascular pattern recognition: and its application in privacy-preserving biometric online-banking systems. Ph.D. thesis, Gjϕvik University College (2012)
38. Barnard, K., Cardei, V., Funt, B.: A comparison of computational color constancy algorithms - part i: methodology and experiments with synthesized data. IEEE Trans. Image Process. **11**(9), 985–996 (2002)
39. Albers, J.: Interaction of Color. Yale University Press, New Haven (1975)

Uncertainty Based Hybrid Particle Swarm Optimization Techniques and Their Applications

J. Anuradha and B.K. Tripathy

Abstract In order to handle uncertainty in data, several uncertainty based models have been introduced. Perhaps the two most notable models among these are the fuzzy set introduced by Zadeh in 1965 and rough set introduced by Pawlak in 1982. These two models address two different aspects of uncertainty and these are complementary by nature. As a result their hybridization leads to better models. Particle Swarm Optimization (PSO) is an optimization technique that performs optimized search in the solution space for optimization by updation. In this chapter, we discuss on different types of PSOs and their application in classification, feature selection and rule generation. Further, we present several hybridization of PSO with fuzzy approach, rough approach and rough fuzzy approach in developing classification algorithms. Also, we discuss on a dynamic clustering algorithm which uses a rough fuzzy hybrid model embedded with PSO. We provide as an illustration the results of application of this algorithm on several real life data sets and provide its superiority through the computation of several index values for measuring classification accuracy like DB, D, α, ρ, α^* and γ. Also, we compile some other applications of PSO.

Keywords Fuzzy set · Rough set · Classification · PSO · Rule generation · Dynamic algorithm · Accuracy measures

1 Introduction

Particle Swarm Optimization (PSO) is an evolutionary algorithm developed by Kennedy and Eberhart [1] in 1995. This model was developed based on the behavior of flock of birds flying coherently with a competition that leads to a flight with uniform speed and direction influenced by their positions, velocity and their

J. Anuradha · B.K. Tripathy (✉)
School of Computing Science and Engineering,
VIT University, Vellore 632014, Tamil Nadu, India
e-mail: tripathybk@vit.ac.in

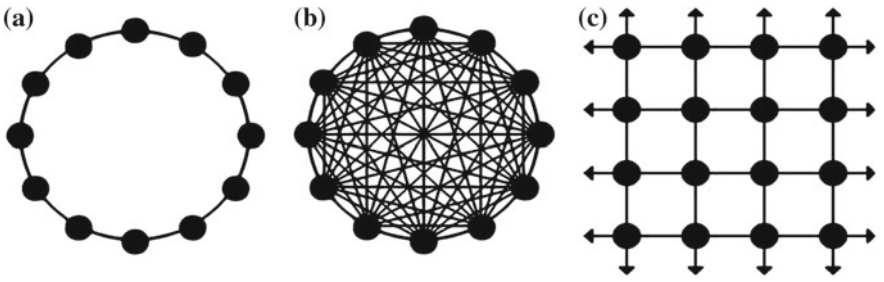
J. Anuradha
e-mail: januradha@vit.ac.in

S. Dehuri et al. (eds.), *Multi-objective Swarm Intelligence*,
Studies in Computational Intelligence 592, DOI 10.1007/978-3-662-46309-3_6

previous state. PSO like any other evolutionary algorithms is an optimization technique that performs randomized search in the solution space for an optimal solution by updating generations. PSO has less number of computations compared to Genetic Algorithm (GA), as it does not have selection procedure and off spring generation. PSO has a swarm of particles, where swarm represents a set of solutions (in solution space) and a particle represents a potential solution.

Each particle keeps track of its coordinates in the problem space which are associated with the best solution (fitness) it has achieved so far. This value is called *pbest*. Another "best" value that is tracked by the particle swarm optimizer is the best value, obtained so far by any particle in the neighbours of the particle. This location is called *lbest*. When a particle takes all the population as its topological neighbours, the best value is a global best and is called *gbest*. Different topologies of swarm exist. The most popular ones in the literature are listed below.

a. Ring (lbest) topology
b. Global (gbest) topology
c. Von-Neumann topology



The particle swarm optimization concept consists of changing the velocity of (accelerating) each particle toward its *pbest* and *lbest* locations (local version of PSO). Acceleration is weighted by a random term, The swarm is guided by the gbest and the solutions are thus accelerate towards the gbest. Then the solutions get improved in every iteration.

In past several years, PSO has been successfully applied in many research and application areas like feature selection, classification, clustering, networking, resource allocation, and VLSI design. It is demonstrated that PSO provides better results and is faster and cheaper when compared to many other methods.

2 Implementation of PSO

PSO has swarm of particles searching for the solution in the solution space. Each particle is associated with position x(t) and velocity (v(t)) (updated after every iteration) evaluated by using (1) and (2) respectively that mutually affects the other particles in the swarm. The parameters gbest and pbest determine the global best and local best solutions respectively. The pbest associated with each particle represents its best solution. The gbest is the overall best solution of the particles in the swarm

adjusted basing upon the performance of the swarm and its outcome (fitness value).

Velocity updation

$$v(t+1) = w \times v(t) + c_1 \times rand() \times (pbest(t) - x(t)) + c_2 \times rand() \times (gbest(t) - x(t)) \quad (1)$$

Position updation

$$x(t+1) = x(t) + v(t+1) \quad (2)$$

Here, w is an inertia weight, c_1 and c_2 are acceleration constants, t represent the tth iteration and rand( ) is the function that generates random values.

2.1 PSO Algorithm

Step 1: Initialize max iteration, constants c_1, c_2, swarm size and particles in swarm initialized randomly where each particle represents a candidate solution.
Step 2: For each particle p_i in the swarms do step 3.
Step 3: Calculate fitness value (fv_i) of the particle p_i.
Step 4: If fv_i is better than the best fitness value (pbest) in history then Set pbest = p_i.
Step 5: Choose the particle with the best fitness value in the swarm as the gbest.
Step 6: For each particle p_i in the swarm do steps 7 and 8.
Step 7: Calculate particle velocity v_i by (1).
Step 8: Update particle position x_i by (2).
Step 9: Repeat the steps 3–8 until the maximum number of iterations is reached or there are no more changes in the gbest value of a particle.

Goodness of each particle is evaluated by fitness function which can be any qualitative measure. The individual best of the particle is represented by pbest which gives the best local solution. The best solution among pbest in the swarm is selected as the global best solution (gbest) and the particle associated with it is the final optimal global solution.

3 Variants of PSO

PSO was developed initially for real number space (numerical data) and has been successfully applied to wider fields including optimization, Artificial Neural Network (ANN), Fuzzy systems and rough sets. In recent years research on PSO has grown in all dimensions based on the requirements of applications. There are variants of PSO like binary PSO, adaptive PSO and hybrid models of PSO. In this section we provide a brief idea about variants of PSO.

3.1 Discrete Binary PSO (DBPSO)

Kennedy and Eberhart introduced binary PSO in 1997 [2]. The particles in the swarm are represented in binary form that indicates the presence or absence of a particle in the search space. The velocity of the particle having good potential is set to 1 and is set to 0 for other particles. Here nonlinear transformation functions like sigmoid function is applied to the velocity computed using (1) that could compress the velocity to lie in the range [0, 1].

Velocity updation:

$$f(v(t+1)) = 1/(1+e^{-v(t+1)}) \tag{3}$$

Particle updation:

$$x(t+1) = \{1, if\ rand < f(v(t+1))\};\ \ 0,\ \text{Otherwise.}$$

where, $f(v(t+1))$ is the sigmoidal function that determines the velocity of the particle at t+1 state and rand is the random number.

The particles with position value 1 will be involved in the next generation of exploration. The conventional PSO algorithm has the drawback of premature convergence [3]. Some particles become inactive when it is similar to the gbest and will lose their velocity. In further generations this particle will become inactive and their role in global and local search becomes very less causing prematurity. This drawback was overcome by Immune Binary PSO (IBPSO) where the particles were given vaccination i.e. modifying the bits based on prior knowledge. Immune selection of the particle is performed to avoid earlier convergence. It works in two steps; namely immune test and probability selection, based on the antibody concentration. Immune test is based on the fitness value and can be computed by using (4)

$$D(x_i) = \frac{1}{\sum_{j=1}^{N} |f_i - f_j|}, i = 1, 2, \ldots, N \tag{4}$$

The probability selection based on antibody concentration is given in (5)

$$P(x_i) = \frac{\sum_{j=1}^{N} |f_i - f_j|}{\sum_{i=1}^{N}\sum_{j=1}^{N} |f_i - f_j|}, i = 1, 2, \ldots, N \tag{5}$$

where f_i is the fitness value of the ith particle.

Probability Based Discrete PSO (PBPSO) is a variant of Binary PSO. Here, the positions of the particle are calculated based on probability obtained by using

$$p_{ij} = (x_{ij} - X_{min})/(X_{max} - X_{min}) \tag{6}$$

PBPSO preserves both the velocity as well as position of the particles in order to update the rules of PSO algorithm. Instead of velocity (V) the particle position (X) is considered as a pseudo probability to determine the actual probability p ∈ [0, 1] for actual state of a bit (each component of the solution vector in the binary space). Probability Based Discrete PSO constantly avoids local optimal solution and reaches to the optimal solution in less number of function evaluations. Modified Discrete PSO (MDPSO) is another variant of PSO introduced in [4]. The particle position updation formula is completely reframed as given in (7) and there are no acceleration constants and inertia weights. The velocity value lies in the interval [0, 1]. α is the predefined static probability, which is usually fixed as 0.5.

$$x_i(t+1) = \begin{cases} x_i(t), \ if \ 0 \le V \le \alpha; \\ p_i(t), \ if \ \alpha \le V \le (\alpha+1)/2; \\ g_i(t), \ if \ (\alpha+1)/2 \le V \le 1. \end{cases} \tag{7}$$

3.2 Adaptive PSO

Three modifications have been made to the conventional PSO to create a new algorithm named the Adaptive PSO (APSO). The first modification is that, the whole search space is divided uniformly into several segments and can be compared to the net of a spider. Hence, each segment has a uniform distribution of solutions for each initial population distribution. In the traditional method this was random. The new method allows better search capability. The second modification is that, the information sharing mechanism of the APSO is very strong. Every swarm can take information from another swarm having better fitness value. Consequently, the swarms with the better fitness values will guide the swarms with lesser fitness value. The third modification is that, the new algorithm uses fitness value to adjust the learning factors or inertia weights c_1 and c_2. In the previous model the fitness value is never used in such regard [5, 6]. Each swarm is ranked basing on its minimum fitness value (minimizing the objective function). The swarm having the minimum value is ranked as 1.

Hence, the equation can be updated as follows,

$$(V_i = V_i * w_i + 1/rank(i) * rand() * (pbest[i] - X_i) + social\ information(i) \tag{8}$$

$$X_i = X_i + V_i \tag{9}$$

Here, rank(i) is the rank assigned to the particle based on the fitness value and social information(i) is the fitness value of the particle having the best fitness value.

3.3 Multi-objective PSO

Multi-objective PSO (MOPSO) is a special class of optimization problems that has several objectives which are competing or incommensurable, whose objective function are to be minimized simultaneously. It can be represented by minimizing $f(\overrightarrow{x})$.

$$f(\overrightarrow{x}) = \{f_1(\overrightarrow{x}), f_2(\overrightarrow{x}), f_3(\overrightarrow{x}), \ldots, f_m(\overrightarrow{x})\}$$

subject to $g_i(\overrightarrow{x}) \leq 0, i = 1, 2, \ldots, k$ and $h_i(\overrightarrow{x}) \leq 0, i = 1, 2, \ldots, p$

where $f_1(\overrightarrow{x}), f_2(\overrightarrow{x}), \ldots, f_m(\overrightarrow{x})$ are the objective functions and they conflict each other, $\overrightarrow{x}$ represent the vector of decision variables and g_i and h_i are the constraint functions of the problem. Such problems are quiet common in real life applications. The objective functions to be optimized of such problems may conflict each other such that there does not exist one solution rather a set of solutions that can compromise the trade-off between themselves. It is a kind of Pareto optimization problem having set of solutions and no other better alternative solution. They are characterized by trade-offs leading to multitude of Pareto optimal solutions [7]. The Pareto optimal solutions are determined based on the dominant and non-dominant solutions called Pareto front. Pareto front can be defined by

$$PF^* = \left\{\overrightarrow{f}(\overrightarrow{x}) \in \Re^k | \overrightarrow{x} \in P^*\right\}$$

Out of this set, the Pareto front optimal solution is identified using PSO. The Fig. 1 shows the Pareto front solution for two objective function f_1 and f_2.

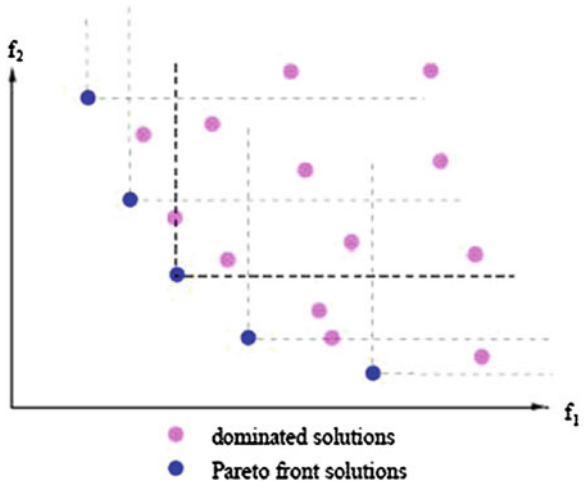


Fig. 1 Pareto front for two objective problem

In real life problems like automobile design, there exists a trade-off between performance and efficiency and similar trade off exists between stability and cost in architecture. Multi-objective PSO can be handled in two ways.

In the first method, particles explore on each objective function separately and determination of best solution is like single objective function. The main challenge in these algorithms is the manipulation of information from each objective function in order to guide the particle to the Pareto optimal solution. In the second method, all the objective functions of each of the particles are evaluated simultaneously and based on the principle of Pareto optimality. The non-dominated best position is selected as leader to guide the particles. There may be many non-dominated solutions that are equally good. The most challenging job is that, selection of one best particle as leader [8].

Multi-objective PSO procedure:

1. Initialize swarm, velocities and best positions
2. Initialize external archive (initially empty)
3. While (stopping criterion not satisfied) do
4. for each particle
   4.1 Select a member of the external archive (if needed)
   4.2 Update velocity and position
   4.3 Evaluate new position
   4.4 Update best position and external archive
5. End For
6. End While

The particles in the swarm are guided by the leader. Choosing the leader in multi-objective problem is a challenging task as it is for Pareto optimization. One way of choosing the leader in such problems is by aggregating the objective functions or any approach that minimizes each objective separately. However, most of the algorithms in MOPSO have adapted different approaches in selecting the leader [8]. Each non dominant solution is selected as the leader and out of those one promising leader is selected based on some quality measure. Some of the quality measures that indicate the closeness of the particles within the swarm are evaluated based on the neighborhood density, distance and kernel density estimation [9]. The fitness value of the particle degrades as they share the resources and it is proportionate to degree of closeness of the particle. The neighborhood is defined in terms of the radius σ_{share}. The radiuses of neighborhoods are called niches. Figure 2 represents the neighborhood defined based on the radius σ_{share} and the closeness measured based on niches.

Hu and Eberhart have proposed a dynamic neighborhood PSO to handle multi-objective problem. Lexico-graphic ordering is used which can optimize only one objective function at a time. This is a novel approach but can be used when the application has fewer objective functions. In order to avoid premature convergence, promotion of diversity during selection of leader is essential which can be attained through position updation and mutation operator. The topology of swarm determines how fast the information are transferred to neighborhood particles. Topologies with

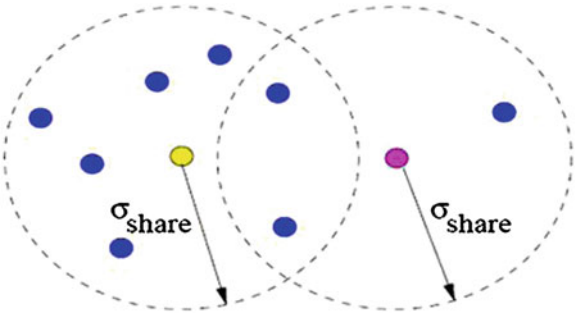


Fig. 2 Niche defined for each particle and a less crowed one is selected

smaller particles as neighborhood for each particle less than swarm, can preserve the diversity for a long time. When the velocity of the particle is almost zero then, it is unable to form generation, that means it got stuck in local optima. In order to escape from the local optima, single particle can be mutated and the mutated particle then will get an opportunity to be the global best solution which can attract other particles towards it [11]. Time variant MOPSO (TV-MOPSO) was proposed by Tripathi et al. [12] that could convert the MOPSO to adaptive based on the iteration. An archive of non-dominant solutions is maintained during exploration of particles, from which the global best solution is determined based on the diversity measure. TV-MOPSO also uses mutation operator such that it does not stuck into local optima [12].

Distributed PSO also exists, where the particles in the swarm work in the distributed environment. However, the best solution (gbest) is selected by considering performance of all particles in the distributed platform. Some of its successful applications is in networking, mobile ad hoc network etc.

4 PSO in Feature Selection

Feature selection (FS) aims at selecting relevant features from a larger set of features in a classification problem in a way that optimizes the classification performance [13]. This can be carried out by finding subsets of features in the search space, either through the filter approach or the wrapper approach as discussed in [14]. Also, to evaluate the best subset (measurement), a nonparametric measurement which calculates probability of errors for subsets is used in [15]. It also expounds a suboptimum measurement search procedure wherein the best selection is paired with the previously selected measurement.

FS has been used by many evolutionary algorithms like the Genetic Algorithms (GAs) and Particle Swarm Optimization (PSO). A comparative analysis of the algorithms on problems generated a multimodal problem generators which reveals that PSO fares much better than GAs [16], including FS. A Multi-Swarm PSO (MSPSO) is proposed in [17] which uses F-score [18] for calculating scores for each feature.

Furthermore, an Improved Feature Selection (IFS) [19] is introduced which uses MSPSO to dynamically execute Support Vector Machines (SVM) and FS. PSO has also been hybridized with SVM for better classification accuracy [20]. Here, BPSO and continuous PSO are used together for subset selection and optimization of SVM parameters. The fitness function for PSO can also be more accurately evaluated after performing FS using SVM with the one-versus-rest method as elucidated in [21]. Recently, much work has been done on a multi-objective approach for FS in classification using PSO, referred to as MOPSO [22]. This approach uses two-level Pareto front FS algorithms [7, 23] namely; non-dominated sorting PSO (NSPSO) and CMDPSO which uses the concepts of crowding, mutation and dominance.

5 PSO in Classification

Data mining is the process of discovering knowledge from large data. It unifies the research in the field of statistics, machine learning, AI and Database systems. Data classification is mostly carried out using supervised learning technique, where the class label of the instance is known previously. Set of learning examples called training set is used to map the instance to that of its class for a known set. Given an unknown instance to such a trained system will predict the class label. The classification result can be further improved using PSO technique. Hence, a hybridization of machine learning methods and PSO will produce the best result compared to conventional classification algorithms [24]. In literature, PSO is used to find the optimal positions of the class centroids, to determine the optimal weights in Neural Network etc. Improved Michigan PSO (IMPSO) classification has an adaptive inertia factor that has the flexibility to control the search space. It maintains a balance between local and global search thus improving accuracy. Further, it is extended to select the best half particle going into the next generation.

Ozbakir et al. [25] proposed a novel approach for exploring comprehensible classification rule from trained neural network with Particle Swarm Optimizer (PSO). Neural network is considered as black box, but hybridization with PSO is used to explore the best classification set. From the set of initial weights in ANN, PSO is applied to find the best weight vector for which the classification is more accurate. From the selected best classifiable instance determined by the ANN, classification rule generation is performed by finding the rule cover. The instance having higher rule cover number is added in the rule set and the cover instances are removed from the set of instances. Figure 3 shows the working principle of PSO Miner algorithm with a neat flowchart. Advantages of PSO are that it has fewer parameters, simple implementation and no overlapping or mutation calculations. This is guided by the speed, which is the velocity of the moving particle. It is considered as efficient global optimization search algorithm.

PSO algorithm here, transforms the behaviour of trained Neural Network into accurate and comprehensible classification rule. PSO with time varying inertia weight makes the PSO adaptable by changing the parameters with respect to iteration and

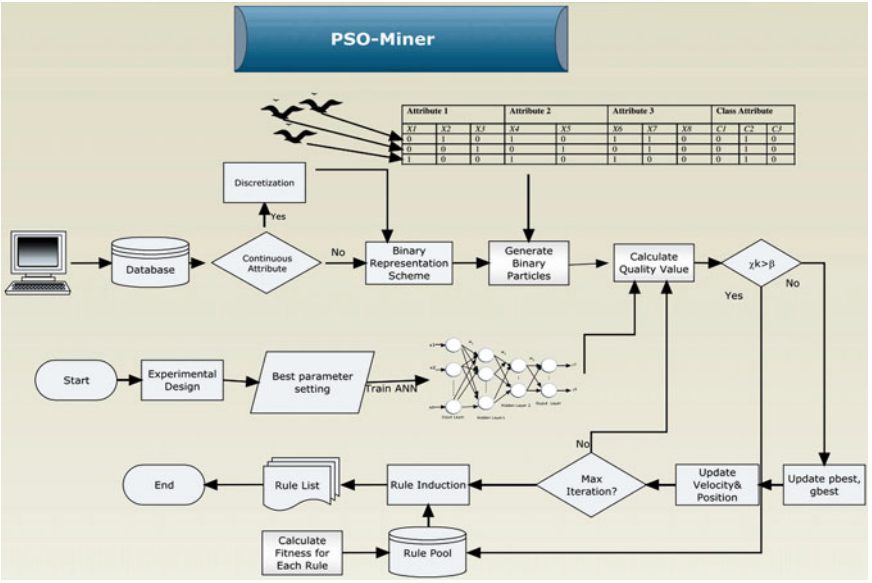


Fig. 3 Flowchart of PSO in Classification rule generation

hence can produce best value to optimize ANN output function. The following algorithm extracts the best classification rules.

5.1 Algorithm to Extract Best Classification Rule Begin

Step 1: Transform data into binary form.
Step 2: Conduct and analyze experimental design
Step 3: Train MLP with best parameter setting
Step 4: Generate weight groups
Step 5: Initialize parameters
Step 6: Set χ_k as quality function and $\beta = 0.9$
Step 7: Set fitness $h_1 = \sqrt{v \times c/2} + 4\ rand(\sqrt{v \times c/c})$
Step 8: Randomly generate population of particles
Step 9: Randomly generate velocity of particles according to this equation $v_{id}^0 = v_{min} + (v_{max} - v_{min}) \times rand[0-1]$
Step 10: Compute quality, fitness values for all particles and pbest and gbest values
Step 11: Repeat
11.1 Update the velocity and position of each particle by using following equations,

$C_1 = [(C_{1f} - C_{1i})iter/Maxiter] + C_{1i}$
$C_2 = [(C_{2f} - C_{2i})iter/Maxiter] + C_{2i}$

C_1, C_2 are the acceleration coefficients, which allow the particle to tune the cognition and social terms.

11.2 Velocity is updated using (1) and the following function is used to force the velocity value to be with in the min and max values
$v_{id}^k = h(wv_{id}^{k-1} + \triangle v_{id}^{k-1})$.

11.3 New binary values for the particle according to its comparison

$$X_{id}^k = \begin{cases} 1, \text{ if } U(0,1) < sigmoid(v_{id}^k) \\ 0, \text{ otherwise.} \end{cases}$$

11.4 Sensitivity and specificity measures are
$S_e = t_p/(t_p + f_n)$
$S_p = t_n/(t_n + f_p)$.

11.5 Compute quality values for all particles.

11.6 Update pbest and gbest based on x_k.

11.7 Keep the solutions whose x_k values are bigger than β.

Step 12: Until max iteration is satisfied.

The algorithm is to select the best population for training in NN. The following is the algorithm for Rule Induction. It is used to extract rules from the trained data and to fine tune the extracted rules.

5.2 Rule Induction Algorithm

Step 1: Begin // *to generate rule list from trained ANN data set*
Initialize the rule list as empty
Step 2: Repeat
2.1 From the solutions ($\beta > 0.9$) find the solution which has the largest fitness ($S_e * S_p$) value.
2.2 Remove the satisfied cases from the training data
2.3 Keep the solution as a rule and add it to the rule list
Step 3: Until a user specified training accuracy
Step 4: End
// extracting best rule form rule list
Step 5: Apply the rules set to the test data and find the accuracy of the rule set
Step 6: Refine and transform the solutions into linguistic rules
Step 7: Repeat
7.1 Start from the first rule in the rule set
7.2 Remove redundant attributes (whose bits in the rule consist of same value)
7.3 Use "AND" and "OR" logical operators for combining the existing values of the same attribute and different attributes, respectively

Step 8: Until the end of rule set is reached.

For extracting the rules from trained ANN by PSO miner, each attribute data set is represented in binary, depending on which the sub interval of original value is decoded. The data set with continuous input values need to be discretized before coding in binary form. After discretization, the continuous value will become categorical value which then is converted into binary. These values are fed to MLP with two hidden layers for training. PSO is applied to obtain the best population by updating the pbest and gbest values from which the rule list is formulated and which is further fine tuned.

Exploring classification rules from trained NN overcomes the pitfalls of neural network. The computation time by this method is drastically reduced by having less number of iterations and improves the accuracy of classification.

PSO algorithm is used for finding the best set of classification rules. The classification rules are generated by the conventional data mining rule generation algorithm of each particle in the swarm and the results (solutions) are optimized by exploring solution space by changing the position of the particles in the swarm. The performance of PSO algorithm is superior to GA and other data mining algorithms.

Different fitness functions used for testing were proposed by Falco et al. [26] for PSO classification is listed below. $\psi_1(i)$ is a function that computes the accuracy based on the number of instances that are correctly classified and misclassified by using (10). Here D_{Train} represents the no. of training instance and (10) represent the instances belonging to class and not belonging to the class as in [11]

$$\psi_1(i) = \frac{100}{D_{Train}} \sum_{j=1}^{D_{Train}} \delta(\overline{x}_j) \tag{10}$$

$$\delta(\overline{x}_j) = 1 \;\; if \;\; CL(\overline{x}_j) \neq \;\; CL_{known}(\overline{x}_j) \;\; | \;\; 0 \;\; \text{otherwise.} \tag{11}$$

In the second method, the function $\psi_2(i)$ finds the average distance of the training instances to the centroid, that are properly classified and its value is normalized.

$$\psi_2(i) = \frac{1}{D_{Train}} \;\; \sum_{j=1}^{D_{Train}} d\left(\overline{x}_j, \overline{p}_i^{CL_{known}(\overline{x}_j)}\right) \tag{12}$$

The third method finds the linear sum of the above two methods.

$$\psi_3(i) = \frac{1}{2}\left(\frac{\psi_1(i)}{100} + \psi_2(i)\right) \tag{13}$$

The PSO with third method produce better classification than the PSO with $\psi_1(i)$ and $\psi_2(i)$ fitness function.

6 PSO in Hybrid Computing

PSO is a simple and efficient technique that has flexibility to blend with other techniques. It is a well known fact that hybrid methods perform well compared to individual methods as these systems take advantages of both the techniques. In the following section, we propose soft computing techniques hybridized with PSO. To add to the proof of its superiority the experimental results of a proposed Rough Fuzzy PSO clustering algorithm are given.

6.1 Fuzzy PSO

Fuzzy models are capable of handling vagueness in data. Fuzzy clustering techniques are sensitive to initialization of parameters and hence they have more chances of getting stuck in local optima. PSO is an ideal global optimization tool and hence the hybrid Fuzzy PSO models achieve the benefits of handling vagueness and providing global optimal solutions.

6.1.1 Fuzzy Set

The concept of Fuzzy set was introduced by Zadeh [10] in 1965, in order to bring in the notion of graded membership of elements in a collection. Very often, we cannot categorize that an element either belongs to a collection or does not belong precisely. It may belong to the collection to a certain degree. This degree of belongingness is called its membership value. In fact, a fuzzy set A in universe U is synonymous with its membership function μ_A defined as, $\mu_A : U \rightarrow [0, 1]$, such that $\forall x \in A,\ \mu_A(x)$ is its membership value and lies in the unit interval [0, 1].

6.1.2 Fuzzy C Means (FCM)

FCM is an extension of the Hard C Means (HCM) classification and was developed by Bezdek [27]. Let $X = \{x_1, x_2, \ldots x_n\}$ be a set of n objects and $V = \{v_1, v_2, \ldots v_c\}$ be the set of c centroids of the c clusters to be formed, where $x_j \in R^m$ and $v_i \in R^m,\ j = 1, 2, \ldots, n$ and $i = 1, 2, \ldots, c$.

This algorithm can form overlapping clusters by computing the membership value μ_{ik} of an object x_k into the cluster c_i. Therefore, data points have partial belongingness to every cluster with the summation of their membership values being equal to 1. The data points (patterns) are assigned to c clusters by minimizing the objective function (14), where N is number of objects and m is a fuzzy parameter whose value is greater than 0. The centroid v_i of the cluster is updated by using (15) and the membership value is computed using (16) where d_{ik} and d_{jk} are the Euclidean distances between the object k and centroids v_i, v_j respectively.

$$\mathrm{J} = \sum_{k=1}^{N} \sum_{i=1}^{c} (\mu_{ik})^m \parallel x_k - v_i \parallel \tag{14}$$

$$v_i = \frac{\sum_{k=1}^{N} (\mu_{ik})^m x_k}{\sum_{k=1}^{N} (\mu_{ik})^m} \tag{15}$$

$$\mu_{ik} = \frac{1}{\sum_{j-1}^{c} \left(\frac{d_{ik}}{d_{jk}}\right)^{\frac{2}{m-1}}} \tag{16}$$

6.1.3 Fuzzy PSO Algorithm

The hybridization of fuzzy C Means and PSO algorithm was proposed by Mehdizadeh [28] in order to obtain the best partition.

Step 1: Select the initial parameters such as centroids, initial particle position, initial velocity of the particles, inertia weight w, constants c_1, c_2 and a small positive number ε for stopping criterion.

Step 2: Calculate the $\mu_{ik}(t)$ for all particles ($i = 1, 2, \ldots, c$ and $k = 1, 2, \ldots, n$) by (14) and update $p^{(t+1)}$.

Step 3: For each particle, calculate the fitness by (14).

Step 4: Update the global (gbest) and local best (pbest) positions.

Step 5: Update velocity $vl^{(t+1)}$ and position $xl^{(t+1)}$ by (1) and (2) respectively.

Step 6: Update $p^{(t+1)}$ by step 2.

Step 7: Update the centroid of the cluster using (15).

Step 8: Compare $p^{(t)}$ and $p^{(t+1)}$. If $|p^{(t+1)} - p^{(t)}| \leq \varepsilon$ then stop; otherwise continue from step 3.

6.2 Rough PSO

Rough set has the ability of handling uncertain and imprecise data. Rough technique is widely used for its advantage that it does not require any additional information on data [29]. It is more suitable for both quantitative and qualitative feature whose results are easy to interpret. PSO performs meta heuristic search on the solution space for the optimization of continuous functions.

6.2.1 Rough Set

The notion of Rough set was introduced by Pawlak [30] as another model to capture impreciseness in data. In rough set theory it is assumed that all the objects in the universe of discourse are associated with some information. The objects having same information are said to be indiscernible (similar). Set of objects that are indiscernible form a granule of knowledge of the universe. A rough set is represented by a pair of crisp sets called the lower and upper approximation of the set. All the objects in the

lower approximation can be classified with certainty to belong to the set whereas the objects in the upper approximation can possibly be classified to belong to the set with the available knowledge. The difference between lower and upper approximation is called the boundary region of the set and comprises of the objects in the set which cannot be classified with certainty as belonging or not belonging to the set with the available knowledge. The above concepts related to a rough set are presented in Fig. 4.

To be precise, let us take a universe U and $X \subseteq U$. Let **R** be a set of equivalence relations defined over U. We know that any equivalence relation defined over U decomposes U into disjoint equivalence classes and for any $x \in U$ we denote the equivalence class of x with respect to R by $[x]_R$ which comprises of all the elements of U those are related to x through R. For any $P \subseteq \mathbf{R}$ we denote by IND(P) the intersection of all the equivalence relations in P, which is also an equivalence relation. $K = (U, R)$ is called an information system. Let IND(K) denote $\{IND(P) | \phi \neq P \subseteq \mathbf{R}\}$. Then for any $A \in IND(K)$ we denote the lower and upper approximations of X with respect to A by $\underline{A}X$ and $\overline{A}X$ and define them as follows:

$$\underline{A}X = \{x \in U | [x]_A \subseteq X\} \text{ and } \overline{A}X = \{x \in U | [x]_A \cap X \neq \phi\}.$$

A set X is said to be A-definable if and only if $\overline{A}X = \underline{A}X$. Otherwise, X is said to be rough with respect to A. The A-boundary of X is defined as $\overline{A}X - \underline{A}X$. $\underline{A}X$, comprises of those elements of U which can be certainly classified as in X to be precise with respect to A. The elements of $\overline{A}X$ are those elements of U which possibly belong to X with respect to A. The boundary region comprises of the uncertain elements, which cannot be classified to be in X or not in X definitely with respect to A.

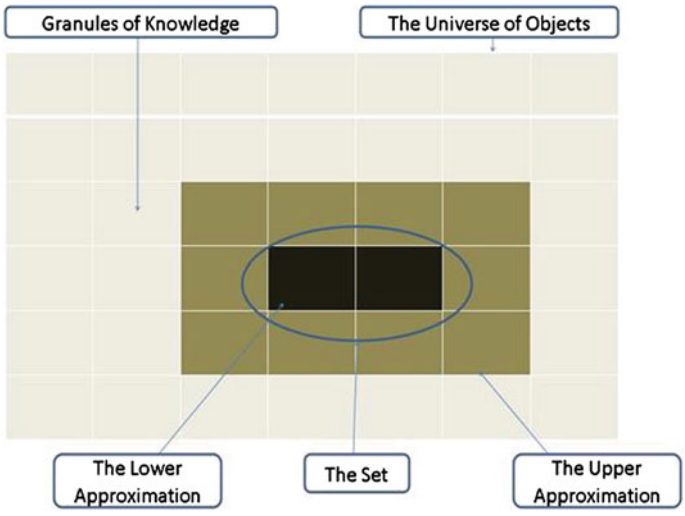


Fig. 4 Approximation space in Rough set

The Rough C means algorithm is an extension of the hard C means algorithm. It uses the following properties of rough sets [31, 32].

Property 1 *An object x_i can belonging to at most one lower approximation.*

Property 2 *If the object $x_i \in \underline{A}X$ (lower approximation) of cluster X, then it also belongs to the upper approximation of the cluster; i.e. $x_i \in \overline{A}X$.*

Property 3 *If x_i is not in any lower approximation, then it belongs to more than one boundary regions $\overline{A}X - \underline{A}X$.*

6.2.2 RCM Algorithm

Step 1: Assign the initial v_i as centroid for c clusters.

Step 2: Assign each object x_k to lower approximation $\underline{A}X_i$ or boundary region $\underline{A}X_i - \overline{A}X_i$, $\overline{A}X_i - \underline{A}X_i$ which is obtained by computing the difference between the two distances d_{ik} and d_{jk}. Where, d_{ik} and d_{jk} are the distance of object x_k from the centroid pairs v_i and v_j respectively.

Step 3: Consider the first and second minimum distance say, d_{ik} and d_{jk} respectively, if $d_{jk} - d_{ik} < \theta$ (threshold value) then $x_k \in \overline{A}X_i - \underline{A}X_i$ and $x_k \in \overline{A}X_j - \underline{A}X_j$ and cannot belong to any lower approximation. (By Property 3.) Else $x_k \in \underline{A}X_i$ since d_{ik} is minimum.

Step 4: Compute the new mean using (17)

$$v_t = \begin{cases} w_{low}\frac{\sum_{x_k \in \underline{A}X_i} x_k}{|\underline{A}X_i|} + w_{up}\frac{\sum_{x_k \in (\overline{A}X_i - \underline{A}X_i)} x_k}{|\overline{A}X_i - \underline{A}X_i|}, \\ \qquad\qquad if\ \underline{A}X_i \neq \phi \wedge \overline{A}X_i - \underline{A}X_i \neq \phi \\ \frac{\sum_{x_k \in (\overline{A}X_i - \underline{A}X_i)} x_k}{|\overline{A}X_i - \underline{A}X_i|}, \quad if\ \underline{A}X_i = \phi \wedge \overline{A}X_i - \underline{A}X_i \neq \phi \\ \frac{\sum_{x_k \in \underline{A}X_i} x_k}{|\underline{A}X_i|}, \qquad otherwise \end{cases} \tag{17}$$

Step 5: Repeat the steps 2–4 until there are no more new assignments in the clusters.

In (17) w_{low} and w_{up} represent the weights attached to the objects in the lower approximation and the boundary region during the clustering process such that their sum is equal to 1. The parameter w_{low} is taken to have values lying in (0.5, 1). The clustering process is influenced by these parameters as they represent the relative importance to the lower approximations and the boundary regions. RCM algorithm forms a pair of lower and boundary region by steps 2 and 3 for each cluster and the objects in boundary region are intersecting by Property 3. The objects in the lower approximation are definitely classifiable as belonging to a cluster whereas objects in boundary region are uncertain. The partition formed by this method can be further

optimized by embedding PSO technique [20] to the clustering procedure and it is given below.

6.2.3 Rough PSO Algorithm

Step 1: Select the initial parameters such as centroids, initial particle position, initial velocity of the particles, inertia weight w, c_1, c_2 and small positive number ε for stopping criterion

Step 2: initialize the threshold value θ

Step 3: Assign each object x_k to lower approximation $\underline{A}X_i$ or boundary region $\underline{A}X_i - \underline{A}X_i$, $\overline{A}X_j$, $\overline{A}X_j$ which is obtained by computing the difference between the two distances d_{ik} and d_{jk}. Where, d_{ik} and d_{jk} are the distance of object x_k from the centroid pairs v_i and v_j respectively.

Step 4: Consider the first and second minimum distance say, d_{ik} and d_{jk} respectively, if $d_{jk} - d_{ik} < \theta$ (threshold value)

then $x_k \in AX_i - \underline{A}X_i$ and $x_k \in \overline{A}X_j - \underline{A}X_j$ and cannot belong to any lower approximation. (by Property 3)

else $x_k \in \underline{A}X_i$ since d_{ik} is minimum.

Step 5: calculate the evaluation function using DB index and D index (given in Sect. 6.3.2) of each particle.

Step 6: Update the pbest and gbest values

Step 7: Update the velocity and position of the particle using (1) and (2)

Step 8: Calculate the new centroid of the cluster using (17).

Step 9: Repeat steps 3–8 until $|p^{t+1} - p^t| \leq \varepsilon$.

Rough PSO has been successful in many applications including feature selection [20] where rough set is used for generating reduct and PSO is applied to get the best reduct, image classification [33], web catching, association rules and the challenging task of numerical rule mining using Rough PSO [34].

6.3 Rough Fuzzy PSO

The proposed algorithm uses rough fuzzy c means for clustering and PSO for finding the best prototype that gives better partition. The particles in swarm are the cluster prototypes that partition the patterns. Here, rough fuzzy C-means (RFCM) algorithm is used for generating clusters. This model constructs the lower approximation $\underline{A}X_i$ and the boundary region $(\overline{A}X_i - \underline{A}X_i)$ of each cluster based on the membership value μ_{ik} of the objects x_k with respect to cluster c_i. The objects in lower approximation can be obtained by considering the difference between the first and second maximum membership of an object x_k to clusters c_i and c_j respectively i.e. $\mu_{ik} - \mu_{jk}$. If their difference is greater than or equal to the threshold value θ, then the object x_k belongs to lower approximation of cluster c_i and may not be the part of other clusters. If

their difference is less than θ, then the object x_k will be put into boundary region of clusters c_i and c_j and cannot belong to lower approximation of any cluster. The centroid value is updated basing upon the membership values of the objects in lower approximation and boundary region using (18). The cluster partitioning of the particle is evaluated by DB and D indices which gives the fitness value of the particle. The fitness value of the particle is compared against its history to identify its best solution and it is pbest the local best solution. The particle with the best fitness value in swarm is gbest the global best solution. The velocity of each particle is updated using (1) and the position of the particle is updated using (2). The above procedure is iterated until convergence i.e. there is no more change in the particle position and their gbest value. The partitioned obtained by this procedure are static forming fixed number of clusters.

$$v_t = \begin{cases} w_{low}\frac{\sum_{x_k \in \underline{A}X_i} x_k}{|\underline{A}X_i|} + w_{up}\frac{\sum_{x_k \in (\overline{A}X_i - \underline{A}X_i)} x_k}{|\overline{A}X_i - \underline{A}X_i|}, \\ \qquad if\, \underline{A}X_i \neq \phi \wedge \overline{A}X_i - \underline{A}X_i \neq \phi \\ \frac{\sum_{x_k \in (\overline{A}X_i - \underline{A}X_i)} x_k}{|\overline{A}X_i - \underline{A}X_i|}, \quad if\, \underline{A}X_i \neq \phi \wedge \overline{A}X_i - \underline{A}X_i \neq \phi \\ \frac{\sum_{x_k \in \underline{A}X_i} x_k}{|\underline{A}X_i|}, \qquad otherwise \end{cases} \tag{18}$$

6.3.1 RFCM—PSO Dynamic Clustering Algorithm

Step 1: Initialize the parameter P (population size), c_1, c_2 (constants), w (inertia weight) and maximum number of iteration.

Step 2: Create a swarm with P particles and initialize X (particle position), pbest (for each particle), and velocity V (for each particle) and gbest for the swarm.

Step 3: Assign initial means v_i for initial number of the c clusters.

Step 4: Compute μ_{ik} by (16) for c clusters and N data objects.

Step 5: Assign each data object (patterns) x_k to the lower approximation $\underline{A}X_i$ or boundary region $\underline{A}X_i - AX_i$, $\overline{A}X_j - \underline{A}X_j$ of cluster pairs X_i and X_j by computing the difference in its membership $\mu_{ik} - \mu_{jk}$ to cluster centroid pairs v_i and v_j.

Step 6: Let μ_{ik} be first maximum and μ_{jk} be the next maximum. If $\mu_{ik} - \mu_{jk}$ is less than θ (threshold value), then $x_k \in \overline{A}X_i - \underline{A}X_i$ and $x_k \in \overline{A}X_j - \underline{A}X_j$ and x_k cannot be a member of lower approximation of any cluster.

else

$x_k \in \underline{A}X_i$ such that membership μ_{ik} is maximum over the c clusters.

Step 7: Compute the new mean for each cluster X_i, applying (18)

Step 8: Compute the fitness value fv_i for each particle using DB and D ratio by (21) and (22) respectively.

Step 9: Calculate the pbest for each particle if ($pbest < fv_i$) then set $pbest = p_i$

Step 10: Calculate the gbest of the swarm if ($gbest < fv_i$) then set gbest to fv_i and its corresponding particle i.

Step 11: Update the velocity matrix for each particle by (1)

Step 12: Update the position matrix for each particle by (2)

Step 13: Perform union of all upper approximation $\bigcup_{i=1}^{c}(\overline{A}X_i - \underline{A}X_i)$.

Step 14: Find the object o_k having maximum number of objects close to it (using Euclidean distance) and it is far away from the centroid v_i (greater than the threshold) of c clusters

Step 15: Add o_k to centroids $\{v\}$ and increment the c (number of clusters).

Step 16: Repeat steps 2–15 until convergence i.e., there are no more new assignments in the clusters or no change in gbest value.

As the static partitioning forms the predetermined number of clusters, it puts restriction in forming pure clusters. In order to form dynamic clusters (new clusters at run time), the objects in the upper approximation alone are considered, for choosing a new centroid. New centroids are selected based on the pattern that is dissimilar from the existing patterns i.e. centroids $\{v\}$. It is obtained by selecting an object from the upper approximation sets having maximum closest pair of objects and that is far away (distance > 0.8) from the existing clusters. The closeness is measured based on the distance and whose value is less than 0.4 (by trial). The above procedure is repeated until there are no more new assignments to the clusters i.e. there is no change in the membership values of the objects in the clusters.

6.3.2 Cluster Validity Index

Cluster validity indices are used to measure the performance of a cluster using inter and intra cluster distances. In this paper we use the Davies Bouldin (DB) index and Dunn (D) index given in [17] for validating the formation of the clusters. DB is a ratio of average dispersion of objects within cluster to separation between clusters. Cluster separation is measured by $D(X_k, X_l)$ given in (22) where X_k and X_l are the kth and the lth clusters respectively. Scatter of objects within cluster is measured by $S_{rf}(x_k)$ given in (19). D index is used to measure the clusters that are compact and separated. The expressions for the DB and the D index for rough Fuzzy clustering is given in (21) and (22) (Mitra et al. [32]).

The average distance between objects within the cluster X_k is given below.

$$S_{rf}(U_i) = \begin{cases} \omega_{low}\dfrac{\sum\limits_{x_k\in\underline{A}U_i} u_{ik}^m \|x_k - v_i\|^2}{\sum\limits_{x_k\in\underline{A}U_i} u_{ik}^m} + \omega_{up}\dfrac{\sum\limits_{x_k\in(\overline{A}U_i-\underline{A}U_i)} u_{ik}^m \|x_k - v_i\|^2}{\sum\limits_{x_k\in(\overline{A}U_i-\underline{A}U_i)} \mu_{ik}^m}, \\ \qquad\qquad if\ \underline{A}U_i \neq \phi \wedge \overline{A}U_i - \underline{A}U_i \neq \phi \\ \dfrac{\sum\limits_{x_k\in(\overline{A}U_i-\underline{A}U_i)} \mu_{ik}^m \|x_k - v_i\|^2}{\sum\limits_{x_k\in(\overline{A}U_i-\underline{A}U_i)} \mu_{ik}^m} \quad if\ \underline{A}U_i \neq \phi \wedge \overline{A}U_i - \underline{A}U_i \neq \phi \\ \dfrac{\sum\limits_{x_k\in\underline{A}U_i} u_{ik}^m \|x_k - v_i\|^2}{\sum\limits_{x_k\in\underline{A}U_i} \mu_{ik}^m}, \qquad otherwise \end{cases} \tag{19}$$

Here, u_{ik}^{m} represent the membership of the object k in ith cluster and the value of m is 1. Between clusters separation is computed by the following equation.

$$d(X_k, X_l) = \frac{\sum_{i,j} \parallel x_i - x_j \parallel}{|c_k \parallel c_l|} \tag{20}$$

Here $x_1, x_2, \ldots, x_{ck}$ is the set of objects in a cluster. Here, x_i and x_j represent the cluster objects and c_k and c_l represent the kth and lth clusters respectively.
The DB index minimizes intra cluster distance and maximizes inter cluster distance and is given by

$$DB_{rf} = \frac{1}{c}\sum_{i=1}^{c} \underset{j\neq i}{max}\left\{\frac{S_{rf}(X_i) + S_{rf}(X_j)}{d(X_i, X_j)}\right\} \tag{21}$$

The compactness and separation of clusters are measured by

$$D_{rf} = \min_{j}\left\{\underset{j\neq i}{min}\left\{\frac{d(X_i, X_j)}{\underset{k}{max}\, S_{rf}(X_k)}\right\}\right\} \tag{22}$$

Here, i, j and k represent the clusters. The denominator of DB_{rf} is similar to numerator of D_{rf}. An optimal partition is obtained by maximizing the inter cluster separation while minimizing the intra cluster distance.

6.3.3 Quantitative Analysis

Various qualitative measures are used to evaluate the performance of the rough fuzzy algorithm [35]. We present below the definitions of these indices. Before presenting these indices we introduce some notations.

Let $\underline{A}(X_i)$ and $\overline{A}(X_i)$ be the lower and upper approximations of cluster X_i, and $B(X_i) = \overline{A}(X_i) - \underline{A}(X_i)$ denote the boundary region of cluster X_i . Let μ_{ij} represent the membership of the object x_j in cluster X_i and $1 < m_1 < \infty$ is the fuzzifier. The parameters ω and $\overline{\omega}$ and correspond to the relative importance of lower and boundary regions respectively. Also, $\overline{\omega} = 1 - \omega$.

Definition 1 (α *index*) It is given by the expression

$$\alpha = \frac{1}{c}\sum_{i-1}^{c} \frac{\omega A_i}{\omega A_i + \varpi B_i} \tag{23}$$

$$A_i = \sum_{x_i \in \underline{A}(X_i)} (v_{ij})^{m_1} = |\underline{A}(X_i)| \text{ and } B_i = \sum_{x_j \in (X_i)} (v_{ij})^{m_1} \tag{24}$$

The α index represents the average accuracy of c number of clusters. It is the average of the ratio of the number of objects in lower approximation to that in upper

approximation of each cluster. In effect, it captures the average degree of completeness of knowledge about all clusters. A good clustering procedure should make all objects as similar to their centroids as possible. The index increases with increase in similarity within a cluster. Therefore, for a given data set and c value, the higher the similarity value of objects within the clusters gives a higher index value. The value of α increases with the value of c. In an extreme case when the number of clusters is maximum, i.e., c = n, the total number of objects in the data set, the value of $\alpha = 1$, when $\overline{A}(X_i) = \underline{A}(X_i), \forall i$, that is, all the clusters X_i are exact or definable. Whereas if $\overline{A}(X_i) = \underline{A}(X_i), \forall i$, the value of $\alpha = 0$. Thus, $0 \leq \alpha \leq 1$.

Definition 2 (ρ *index*) It represents the average roughness of c number of clusters and is obtained by subtracting the average accuracy α from 1.

$$\rho = 1 - \alpha = 1 - \frac{1}{c} \sum_{i=1}^{c} \frac{\omega A_i}{\omega A_i + \varpi B_i} \tag{25}$$

Definition 3 (α^* *index*) It represents the accuracy of approximation

$$\alpha^* = \frac{\sum_{i=1}^{c} \omega A_i}{\sum_{i=1}^{c} \{\omega \mathrm{A}_i + \overline{\omega} B_i\}} \tag{26}$$

Definition 4 (γ *index*) It represents the approximation of the clustering algorithm obtained by finding the ratio of the total number of objects in lower approximation to the of cardinality of objects in the universe.

$$\gamma = \frac{\sum_{i=1}^{C} |\underline{A}(X_i)|}{|U|} \tag{27}$$

Clustering is formed based on Rough C Means and their results are accelerated using PSO algorithm by finding the global best particle (best solution). The experimental results with validity index (DB and D values) and their ratio across various real life data sets are tabulated in the following section.

7 Implementation and Results

The proposed dynamic RFPSO algorithm is applied to various data sets available in UCI repository and to ADHD data set for diagnosis of ADHD [36]. ADHD is about the children's behavior in class, measured by the parameters given in DSM—IV test. Under the guidance of pediatrician Dr. K. V. Arulalan, the data of nearly 200 students are collected at various schools in Vellore district, Tamil Nadu, India. The proposed algorithm outperforms the other algorithms of this type. The performance

Table 1 Experimental result of various algorithms on real life data sets

Data set	Index	Dynamic RFCM	V (Dynamic) RFCM)	Static RFCM	V (Static RFCM)	Rough PSO	V (Rough PSO)	Dynamic RFPSO	V (Dynamic REPSO)
Libra movement	DB	0.5695	0.4846	0.67	0.6713	1.3239	0.1903	0.3409	0.1841
	D	1.17531		0.998		6.9595		1.8515	
CMC	DB	0.376255	0.1071	1.03	0.8084	1.8705	3.409	0.4009	0.1473
	D	3.51259		1.27411		0.5487		2.7225	
Iris	DB	0.078	0.004	0.0782	0.0039	0.085	0.0023	0.0896	0.0052
	D	19.8458		19.8458		36.2263		17.1943	
Wine	DB	0.49	0.1444	0.49012	0.1444	0.5857	0.2375	0.3779	0.0901
	D	3.3932		3.3932		2.4659		4.1928	
Glass	DB	0.3743	0.5923	0.3743	0.5923	0.417	0.455	0.1894	0.0493
	D	0.6319		0.6319		0.9165		3.8398	
Cancer	DB	0.4919	0.2657	0.5819	0.2735	1.8947	2.5549	0.4829	0.2225
	D	1.85143		2.1276		0.7416		2.1694	
ADHD	DB	1.633	1.5687	1.633	1.3029	4.5166	10.9414	0.3534	0.066
	D	1.041		1.041		0.4128		5.3461	

Table 2 Qualitative measures of RFPSO on various data sets

Data set	α	ρ	α^*	γ
Libra movement	0.9987	0.0013	0.999	0.9031
CMC	0.994	0.006	0.9943	0.7253
Iris	0.9828	0.017	0.988053	0.8970
Wine	0.974	0.02596	0.97566	0.8832
Glass	0.99613	0.00387	0.9964	0.7588
Cancer	0.9954	0.004	0.9958	0.8766
ADHD	0.9926	0.0073	0.9927	0.8128

of the algorithm was measured by DB and D index as given in (21) and (22). For a cluster with high purity D index should be higher and DB index should be lower. In order to have a quick comparison of their performance, ratio between DB and D is calculated.

$$V = \frac{DB}{D} \tag{28}$$

The ratio V should be low for a pure cluster as D is maximized and DB is minimized (explained in Sect. 6.3.2). In other words, higher the v value indicates impure cluster. Here, the proposed algorithm is tested with various real life data sets in UCI repository and their details are summarized in Table 1.

From the above Table 1 it is apparent that RFPSO performance is better than other algorithm followed by dynamic RFCM. In order to show the improvement in the results of RFPSO, further the partitions formed by both algorithms and their efficiency are measured through average accuracy of the cluster α, average roughness of the cluster ρ, accuracy of approximation (α^*), quality of approximation γ. The qualitative measures of RFPSO are listed down in Table 2.

The comparison between various algorithms like static RFCM, dynamic RFCM, RPSO and RFPSO on real life data set is shown graphically in Fig. 5. It is more apparent that performance of RFPSO is superior. The results of dynamic RFCM are also consistent next to RFPSO. The DB and D ratio is low and its performance is better than other algorithms and is indicated by blue line that is closer to 0. It is followed by dynamic RFCM, static RFCM and Rough PSO indicated by rose, dark blue and yellow lines respectively. Rough PSO, though has the result is closer to 0 on many data set, still has extreme results on higher dimensional data sets and the results are fluctuating.

The qualitative measures DB and D index of RFPSO on various parameter values are recorded and represented as graph in Figs. 6 and 7. The graph was plotted across different w_{low}/w_{up} values as x-axis with different values (parameter used to assign objects in lower or boundary region) values as z axis and the range of D values in y-axis for various data sets mentioned in Table 1. The vertical lines in the graph represent various data sets on each of parameters w_{low}/w_{up} and θ. From Fig. 6, one can understand that the performance of the proposed RFPSO algorithm is good with

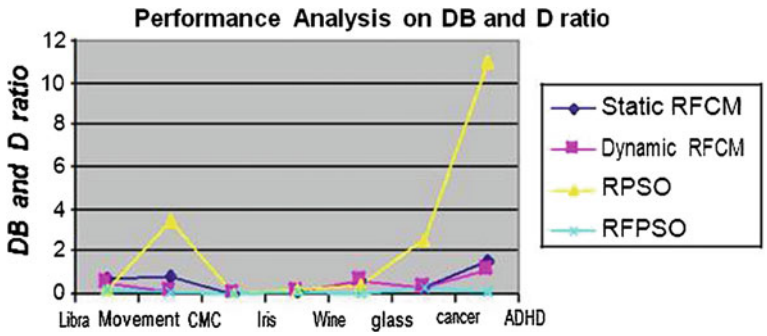


Fig. 5 Performance analyses on various algorithms

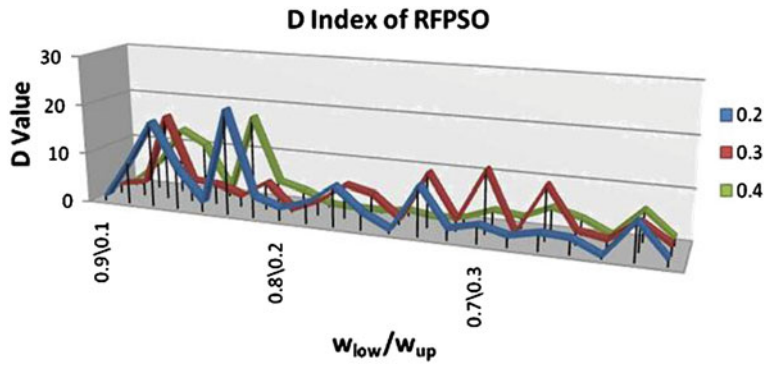


Fig. 6 D index measure on various parameter of RFPSO

parameters $w_{low} = 0.9$, $w_{up} = 0.1$ and $\theta = 0.3, \theta = 0.2$. The same parameter values holds good for DB index also shown in Fig. 7.

From the above experimental analysis, RFPSO and dynamic RFCM can be applied to a larger data set for high quality decision making irrespective of dimensions of data sets.

8 Applications of PSO

Application of PSO is exhaustive and are in various fields like data mining, networking, job scheduling and pattern recognition.

8.1: PSO is used for optimal design of hybrid renewable energy systems (HRES). It is applied to minimize the total cost of the system, unmet load, and fuel emission.

8.2: The PSO-based optimization approach has allowed the design of a micro sphere-based amplifying system more efficient than a similar device. In fact, the

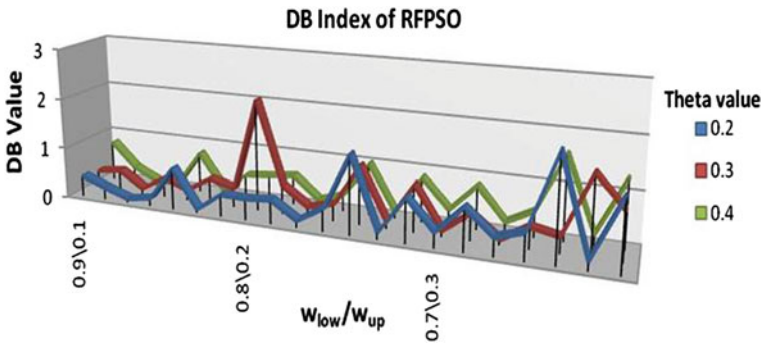


Fig. 7 DB index measure on various parameter of RFPSO

amplifier designed via the PSO exhibits a simulated gain, which is higher than the gain of the amplifier designed via the other method.

8.3: Finding association rule on numerical data is challenging. PSO is used for mining association rule on data without prior discretization and with multiple objectives.

8.4: Finding optimal weight in neural network which will enhance the classification rate.

8.5: It has been applied to dynamically changing applications in cloud and high performance computing for rapid and efficient service at any instance of time.

8.6: The Hierarchical Particle Swarm Optimization with receives the request in cloud from various resources, employs multiple swarm interaction and implements cyclic and orthogonal properties in a hierarchical manner to provide the near optimal solution.

8.7: Electricity is big a demand in countries like India. Yearly electricity consumption trends of most developing countries usually exhibit approximately exponential growth curves. PSO is applied to improve forecasting precision of electricity consumption, based on which resource planning can be done.

8.8: Its application is successful in job scheduling, feature selection, prediction of protein sequence, molecule structures, survival of ICU patients, load forecasting, object detection, fault detection etc.

8.9: It is successful in chemical industry for optimization of chemical reaction. One such application is to optimize the process variables for enhanced lipopeptide production by marine Bacillus megaterium, using food waste.

8.10: The determination of type of generation technology suitable for an autonomous power system calls for comprehensive planning is necessary.

8.11: To determine optimal mix of resources, PSO performs optimal component sizing for each of the configuration. Optimal power flow and reactive compensation in power system planning.

9 Conclusion

PSO is a successful optimization algorithm that has shown its superiority over the conventional algorithms. It is a competitive technique to other optimization algorithms like GA, ACO and Bee Colony Optimization. The popularity of PSO is due to its simple and less number of computations as it does not have mutation and crossover. Hence it is effortless procedure for hybridization to other techniques. It has been widely applied to scientific and engineering fields for varied applications. Dynamic Rough Fuzzy PSO model proposed shows an evident of superiority of PSO and its flexibility in hybridization. In this chapter we focused on the different aspects of PSO and presented several general and hybrid classification algorithms in an attempt to establish the superiority of PSO based algorithms over the general ones where PSO is not used. Several accuracy measure indices and real life data sets have been used for the experimental analysis leading to this conclusion.

References

1. Kennedy, J., Eberhart, R.C.: Particle swarm optimization. In: Proceedings of the 1995 IEEE International Conference on Neural Networks, vol. 4, pp. 1942–1948. Perth (1995)
2. Kennedy, J., Eberhart, R.C.: A discrete binary version of the particle swarm algorithm, systems, man and cybernetics. In: Proceedings of the IEEE International Conference on Computational Cybernetics and Simulation, 12–15 October 1997, vol. 5, pp. 4104–4108 (1997)
3. Lin Li, X.: A particle swarm optimization and immune theory based algorithm for structure learning of Bayesian networks. Int. J. Database Theory Appl. **3**(2), 61–69 (2010)
4. Menhas, M.I., Wang, L., Fei, M., Pan, H.: Comparative performance analysis of various binary coded PSO algorithms in multivariable PID controller design. Expert Syst. Appl. **39**, 4390–4401 (2013)
5. Hossen, S.Md., Rabbi, F., Rahman, M.Md: Adaptive particle swarm optimization (APSO) for multimodal function optimization. Int. J. Eng. Technol. **1**(3), 98–103 (2009)
6. Xiao-Feng, X., Wen-Jun, Z., Zhi-Lian, Y..: Adaptive particle swarm optimization on individual level. In: International Conference on Signal Processing (ICSP), pp. 1215–1218. Beijing (2002)
7. Anuradha, J., Tisha, V.R., Tripathy B.K., Arulalan.: Diagnosis of ADHD using SVM algorithm. ACM Compute (2010)
8. Parsopoulos, K.E., Vrahatis, M.N.: Multiobjective Particle Swarm Optimization Approaches. Chapter II, pp. 20–42. IGI Global Publishers (2008)
9. Reyes-Sierra, M., Coello Coello, C.A.: Multi-objective particle swarm optimizers: a survey of the state-of-the-art. In: Technical Report EVOCINV-01-2006, Evolutionary Computation Group at CINVESTAV, Mexico, March 2006
10. Zadeh, L.A.: Fuzzy sets. Inf. control **8**, 338–353 (1965)
11. Esquivel, S.C., Coello, C.A.C.: On the use of particle swarm optimization with multimodal functions. In: The 2003 Congress on Evolutionary Computation (CEC'03), vol. 2, pp. 1130–1136 (2003)
12. Tripathi, P.K., Bandyopadhyay, S., Pal, S.K.: Multi-objective particle swarm optimization with time variant inertia and acceleration coefficients. Inf. Sci. **177**, 5033–5049 (2007)
13. Dash, M., Liu, H.: Feature selection for classification. Intell. Data Anal. **1**(4), 131–156 (1997)
14. Kohavi, R., John, G.H.: Wrappers for feature sub set selection. Artif. Intell. **97**, 273–324 (1997)
15. Whitney, A.: A direct method of nonparametric measurement selection. IEEE Trans. Comput. **C-20**(9), 1100–1103 (1971)
16. Kennedy, J., Spears, W.: Matching algorithms to problems: an experimental test of the particle swarm and some genetic algorithms on the multimodal problem generator. In: IEEE Congress on Evolutionary Computation (CEC'98), pp. 78–83. Anchorage (1998)

17. Liu, Y., Wang, G., Chen, H., Dong, H.: An improved particle swarm optimization for feature selection. J. Bionic Eng. **8**(2), 191–200 (2011)
18. Chen, Y.W., Lin, C.J.: Combination of feature selection approaches with SVM in credit scoring. Expert Syst. Appl. (37), 315–324 (2006)
19. Chuang, L.Y., Chang, H.W., Tu, C.J., Yang, C.H.: Improved binary PSO for feature selection using gene expression data. Comput. Biol. Chem. **32**, 29–38 (2008)
20. Huang, C.L., Dun, J.F.: A distributed PSO-SVM hybrid system with feature selection and parameter optimization. Appl. Soft Comput. **8**(4), 1381–1391 (2008)
21. Tu, C.J., Chuang, L.Y., Chang, J.Y., Yang, C.H.: Feature selection using PSO-SVM, IAENG Int. J. Comput. Sci. **33**(1), No. 18. IJCS (2007)
22. Xue, B., Zhang, M., Browne, W.N.: Particle swarm optimization for feature selection in classification: a multi-objective approach. IEEE Trans. Cybern. **43**(6), 1656–1671 (2013)
23. Knowles, J.D., Corne, D.W.: Approximating the non dominated front using the pareto archived evolution strategy. Mass. Inst. Technol. Evoluti. Comput. **8**(2), 149–172 (2000)
24. Sousa, T., Silva, A., Neves, A.: Particle swarm based data mining algorithms for classification tasks. Parallel Comput. **30**, 767–783 (2004)
25. Ozbakir, L., Delice, Y.: Exploring comprehensible classification rules from trained neural networks integrated with a time-varying binary particle swarm optimizer. Eng. Appl. Artif. Intell. (2010). doi:10.1016/j.engappai.2010.11.008
26. De Falco, I., Della Cioppa, A., Tarantino, E.: Facing classification problems with particle swarm optimization. Appl. Soft Comput. **7**(3), 652–658 (2007)
27. Bezdek, J.C.: Pattern Recognition with Fuzzy Objective Function Algorithm. Plenum, New York (1981)
28. Mehdizadeh, E.: A fuzzy clustering using PSO algorithm. Int. J. Manag. Sci. Eng. Manag. **4**(4), 311–320 (2009)
29. Pawlak, Z.: Rough Sets: Theoretical Aspects of Reasoning about Data. Kluwer Academic Publishers, The Netherlands (1991)
30. Pawlak, Z.: Rough set. Int. J. Inf. Comput. Sci. 341–356 (1982)
31. Mitra, S., Banka, H., Pedrycz, W.: Rough—fuzzy collaborative clustering. IEEE Trans. Syst., Man Cybern.—Part B: Cybern. **36**(4), 795–805 (2006)
32. Mitra, S., Pedrycz, W., Barman, B.: Shadowed C means: integrating fuzzy and rough clustering. Pattern Recognit. **43**, 1282–1291 (2010)
33. Hung, C.C., Hendri, P., Chen, K., Letkeman.: Multispectal image classification using rough set theory and particle swarm optimization. In: Advances in Geosciences and Remote Sensing—2009), pp. 569–595 (2009)
34. Abido, M.A.: Two-level of non-dominated solutions approach to multi objective particle swarm optimization. ACM 978-1-59593-697-4/07/0007 (2007)
35. Maji, P., Pal, S.K.: RFCM: a hybrid clustering algorithm using rough and fuzzy sets. Fundamenta Informaticae **80**, 475–496 (2007)
36. Alatas, B., Akin, E.: Rough particle swarm optimization and its applications in data mining. Soft Comput. (12), 1205–1218 (2008)

Hybridization of Evolutionary and Swarm Intelligence Techniques for Job Scheduling Problem

R. Uma Rani

Abstract **Scheduling** is the process of deciding how to commit resources to varieties of possible jobs. It also involves the allocation of resources over a period of time to perform a collection of jobs. **Job Scheduling** is the sequencing of the different operations of a set of jobs in different time intervals to a set of machines. In this chapter a hybrid technique combing an evolutionary and swarm intelligence technique to the job scheduling problem is proposed. In literature, different hybrid techniques are used for performing job scheduling process in various fields. The existing systems have utilized techniques such as **Artificial Bee Colony (ABC)**, **Genetic Algorithm (GA)**, **Particle Swarm Optimization (PSO)**, **Ant Colony Optimization (ACO)**, **Simulated Annealing (SA)** etc. and **hybrid techniques** were derived from the combination of these algorithms. These hybrid techniques mostly concentrate on two factors such **as the minimization of the makespan and completion time**. The performance of these hybrid techniques lack in the scheduling process because, they have not considered the efficient factors like **(i) turnaround time, (ii) throughput and (iii) response time** during the job scheduling process. The main aim of this work is to provide a better hybrid job scheduling technique by overcoming the problems that exist in the literary works and to minimize the completion and makespan time. The proposed technique considered the efficient factors **(i)** turnaround time **(ii)** throughput and **(iii)** response time which were left out in the existing hybrid techniques for job scheduling process. The performance of the proposed hybrid job scheduling technique was analyzed with the existing hybrid techniques. The experimental results proved that the proposed job scheduling technique attained high accuracy and efficiency than the existing hybrid techniques.

Keywords Scheduling · Optimization · Job scheduling · Optimization algorithms

R. Uma Rani (✉)
Associate Professor of Computer Science, Sri Sarada College for Women (Autonomous), Salem, Tamil Nadu, India
e-mail: umainweb@gmail.com

S. Dehuri et al. (eds.), *Multi-objective Swarm Intelligence*,
Studies in Computational Intelligence 592, DOI 10.1007/978-3-662-46309-3_7

1 Introduction

Scheduling is the process of deciding how to commit resources between varieties of possible jobs. It involves the allocation of resources over a period of time to perform a collection of jobs. It has been the subject of a significant amount of literature in the operation research field. It is a decision-making process that plays an important role in most manufacturing and service industries [1]. It can also be described as a decision-making process with the goal of optimizing the objectives such as completion and makespan reduction efficiency.

Job Scheduling Problem (JSP) is a finite set of jobs processed on a finite set of machines. Job Scheduling is the sequencing of the different operations of a set of jobs in the different time intervals of a set of machines. Each job follows a predefined machine order and has a deterministic processing time. Each machine can process at most one job at a time, which cannot be interrupted until its completion. The objective of the JSP is to find a schedule to minimize the time required to complete all jobs; that is, to minimize the makespan time [2]. The main objective of scheduling is to arrive at a position where minimum processing time is achieved. In addition, Job scheduling also aids in determining an optimal resource for improving the overall system performance, as well as to exploit the available resources more competently. The job scheduling problem (JSP) is the most challenging one which has been the subject of many research studies during the recent decades. A schedule is an assignment of operations to time slots on a machine. The objective is to find the sequence of all the jobs in such a way that an established criterion is optimized. These types of problems are found in areas like flexible manufacturing systems, production planning, computer design, logistics and others.

The Scheduling of the job shop provides a set of resources to tasks over time. In the past years vast amount of research have been done on operational research. This mainly focuses on finding ways of giving jobs to machines such that it meets certain criteria and an objective function is optimized. Till now a 100 % perfect method has not been found to get the optimal solution in all types of job shop scheduling.

The constraints for a job shop scheduling problems are:

- A job must not visit the same machine more than once.
- There are no precedence constraints on operations of different jobs.
- Operations can't be interrupted.
- Each machine can process only one job at a time.
- Each job should go through a particular sequence of operations as predefined.

Each optimization problem must have an objective function which has to be either minimized or maximized in order to get a solution. In this case the objective function is the makespan value or the length of the schedule.

2 Evolutionary Algorithms

Evolutionary Algorithms (EAs) are stochastic search algorithms that are inspired by the metaphor of natural Darwinian Theory of biological evolution. In the computer science field of artificial intelligence, a genetic algorithm (GA) is a search heuristic that mimics the process of natural evolution. This heuristic is routinely used to generate useful solutions to optimization and search problems [3]. Genetic algorithms belong to the larger class of evolutionary algorithms (EA), which generate solutions to optimization problems.

2.1 Genetic Algorithm

Genetic Algorithm (GA) was introduced by John Holland and developed based on the genetic processes of biological organisms and they are categorized as global search heuristics. Genetic algorithms are a particular class of evolutionary algorithms that use techniques inspired by evolutionary biology such as inheritance, mutation, selection, and crossover (also called recombination). Algorithm is started with a set of solutions (represented by chromosomes) called population. Solutions from one population are taken and used to form a new population. In GA, each generation consists of a population of binary strings which are called chromosomes. Every time evolution takes place, the new attributes are encoded in the chromosomes. The chromosomes with the highest fitness are always being copied into the next generation. In addition, the crossover operator enables an exchange of substrings between two parents, in order to increase the diversity of the perturbed offspring. The selection operator decides whether or not a new solution should be selected to be a member of the next generation. Meanwhile, the random modification of a new configuration is controlled by the mutation operator. The entire genetic algorithm is explained in Fig. 1. It can be used for combinatorial optimization, pattern recognition, machine learning, planning strategy and information processing [4].

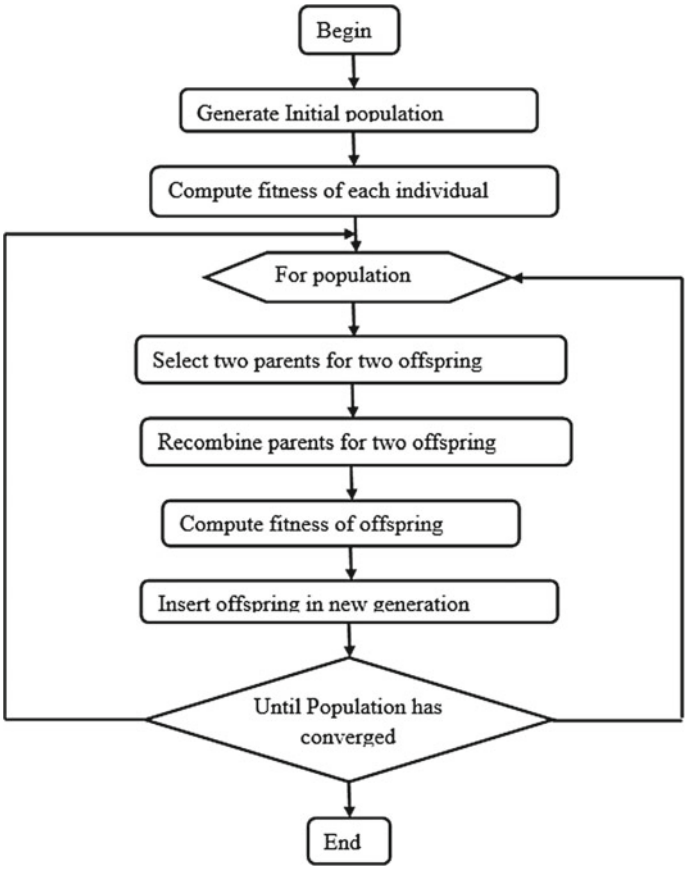


Fig. 1 Genetic algorithm

3 Swarm Intelligence

Swarm intelligence (SI) is the collective behaviour of decentralized, self-organized systems, natural or artificial [5]. The concept is employed in work on artificial intelligence. Swarm Intelligence principles have been successfully applied in a variety of problem domains including function optimization problems, finding optimal routes and scheduling. The main Swarm Intelligence based methods are Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), and Artificial Bee Colony Optimization algorithms [6].

3.1 Ant Colony Optimization

In computer science and operations research, the ant colony optimization algorithm (ACO) is a probabilistic technique for solving computational problems which can be used for finding good paths through graphs. Figure 2 shows the ant colony optimization in graphical representation.

(A. Ants in a pheromone trail between nest and food; B. an obstacle interrupts the trail; C. ants find two paths to go around the obstacle; D. a new pheromone trail is formed along the shorter path.) [7]

This algorithm is a member of ant colony algorithm family in swarm intelligence methods and it constitutes some metaheuristic optimizations [8]. Initially proposed by Marco Dorigo in 1992 in his PhD thesis, the first algorithm was aiming to search for an optimal path in a graph, based on the behaviour of ants seeking a path between their colony and a source of food. The original idea has since diversified to solve a wider class of numerical problems, and as a result, several problems have emerged, drawing on various aspects of the behaviour of ants [9]. Then ant colony optimization algorithm is explained in Fig. 3.

3.2 Particle Swarm Optimization

Particle swarm optimization is a population stochastic optimization technique for the solution of continuous optimization problems. It is a simple evolutionary algorithm which differs from other evolutionary algorithms in which it is motivated the simulation of social behaviour [10]. PSO has shown good performance in finding good solutions to optimization problems. It is inspired by the social behaviours of the flocking of birds and schools of fish. The PSO is a stochastic, population-based computer algorithm modelled on swarm intelligence. Swarm intelligence is based on

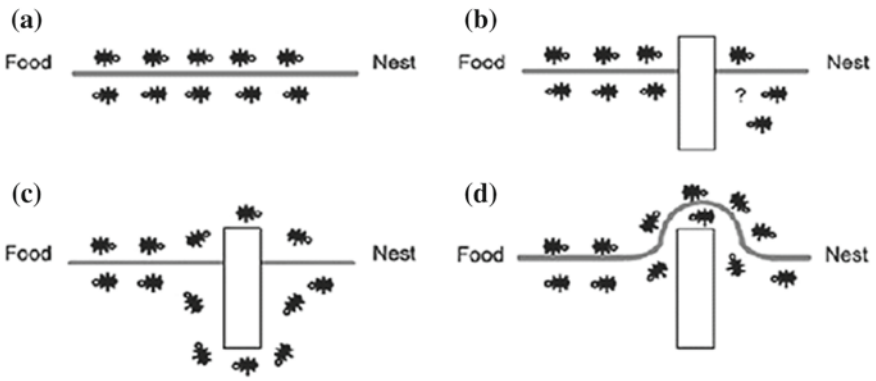


Fig. 2 Ant colony optimization graph representation

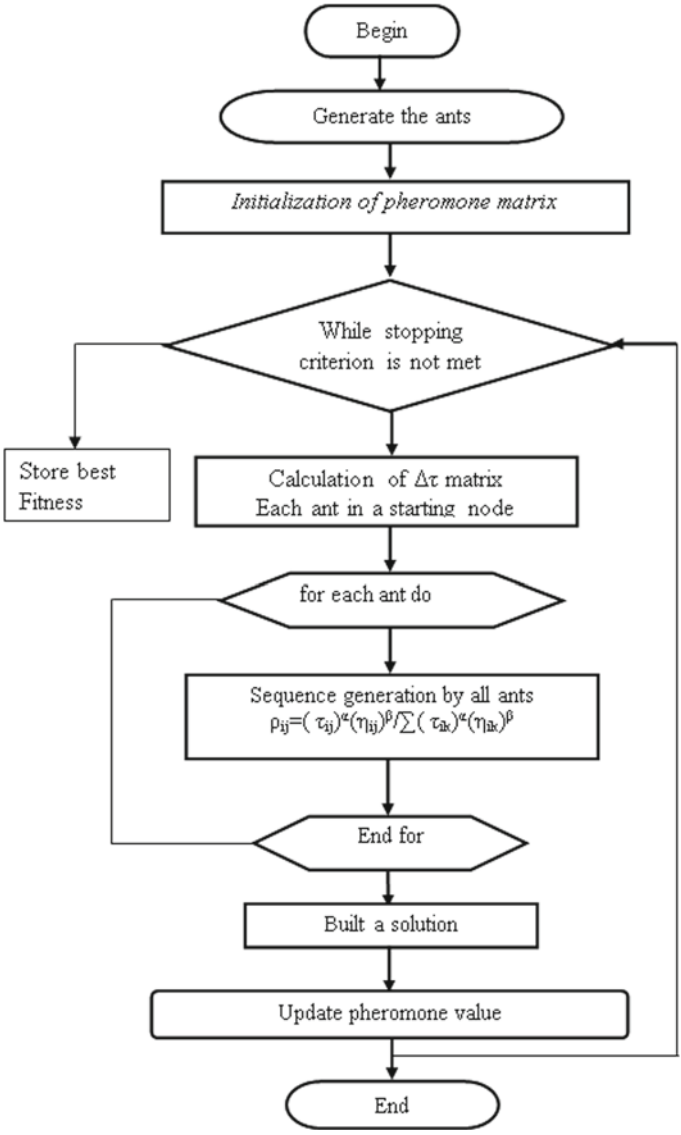


Fig. 3 Ant colony optimization

social-psychological principles and provides insights into social behaviour, as well as contributing to engineering applications [11]. In particle swarm optimization (PSO), a set of software agents called particles search for good solutions to a given continuous optimization problem. Each particle is a solution to the considered problem and uses its own experience and the experience of neighbour particles to choose how to

move in the search space. In practice, in the initialization phase of each particle is given a random initial position and an initial velocity [12]. The position of the particle represents a solution of the problem and therefore has a value, given by the objective function. While moving in the search space, particles memorize the position of the best solution they found. At each iteration of the algorithm, each particle moves with a velocity that is weighted sum of three components: the old velocity, a velocity component that drives the particle towards the location in the search space where it previously found the best solution so far, and a velocity component that drives the particle towards the location in the search space where the neighbour particles found the best solution so far [13]. PSO has been applied to many different problems and is another example of successful artificial/engineering swarm intelligence system [14].

In PSO, each single solution is a "bird" in the search space called "particle". The performance of a particle is measured by a fitness value. The PSO algorithm is similar to other evolutionary algorithms. The particles fly through the problem space by following the current optimum particles. SO is initialized with a group of random particles (solutions) and then searches for optima by updating generations. The position of a particle is influenced by the best position visited by itself i.e. its own experience and the position of the best particle in its neighbourhood i.e. the experience of neighbouring particles [15]. Each particle knows its best position pbest and the best position so far among the entire group of particles gbest. The pbest of a particle is the best result (fitness value) so far reached by the particle, whereas gbest is the best particle in terms of fitness in an entire population. When a particle takes part of the population as its topological neighbours, the best value is a local best and is called lbest. The PSO algorithm is given in Fig. 4.

3.3 Artificial Bee Colony Optimization

Inspired by the intelligent foraging behaviours of honeybee swarm, an Artificial Bee Colony algorithm is developed, which is a new population-based meta-heuristic approach [16]. In ABC algorithm, there are three kinds of foraging bees: employed bees, onlooker bees, and scout bees. The procedure followed in the ABC algorithm is as follows. The flowchart of this algorithm is given in Fig. 5.

At the first step, a randomly distributed initial population (food source positions) is generated. Provided that the nectar amount of the new one is higher than that of the previous source. The bee memorizes the new source position and forgets the old one. Otherwise it keeps the position of the one in its memory. As in the case of the employed bee, it produces a modification on the source position in its memory and checks its nectar amount providing that its nectar is higher than that of the previous one [17].

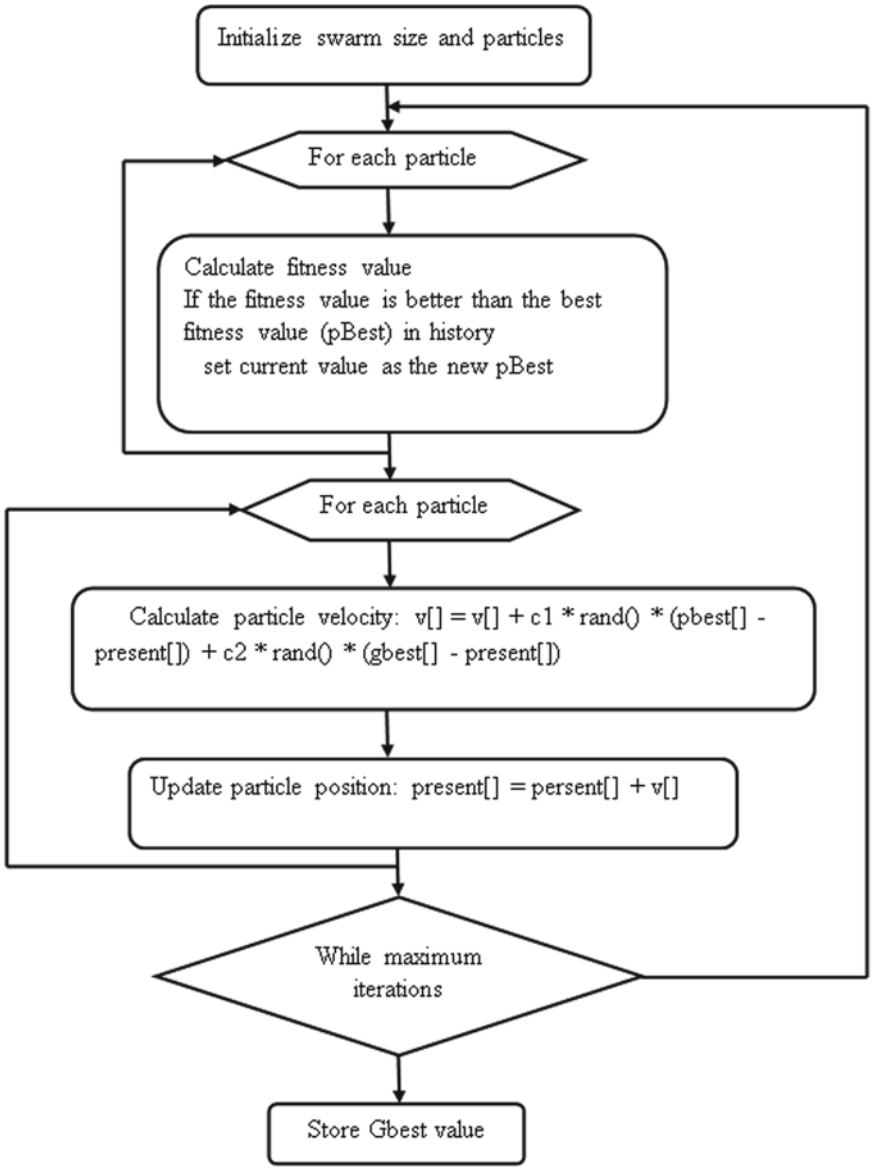


Fig. 4 Particle swarm optimization

The basic Artificial Bee Colony algorithm.

Initialization phase

Step 1: Initialize parameters, including the number of food sources or the number of employed bees, number of onlooker bees and number of scout bees.

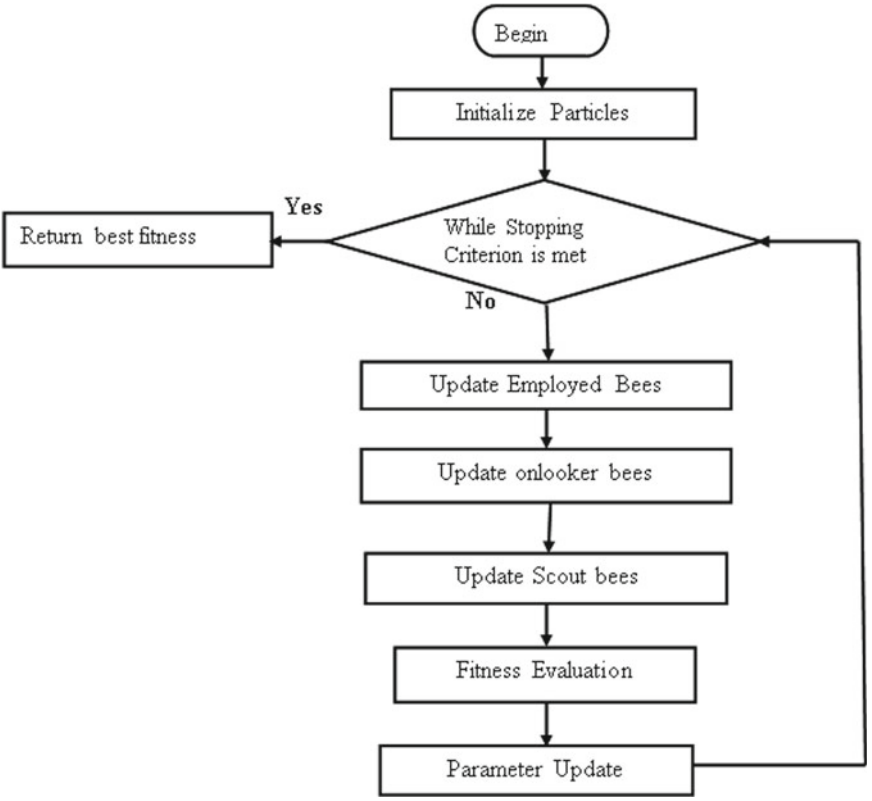


Fig. 5 Artificial bee colony optimization

Step 2: Initialize population of food sources with random solutions.
Step 3: Calculate the objective value of each food source and then determine the best food resource.

Employed bee phase

Step 4: For every employed bee, generate a new food source.
Step 5: Calculate the objective value for every new food sources and compute the best food source.

Onlooker bee phase

Step 6: Calculate the probability of selecting food source using the following formulae.
The probability Pi of selecting a food source i is determined by the following expression:

$p_i = fit_i / \sum_{i=1}^{l} fit_i$

Here, fit_i is the objective value of the solution represented by the food sources i and l is the total number of food sources. Clearly, good food sources will attract more onlookers than the bad ones.

Step 7: Calculate the number of onlooker bees to be sent to the food source.

Step 8: For every onlooker bee, generate the new food sources.

Scout bee phase

Step 9: Initialize scout bees with random solutions and update the best food sources.

Step 10: Determine the worst employed bees and replace them with the scout bees if the scout bees are better.

Step 11: If a stopping criterion is met, then the best food source is obtained with its objective value [18].

4 Related Work on Hybrid Techniques for Job Scheduling

4.1 Introduction

Most existing research work is based on Job Scheduling Problem to reduce completion time and makespan time. It is found that existing soft computing techniques such as Genetic Algorithm, Ant Colony Optimization, Particle Swarm Optimization and Artificial Bee Colony for Job Scheduling Problem do not produce efficient results. Hence the existing algorithms are combined to produce better results. They can also be used in dynamic applications and for larger optimizations. They are more efficient when compared to existing soft computing techniques.

This division is about existing hybrid techniques for job scheduling. First hybrid technique is the combination of ACO with PSO to produce an optimum scheduling. Second technique is about the combination of PSO with GA to overcome the limitations. Third technique is about the hybrid Artificial Bee Colony for job scheduling.

4.1.1 Ant Colony Optimization with Particle Swarm Optimization

It is a new hybrid algorithm for job scheduling, which is based on the frame work of ACO and the merits of PSO. An ACO finds out the sequence using transition probability and transition rule. The sequence value of ACO is used as an input to

PSO, so that the local best makespan time for each sequence can be found. The gbest is found by comparing the lbest so that the optimum sequence with minimum makespan time is found. Finally optimum scheduling is produced in a minimum makespan time.

Hybrid Ant Colony Optimization

Step 1. Generating n number of ants.(ants equals number of jobs)
Step 2. Initializing pheromone matrix at random.
Step 3. While terminated condition is not met, do the following steps

- Step 3.1. Construct the first scheduling, according to transition probabilities and transition rule.
- Step 3.2. Using transition probability rule to find the sequence generated by n movements of each ant.

Step 4. Initialize the input values for PSO. Assume sequence value of ACO as the input of PSO.
Step 5. Find local best makespan time for each sequence.
Step 6. Compare local best and find global best.
Step 7. Find optimum sequence with minimum makespan time.
Step 8. Produce the optimum scheduling and minimum makespan time.

Advantages:

1. Positive feedback leads to rapid discovery of good solutions based on the applications.
2. It can be applied in to both scientific research and engineering research problems.
3. Computation is distributed so that the premature convergence can be avoided.

Disadvantages:

1. Distribution of probability changes by iteration and the problem of scattering and optimization cannot work out by this method.
2. Cause of less speed and direction and sequences of random decisions.

4.1.2 Particle Swarm Optimization with Genetic Algorithm

The problem-dependent performance can be addressed through hybrid mechanism. To overcome the limitations of PSO, it is combined with GA to give hybrid algorithm. The basis behind this is that, such a hybrid approach is expected to have merits of PSO and those of GA. One advantage of PSO over GA is its algorithmic simplicity. To prevent the occurrence, position update of the global best particles is changed. The position update is done through a mechanism in GA. The idea behind GA is due to its genetic operator's crossover and mutation. By applying crossover operation, information can be swapped between two particles to have the ability to fly to the

new search area. The purpose of applying mutation to PSO is to increase the diversity of the population and the ability to have the PSO to avoid the local maxima.

In PSO_GA the *gbest* particle position does not change its position over some designated time steps, the crossover operation is performed on gbest particle with chromosome of GA. The total number of iteration is equally shared by GA and PSO [19]. First half of the iterations are run by GA and the solutions are given as initial population to PSO. Remaining iteration is run by PSO.

Particle Swarm Optimization with Genetic Algorithm

Step 1. Initialize swarm size and particle values for each particle (Initialize the number of resources and number of jobs)
Step 2. Choose the particle with the best fitness value of all the particles as the gbest
Do
Begin
Step 3. For each particle:
Step 3.1. Perform two-point crossover for a Job. Choose a random point exchange machine assignments.
Step 3.2. Mutation: Randomly select a job. Randomly, reassign it to the new machine.
Step 4. Update particle velocity:
Step 5. Calculate particle velocity
Step 6. Update particle position

End

While maximum iterations or minimum error criteria are not attained (or the process is repeated until the stopping criteria is met. (Best fitness, minimum completion time)

Advantages:

1. Every optimization problem can be solved with chromosome encoding and should have no overlapping and mutation calculation.
2. Can solve multidimensional non differential problems and search can be carried out by the speed of the particle.

Disadvantages:

1. There is no time to convergence uncertain and the problems of non coordinate system cannot be worked out.
2. The method easily suffers from the partial optimism.

4.1.3 Hybrid Artificial Bee Colony (ABC) Algorithm

In hybrid ABC algorithm, new food sources are generated by accomplishing the crossover and mutation operations. The crossover and mutation are the genetic algorithm operations. In this existing hybrid ABC method, crossover and mutation operator is added after the employed bee phase of the Artificial Bee Colony (ABC) algorithm. ABC algorithm has four phases namely initialization phase, employed bees phase, onlooker bees phase and scout bees phase, adding mutation phase after the employed bees phase. Employed bees phase do the local search and mutation after the employed bees phase explore the search space and search for new area of solution space. Through mutation, on the one side, there is a chance of changing the local best position and the algorithm may not be trapped into local optima. In this method, the mutation step is carried out on the probabilistic way in each food searching operation for each iteration during the life cycle of ABC optimization technique. Food Source is selected arbitrarily from the food size and mutation is performed. In mutation, the generated offspring replace the older offspring. The mutation operator used here is uniform mutation. When performing mutation, food source x_{ij} is randomly selected and replaces one of the dimension values by random number generated between the lower and upper bound value of the food source [20].

The procedure that is followed in the adaptive ABC algorithm is as follows:

Initialization phase

Step 1. Initialize input parameters, including the number of food sources or the number of employed bees e_b, number of onlooker bees o_b and number of scout bees s_b.
Step 2. Initialize populations by generating random food sources f_s.
Step 3. Calculate the fitness value $F(f_s)$ of each food source f_s and then determine the best food resource bf_s.

Employed bee phase

Step 4. For every employed bee, generate a new food source $Nf_s(e_b)$ by using the crossover, mutation operations.
Step 5. Calculate fitness value $F(f_s(e_b))$ for every newly generated food source and compute the best food source.

Onlooker bee phase

Step 6. For every onlooker bee, generate a new food source $Nf_s(o_b)$ by using the crossover, mutation operations.
Step 7. Calculate fitness value $F(f_s(o_b))$ for every newly generated food source and compute the best food source.

Scout bee phase

Step 9. Initialize scout bees with random solutions and compute fitness value $F(f_s(s_b))$ for these random solutions.
Step 10. Find the best scout bee among the randomly generated food sources using the fitness value $F(f_s(s_b))$.
Step 11. The scout bee's best food source $B(f_s(s_b))$, the employed bee's best food source $B(f_s(s_b))$ and the onlooker bee's best food source $B(f_s(s_b))$ are compared based on their fitness values.
Step 12. Among these food sources, the best food source is stored in the scout bee's phase and remaining food sources are given to the next iteration.
Step 13. The process is repeated until the stopping criterion is met. Then, the best food source is obtained with its objective value from the scout bee's phase.

Advantages

1. Positive feedback accounts for rapid discovery of good solution and the bees algorithm is more stable which requires less computation time to complete task.
2. Bees can return to home through direct root instead of back tracking their original route.

Disadvantages:

1. It takes longer computation time and less guaranteed convergence.
2. Certain optimization problems (they are called variant problems) cannot be solved.
3. The population of solutions increases the computational cost due to slowdown, many iterations and memory capacity required.

4.2 Experimental Results and Discussion

The existing hybrid algorithms for job scheduling problem is implemented in the working platform of JAVA (6) and Netbeans 7.2.1 with machine configuration as follows: Processor: Intel core i3, OS: Windows 7, CPU speed: 2.20 GHz, RAM: 3 GB

The experiments, by varying the number of resources with varying number of jobs are conducted and then the results with that of three existing hybrid algorithms are compared. The comparative results are tabulated in Table 1 and charts are shown in Figs. 6, 7 and 8.

Table 1 Mean Completion time for 10 jobs with 50, 100, 150, 200 and 250 resources for, PSO-GA, PSO-ACO and ABC-GA algorithm

Iterations	Resources	ACO-PSO	PSO-GA	ABC-GA
50	50	249.8	151.2	160.3
	100	196.5	237.3	224.9
	150	116.2	110.5	96.2
	200	128	117.6	110.9
	250	196.3	188.9	199.3
100	50	253.5	266.5	161.9
	100	225.5	223.9	227.6
	150	115.7	85.9	98.6
	200	130.3	130.4	115.4
	250	191	192.3	193.9
200	50	197.3	224.2	127.3
	100	137.4	177.9	183.1
	150	93.4	72.5	74
	200	101.4	104.3	91.3
	250	146.8	176	173.5

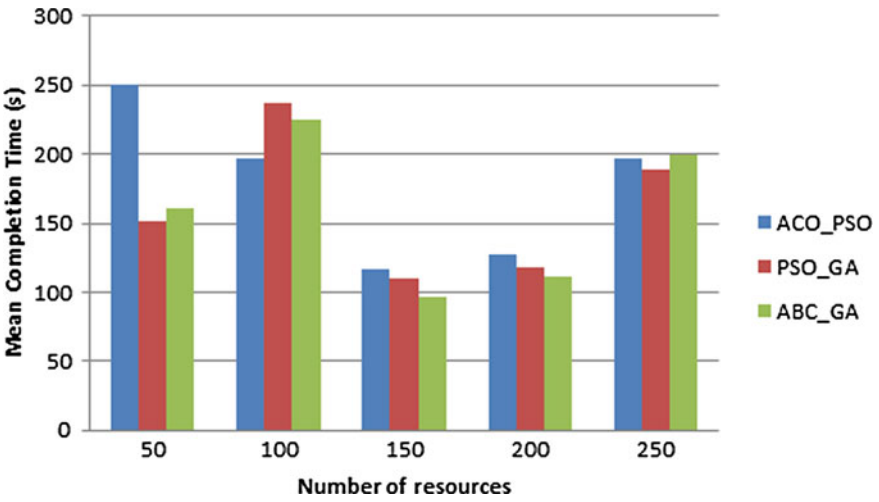


Fig. 6 Mean completion time for 10 Jobs with 50 iterations for ACO_PSO, PSO_GA and ABC_GA

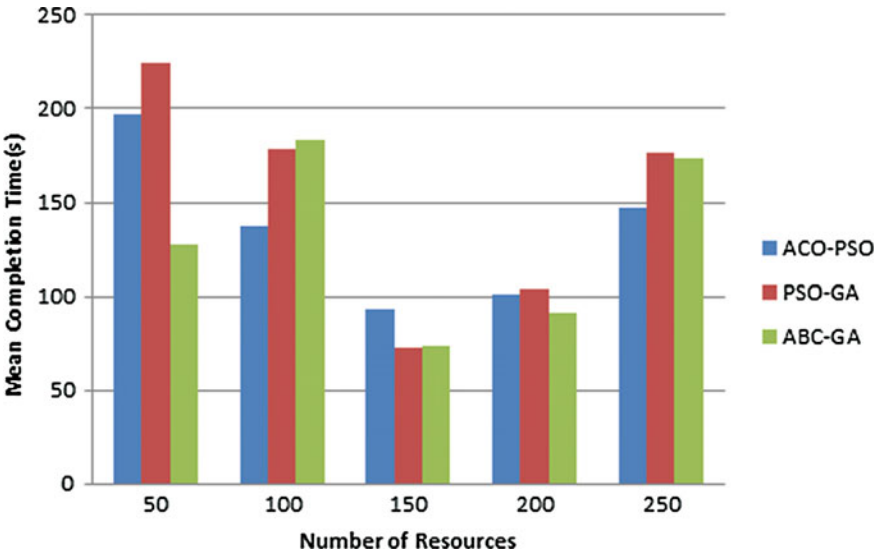


Fig. 7 Mean completion time for 10 Jobs with 100 iterations for ACO_PSO, PSO_GA and ABC_GA

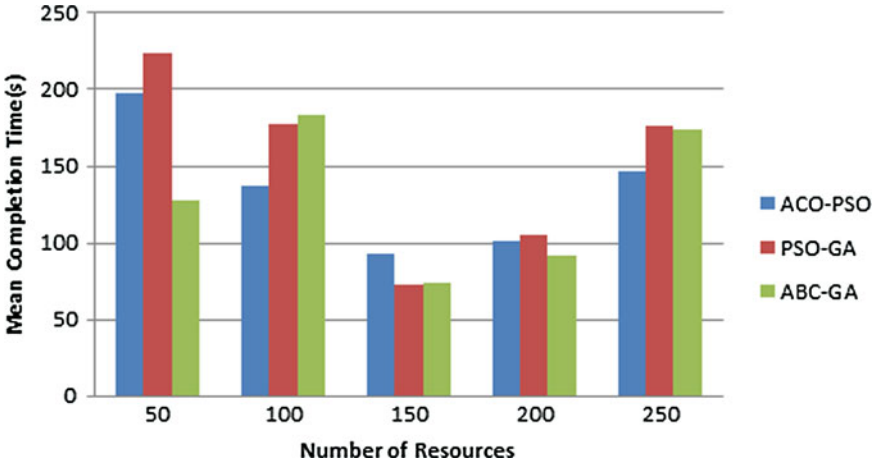


Fig. 8 Mean completion time for 10 Jobs with 200 iterations for ACO_PSO, PSO_GA and ABC_GA

5 Proposed Hybrid Technique

5.1 Introduction

The job scheduling is one of the active research fields, where the researchers work to enhance the efficiency of the job scheduling process in a scheduling environment. In existing hybrid techniques ACO_PSO, PSO_GA and ABC_GA some efficient factors related to jobs like turnaround time, job execution time are not taken in to account in the job scheduling process. The performance has been reduced due to the lack of considering such factors. An adaptive ABC technique is proposed to overcome such drawbacks in the existing methods. In this proposed adaptive ABC technique, the adaptiveness is achieved by using mutation, crossover and velocity in the employed bee phase for finding the new food sources. Moreover, these existing techniques mostly concentrate on two factors such as the minimization of the makespan and the completion time.

5.2 Adaptive ABC Algorithm

Here, an Adaptive Artificial Bee Colony (AABC) algorithm by making the adaptiveness in their new food source generation is introduced. In AABC algorithm, new food sources are generated by accomplishing the crossover, mutation, and velocity operations. The crossover and mutation are the genetic algorithm operations and velocity is related to the PSO technique. In PSO, the velocity formula has been utilized to generate new particles. These two optimization techniques, i.e., new chromosome and particles generation operations are incorporated into the standard ABC algorithm to make the adaptiveness as well as to optimize the performance. The procedure followed in the adaptive ABC algorithm is as follows: The flowchart of this algorithm is given in Fig. 9.

Initialization phase

Step 1. Initialize input parameters, including the number of food sources or the number of employed bees e_b, number of onlooker bees o_b and number of scout bees s_b.
Step 2. Initialize populations by generating random food sources f_s.
Step 3. Calculate the fitness value $F(f_s)$ of each food source f_s and then determine the best food resource bf_s.

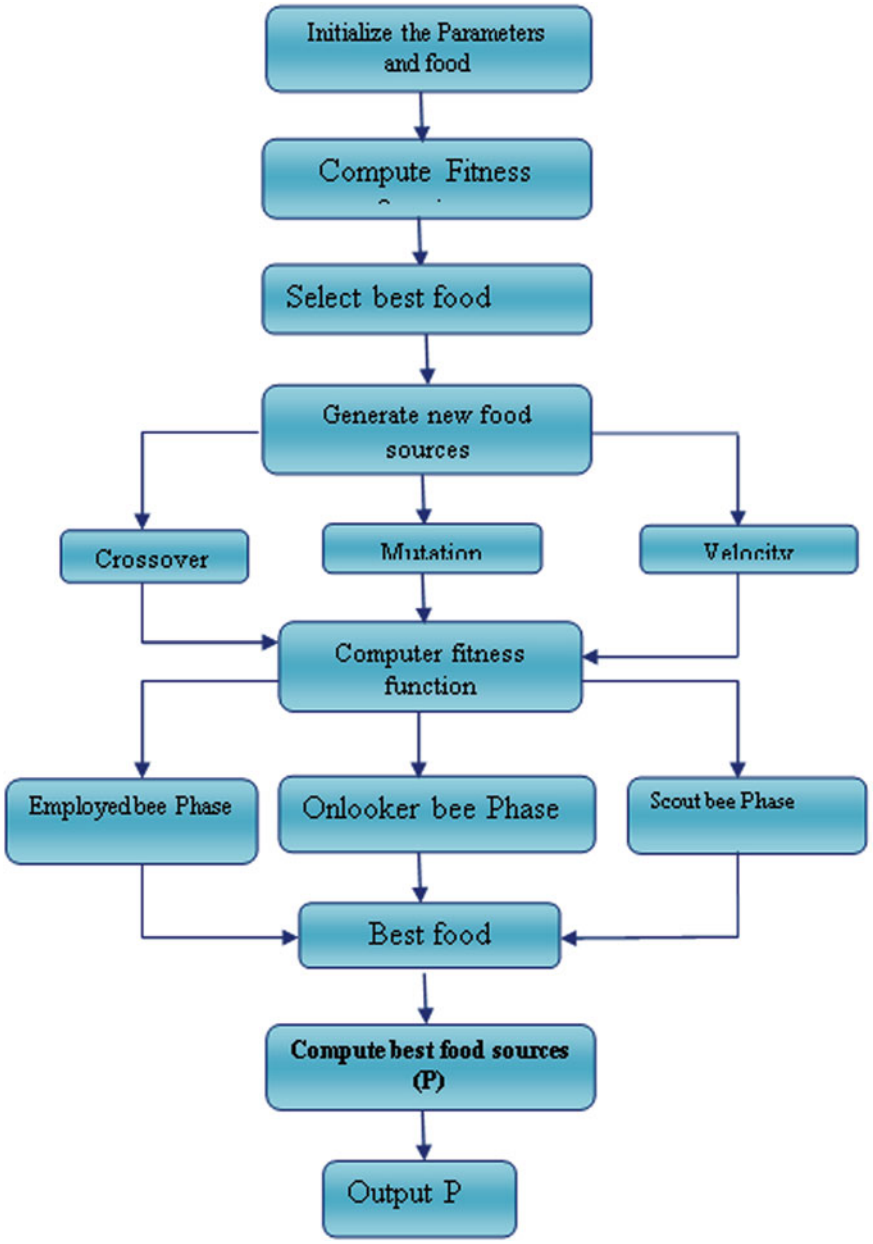


Fig. 9 The proposed adaptive ABC algorithm framework

Employed bee phase

Step 4. For every employed bee, generate a new food source $Nf_s(e_b)$ by using the crossover, mutation and velocity operations.
Step 5. Calculate fitness value $F(f_s(e_b))$ for every newly generated food source and compute the best food source.

Onlooker bee phase

Step 6. For every onlooker bee, generate a new food source $Nf_s(o_b)$ by using the crossover, mutation and velocity operations.
Step 7. Calculate fitness value $F(f_s(o_b))$ for every newly generated food source and compute the best food source.

Scout bee phase

Step 9. Initialize scout bees with random solutions and compute fitness value $F(f_s(s_b))$ for these random solutions.
Step 10. Find the best scout bee among the randomly generated food sources using the fitness value $F(f_s(s_b))$.
Step 11. The scout bee's best food source $B(f_s(s_b))$, the employed bee's best food source $B(f_s(s_b))$ and the onlooker bee's best food source $B(f_s(s_b))$ are compared based on their fitness values.
Step 12. Among these food sources, the best food source is stored in the scout bee's phase and remaining food sources are given to the next iteration.
Step 13. The process is repeated until the stopping criterion is met. Then, the best food source is obtained with its objective value from the scout bee's phase.

5.3 Adaptive ABC for Job Scheduling Problem

The adaptive ABC algorithm is initiated by generating food sources f_s. The food sources are represented as $f_{s_i} = \{R_1, R_2, \ldots, R_k\}; k \in M$, where i represents the number of generated food sources. In this food source, each resource has the allocated jobs based on their execution and processing times e_n and $p_{n,m}$. The generated food source representation for the proposed job scheduling problem is shown in Fig. 10.

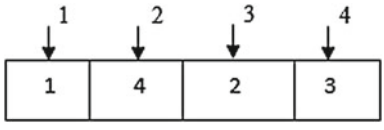


Fig. 10 Generated food source representation

After that, the fitness value is computed for the food sources. The fitness function calculation formula is given below,

$$p_i = \frac{fit_i}{\sum_{i=1}^{l} fit_i} \tag{1}$$

$$F(f_{si}) = \frac{1}{(\omega \times (1/S)) + (\alpha \times D)} \tag{2}$$

$$S = MAX(p_{j,k}) \tag{3}$$

$$D = \sum T(j) \tag{4}$$

$$T(j) = d_j - t_j \tag{5}$$

In Eq. 2, S and D represents the makespan and delay time of the resources and jobs respectively. These makespan and delay times are multiplied with the constant factor values ω and α. In Eq. 3, the maximum processing time is calculated from the set of resources in the food sources and in Eq. 5 the jobs total delay time is calculated by utilizing the jobs deadline time d_j and turnaround time t_j. Based on the fitness values, the best food sources (b_s) having minimum fitness value are selected from the generated i number of food sources. Next, the new food sources are generated in employed bee phase by using the following three operations Crossover Mutation and Velocity.

Pseudo code for fitness

```
double findFitness(ArrayList<Resource> resources) {
    double max=findMax(resources);
    double D=getD(resources);
    double alpha=0.5;
    double omega=0.2;
    double fitness=1/((omega*(1/max))+(alpha*D));

    if (fitness < 0) {
        fitness=(1+Math.abs(fitness));
    } else {
        fitness=(1/(1+fitness));
    }
    return fitness;
}
```

Crossover and Mutation in the employed bee phase
In the adaptive ABC algorithm, the crossover, mutation, and velocity operations are performed for generating the new food sources. A uniform crossover process is performed here to generate the new food sources, which is illustrated as follows.

Crossover

Step 1. For every employed bee, we select one best food source from b_s as u' and select the employed bee solution from the best population $u''(u', u' \in b_s$ and $u' \neq u')$.
Step 2. Uniformly generate a binary string i.e., 0 and 1 with the same length of the food source. Then, generate new food source by placing the element of u'' at position with bit 1 and placing the element of u' at position with bit 0.

Pseudo code for crossover

```
for (int a=0; a<bstring.length; a++) {
    if (bstring[a]==0) {
        newfoodsource.add(bestfoodsource.resources.get(a));
    } else {
        newfoodsource.add(bestpopulation.resources.get(a));
    }
}
```

Mutation
An integer r is randomly generated from 1 to N, in mutation, where N is the total number of jobs. Based on the generated integer value, letter position is selected from the food sources. For each selected position, replace the resources with a randomly chosen different resource from the candidate resource set. The generated new food sources from the crossover and mutation operations are $e^1(b_s)$. Then, the fitness value is calculated for these new food sources by utilizing the formula described in Eq. 2.

Pseudo code for mutation

```
int pos=r.nextInt(newfoodsource.size());
for (int x=0; x<pos; x++) {
    newfoodsource.add(newfoodsource.get(x));
}
```

Velocity in the employed bee phase
In PSO optimization algorithm, a velocity value is utilized for generating the new particles. In employed bee phase, the same velocity formula is exploited for generating the new food sources. Based on the velocity value, the new food sources are generated, which is represented as $e^2(b_s)$. Subsequently, the fitness value for this $e^2(b_s)$ food source is computed by exploiting (2). In employed bee phase, the new food sources are $e^1(b_s)$ and $e^2(b_s)$. From this set of food sources, some of the best food sources having minimum fitness value are selected. The selected food sources from employed bee phase are $e(b_s)$.

Similarly in the onlooker and scout bee phases, the new food sources are generated based on the aforementioned crossover, mutation, and velocity operations. The new best food sources are computed by the fitness function values and these selected food sources are represented as $o(b_s)$ and $s(b_s)$. Among the food sources obtained from these three phases, select one best food source having minimum fitness value, which is denoted as and the remaining food sources are given to the next iteration. This process is repeated until it reaches the maximum number of iterations I. The obtained best food sources are the optimal solutions for the jobs.

Pseudo code for velocity

```
double value=(wmax*velocity[a][b])+((pbest.resources.get(b).getJob().name -
    bestpopulationsource.get(a).resources.get(b).getJob().name)*c1*r.nextDouble())
    + ((gbest.getResources().get(b).getJob().name - bestpopulationsource.get(a)
    .resources.get(b).getName())*c2*r.nextDouble());
velocity[a][b]=velocity[a][b]+value;
```

6 Experimental Results

Table 2, Figs. 11 and 12 show the tested completion time and makespan time in Real Time Environment.

6.1 Environment Infrastructure

The comprehensive infrastructure includes 8 Servers (Rack and Tower), 950 Desktops (Dell, HP, Acer and Lenovo), 100 Printers and over 1,000 Laptops (Dell, HP, IBM and Lenovo). All the computers are connected to a switched network. It maintains

Table 2 Completion time and makespan time for 10 jobs with 50 resources

Iterations	Completion time	Makespan time
1	1471	29.42
2	935.5	18.71
3	1326	26.52
4	1111.5	22.23
5	935	18.70
6	1367.5	27.35
7	978	19.56
8	1016.5	20.33
9	1563.5	31.27
10	1150	23.00

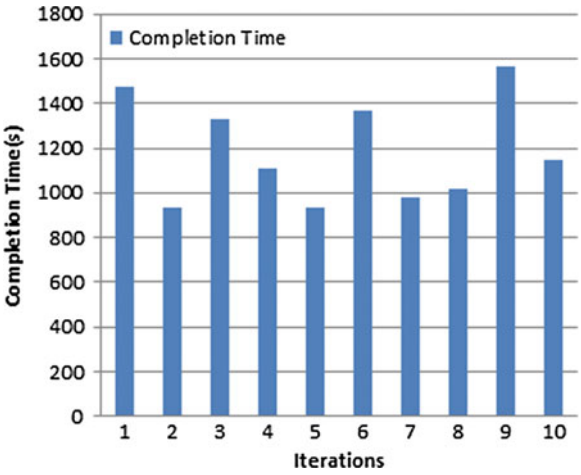


Fig. 11 Completion time for 10 jobs with 50 resources

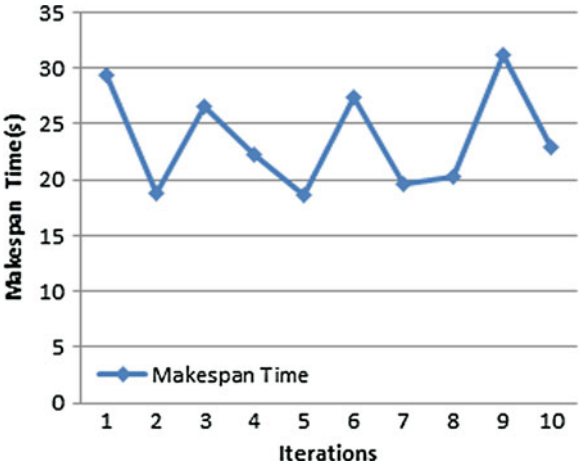


Fig. 12 Makespan time for 10 jobs with 50 resources

a firewall for internet security. Entire campus network is configured and connected through centralized server room.

6.2 Implementation and Results

are some control parameters for GA, ACO and PSO algorithms. For every algorithm, there are some control parameters, which are used for its efficient working. An extensive literature survey is done and carried out an experiment for determining the values of these control parameters. From this it is found that the values which have been taken in this experiment are standard values and they are also suitable for these

Table 3 Parameter settings for the algorithms

Algorithm	Parameter	Value
GA	Size of the population	User defined
	Probability of crossover	0.8
	Probability of mutations	0.5
ACO	Swarm size	User defined
	Weight of pheromone trail α	0.5
	Weight of heuristic	0.2
	Pheromone evaporation	0.5
PSO	Swarm size	User defined
	Self-recognition coefficient c1	0.2
	Social coefficient c2	0.3
	Inertia weight w	0.9
ABC	Swarm size	User defined
	Number of employee bees	50 % of swarm size
	Number of onlooker bees	50 % of swarm size
	Number of scout bees	1

experiments. Different values for the different parameters in each of the algorithms and system parameters are shown in Table 3.

The result obtained by the proposed algorithm is analyzed. To test the efficiency of the algorithm results of AABC is compared with results of the three hybrid algorithms viz., ACO_PSO, PSO_GA and ABC_GA. The experiments by varying the number of resources as well as varying the number of jobs are conducted and then the results with that of three existing hybrid algorithms are compared. The comparative results are shown in the following tables. Table 4 shows the mean makespan time for 10 jobs with 50, 100, 150, 200 and 250 resources for the existing and proposed algorithms. Figures 13, 14 and 15 show the corresponding graphs.

The various **hybrid swarm intelligence algorithms** (ACO_PSO, PSO_GA and ABC_GA) were compared with the proposed AABC algorithm and the performance of makespan and completion time for solving job scheduling problem were analyzed. Table 5 shows the mean completion time for 10 jobs with 50, 100, 150, 200 and 250 resources for the existing and proposed algorithms. Figures 16, 17 and 18 show the corresponding graphs.

Table 4 Mean makespan time for 10 jobs with 50, 100, 150, 200 and 250 resources for ACO-PSO, PSO-GA, ABC-GA and AABC algorithms

Iterations	Resources	ACO_PSO	PSO-GA	ABC-GA	AABC
50	50	32.7	32.3	34	28.3
	100	34	38	37.6	26
	150	24	27.9	32	22.6
	200	25.8	29.1	26.4	23.3
	250	24.3	31.7	27.5	23.2
100	50	33.4	34.2	35.6	28.3
	100	31.3	36.2	31.8	29.1
	150	29	27	26.7	25.2
	200	31.3	33.4	37.2	24
	250	32.5	29.2	31	27.3
200	50	27.3	21.8	24.3	21.2
	100	23.5	22.6	27.3	18.7
	150	19.7	18	21	16.3
	200	21.3	19.6	22.5	17
	250	25.6	21.6	23.6	19.2

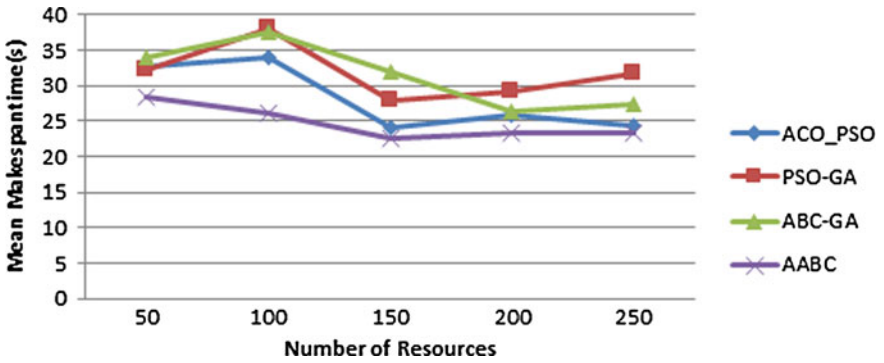


Fig. 13 Mean makespan time for 10 jobs with 50 iterations for ACO_PSO, PSO_GA, ABC_GA and AABC

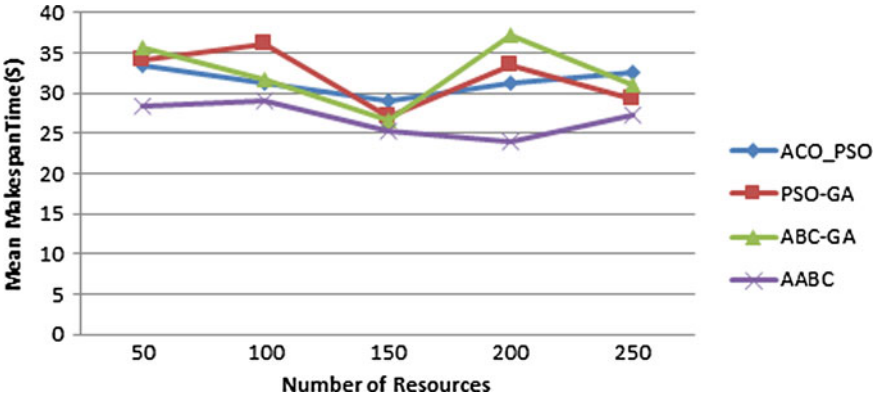


Fig. 14 Mean makespan time for 10 jobs with 100 iterations for ACO_PSO, PSO_GA, ABC_GA, and AABC

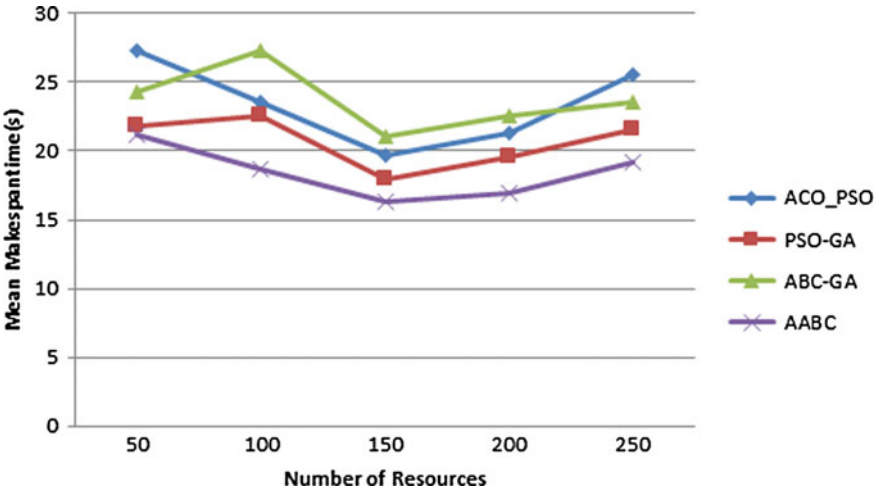


Fig. 15 Mean makespan time for 10 Jobs with 200 iterations for ACO_PSO, PSO_GA and ABC_GA, and AABC

Table 5 Mean completion time for 10 Jobs with 50, 100, 150, 200 and 250 resources for PSO-ACO, PSO-GA, ABC-GA, AABC algorithm

Iterations	Resources	ACO_PSO	PSO-GA	ABC-GA	AABC
50	50	249.8	151.2	160.3	141.8
	100	196.5	237.3	224.9	191.5
	150	116.2	110.5	96.2	85.6
	200	128	117.6	110.9	97.3
	250	196.3	188.9	199.3	190.8
100	50	253.5	266.5	161.9	149
	100	225.5	223.9	227.6	216
	150	115.7	85.9	98.6	78.8
	200	130.3	130.4	115.4	99
	250	191	192.3	193.9	185.9
200	50	197.3	224.2	127.3	119
	100	137.4	177.9	183.1	130.4
	150	93.4	72.5	74	63.1
	200	101.4	104.3	91.3	73.4
	250	146.8	176	173.5	139.8

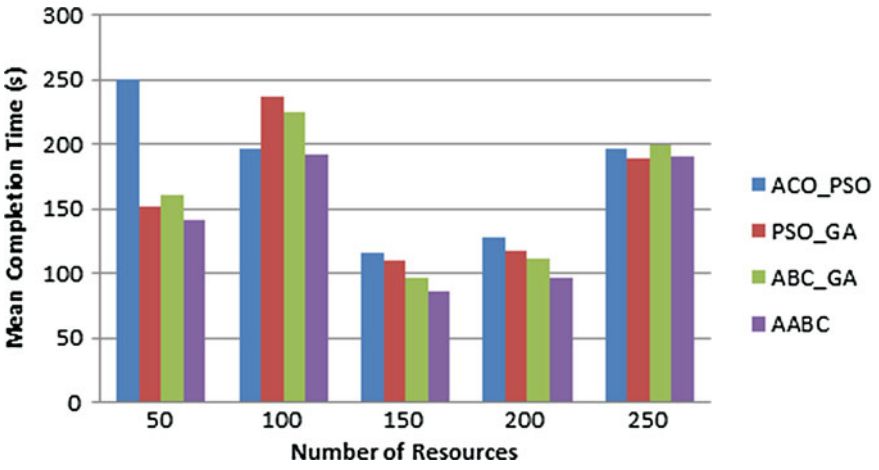


Fig. 16 Mean completion time for 10 jobs with 50 iterations for ACO_PSO, PSO_GA, ABC_GA and AABC

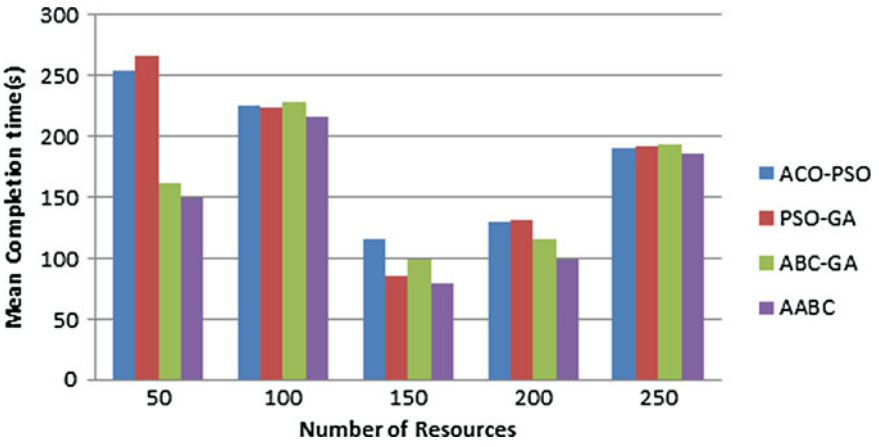


Fig. 17 Mean completion time for 10 jobs with 100 iterations for ACO_PSO, PSO_GA, ABC_GA and AABC

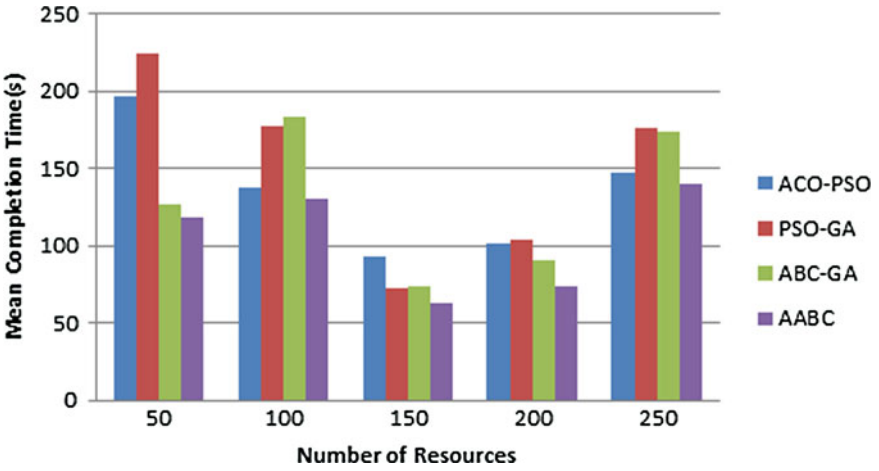


Fig. 18 Mean completion time for 10 jobs with 200 iterations for ACO_PSO, PSO_GA, ABC_GA and AABC

7 Multiobjective Problems

Multi-objective optimization (also known as multi-objective programming, vector optimization, multicriteria optimization, multiattribute optimization or Pareto optimization) is an area of multiple criteria decision making, that is concerned with mathematical optimization problems involving more than one objective function to be optimized simultaneously. Multi-objective optimization has been applied in many fields of science, including engineering, economics and logistics where optimal decisions need to be taken in the presence of trade-offs between two or more conflicting objectives. Minimizing weight while maximizing the strength of a particular component, and maximizing performance whilst minimizing fuel consumption and emission of pollutants of a vehicle are examples of multi-objective optimization problems involving two and three objectives, respectively. In practical problems, there can be more than three objectives.

Multi objective optimization differs from the single objective optimization in many ways. For two or more conflicting objectives, each objective corresponds to a different optimal solution, but none of these trade-off solutions is optimal with respect to all objectives. Thus multi-objective optimization does not try to find one optimal solution but all trade-off solutions. Apart from having multiple objectives, the fundamental difference is that multi-objective optimization deals with two goals. The first goal is to find a set of solutions as close as possible to Pareto optimal front. The second goal is to find the set of solutions as diverse as possible.

The proposed **Adaptive Artificial Bee Colony Optimization** algorithm combines the crossover, mutation, and velocity operations. The crossover and mutation are the genetic algorithm operations and velocity is related to the PSO technique. In PSO, the velocity formula has been utilized to generate new particles. These two optimization techniques, i.e., new chromosome and particles generation operations are incorporated into the standard ABC algorithm to make the adaptiveness as well as to optimize the performance. Since the features taken from the existing algorithms, i.e. crossover, mutation and velocity pertain to single objective optimization, the proposed hybrid system outperforms the existing hybrid algorithms for single objective problems. For multi objective problems, the crossover, mutation and velocity factors of the proposed hybrid technique are to be modified to suit problems with multiple objectives.

The proposed technique finds out the way for the assignment of jobs to the resources by achieving maximum performance in minimum completion time. The adaptive ABC algorithm is initiated by generating food sources f_s. The food sources are represented as $f_{si} = R1, R2, \ldots, Rk; k \in M$, where i represents the number of generated food sources. In this food source, the resources have the allocated jobs based on their execution and processing times e_n and p_{nm}.

Crossover and mutation in the employed bee phase

In the ABC algorithm, employed bees perform global exploration with multiple different neighbourhoods for promising food sources over the entire region. Since the MOP consists of machine assignment and operation sequence, the crossover

and mutation operators are to be designed to evolve the machine assignment and operation sequence.

Crossover and mutation for machine assignment
To evolve the machine assignment, more than one crossover and mutation operators are applied with equal probability, i.e., the two-point crossover and mutation and uniform crossover and mutation. These crossover and mutation operators change the machine assignment but do not change the operation sequence. To the two feasible parents, the offspring generated by these crossover and mutation operators is feasible.

To enhance the exploration capability in the employed bee search phase, more than one mutation operator for machine assignments are proposed and embedded in the MOP. To reduce the computation load, the following mutation procedure may be used in the MOP with a probability of 50 %:

Step 1. Randomly generate an integer i from 1 to n, where n is the total number of operations.
Step 2. Randomly select two or more positions from the machine assignment vector.
Step 3. For each selected position, replace the machine with a different machine randomly chosen from the candidate machine set.

8 Conclusion

The proposed **Adaptive Artificial Bee Colony (AABC)** algorithm is a hybrid job scheduling algorithm which has achieved the minimum completion and makespan time. The existing hybrid techniques for job scheduling such as ACO_PSO, PSO_GA and ABC_GA failed to achieve the required result since they have not considered certain efficient factors like (i) turnaround time (ii) throughput and (iii) response time in the scheduling process. The proposed technique has achieved high performance in allocating the available jobs to the precise resources by considering these efficient factors. The performance of the proposed job scheduling technique was analyzed with the three existing hybrid techniques namely, ABC with GA, PSO with GA and ACO with PSO and the experimental results proved that the proposed job scheduling technique attained high accuracy and efficiency than the existing hybrid techniques. The proposed algorithm combines the velocity value for generating the new solutions. It also makes use of the crossover and mutation operators during the replacement processes. The proposed hybrid technique performed well with these two factors and allocates jobs to the resources and increases the resource utilization. Hence, the proposed adaptive ABC job scheduling technique is capable of finding the optimal jobs to the resources and also in achieving the minimum completion and makespan time.

References

1. Girish, B.S., Jawahar, N.: A scheduling algorithm for flexible job shop scheduling problem. In: 5th Annual IEEE Conference on Automation Science and Engineering, Bangalore, India, 22–25 August 2009
2. Shaa, D.Y., Linb, H.-H.: Multi-objective PSO for job-shop scheduling problems. In: 5th Annual IEEE Conference on Automation Science and Engineering, Bangalore, India, 22–25 August 2009
3. Omar, M., Baharum, A., Hasan, Y.A.: A job-shop scheduling problem (JSSP) using genetic algorithm (GA). In: Proceedings of the 2nd IMT-GT Regional Conference on Mathematics, Statistics and Applications, University Sains Malaysia, Penang, 13–15 June 2006
4. Akhshabi, M., Akhshabi, M., Khalatbari, J.: Parallel genetic algorithm to solving job shop scheduling problem. J. Appl. Sci. Res. **1**(10), 1484–1489 (2011)
5. Vaisakh, K., Srinivas, L.R.: Unit commitment by evolving ant colony optimization. Int. J. Swarm Intell. Res. **1**(3), 67–77 (2012). Online publication date: 1-Sep-2012
6. Martens, D., Baesens, B., Fawcett, T.: Editorial survey: swarm intelligence for data mining. Mach. Learn. **82**(1), 1–42 (2011)
7. Vaisakh, K., Srinivas, L.R.: Unit commitment by evolving ant colony optimization. 207–218 (2012)
8. Wang, L., et al.: An effective artificial bee colony algorithm for the flexible job-shop scheduling problem. **60**, 303–315 (2012)
9. Zhang, H.: Ant colony optimization for multimode resource-constrained project scheduling. J. Manag. Eng. **28**(2), 150 (2012). Online publication date: 1-Jan-2012
10. Roy, R., Dehuri, S., Cho, S.B.: A novel particle Swarm optimization algorithm for multi-objective combinatorial optimization problem. Int. J. Appl. Metaheuristic Comput. (IJAMC) **2**(4), 41–57 (2012)
11. Chauhan, P., Deep, K., Pant, M.: Novel inertia weight strategies for particle Swarm optimization. Memet. Comput. **5:3**, 229–251 (2013). Online publication date: 1-Sep-2013
12. Pontani, M., Ghosh, P., Conway, B.A.: Particle Swarm optimization of multiple-burn rendezvous trajectories. J. Guid. Control, Dyn. **35:4**, 1192–1207 (2012). Online publication date: 1-Jul-2012
13. Pontani, M., Conway, B.A.: Particle Swarm optimization applied to impulsive orbital transfers. Acta Astronautica **74**, 141–155 (2012). Online publication date: 1-May-2012
14. Vaisakh, K., Srinivas, L.R., Meah, K.: Genetic evolving ant direction particle swarm optimization algorithm for optimal power flow with non-smooth cost functions and statistical analysis. Appl. Soft Comput. **13**(12), 4579–4593 (2013). Online publication date: 1-Dec-2013
15. Jansen, P.W., Perez, R.E.: Constrained structural design optimization via a parallel augmented Lagrangian particle swarm optimization approach. Comput. Struct. **89**(13–14), 1352–1366 (2011). Online publication date: 1-Jul-2011
16. Caraffini, F., Neri, F., Picinali, L.: An analysis on separability for memetic computing automatic design. Inf. Sci. **265**, 1–22 (2014). Online publication date: 1-May-2014
17. Yang, J., Tang, G., Cao, M., Zhu, R.: An intelligent method to discover transition rules for cellular automata using bee colony optimisation. Int. J. Geogr. Inf. Sci. **27**(10), 1849–1864 (2013). Online publication date: 1-Oct-2013
18. Li, J-q., et al, A hybrid artificial bee colony algorithm for flexible job shop scheduling problems. Int. J. Comput. Commun. Control **6**(2), 286–296 (2011). ISSN 1841–9836
19. Toal, D.J.J., Bressloff, N.W., Keane, A.J., Holden, C.M.E.: The development of a hybridized particle Swarm for rigging hyperparameter tuning. Eng. Optim. **43**(6), 675–699 (2011). Online publication date: 1-Jun-2011
20. Yildiz, A.R.: A new hybrid artificial bee colony algorithm for robust optimal design and manufacturing. Appl. Soft Comput. **13:5**, 2906–2912 (2013). Online publication date: 1-May-2013