

**International Series in
Operations Research & Management Science**

Burcu Adıgüzel Mercangöz *Editor*

Applying Particle Swarm Optimization

New Solutions and Cases for Optimized
Portfolios



 Springer

International Series in Operations Research & Management Science

Founding Editor

Frederick S. Hillier, Stanford University, Stanford, CA, USA

Volume 306

Series Editor

Camille C. Price, Department of Computer Science, Stephen F. Austin State University, Nacogdoches, TX, USA

Associate Editor

Joe Zhu, Foisie Business School, Worcester Polytechnic Institute, Worcester, MA, USA

More information about this series at <http://www.springer.com/series/6161>

Burcu Adıgüzel Mercangöz

Editor

Applying Particle Swarm Optimization

New Solutions and Cases for Optimized Portfolios



Springer

Editor

Burcu Adıgüzel Mercangöz
Faculty of Transportation and Logistics
Istanbul University
Avcılar/Istanbul, Turkey

ISSN 0884-8289

ISSN 2214-7934 (electronic)

International Series in Operations Research & Management Science

ISBN 978-3-030-70280-9

ISBN 978-3-030-70281-6 (eBook)

<https://doi.org/10.1007/978-3-030-70281-6>

© The Editor(s) (if applicable) and The Author(s), under exclusive license to Springer Nature Switzerland AG 2021

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

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

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

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

Preface

This book aims to provide theoretical and empirical research and application of portfolio optimization for the PSO technique. Therefore, it is hoped that this book provides the resources necessary for researchers, teachers, engineers, managers, and practitioners to adopt and implement the PSO technique in portfolio optimization with a comprehensive discussion on the issues.

The book focuses on one of the heuristic optimization techniques that proceeds from the inspiration of swarm intelligence: particle swarm optimization (PSO). The PSO is a very much popular swarm intelligence algorithm. It is a robust and well-researched optimization technique. It has its roots in artificial intelligence and animal communication strategy. It is one of the most preferred solution approaches in optimization problems due to its structure and advantages. Since its inception in the year 1995 by Eberhart and Kennedy, it is being applied to solve optimization problems in many domains, including portfolio optimization.

Optimization is the process of obtaining the best solution when performing certain operations for a given purpose. The problem of portfolio optimization is an important discipline of risk management in finance that consists in finding the optimum allocation among several assets. Constructing a highest return for a given portfolio of assets is a financial expert's indisputable problem in which investors' interest is to construct a portfolio having a balance between the investors' risk and their expectations about the portfolio returns. The general purpose of portfolio optimization is to discover an efficient frontier that yields the highest expected return on each level of portfolio risk. In reality, this problem usually deals with some constraints, such as the number of assets in a portfolio, transaction costs, and short sales. Solving this kind of problem is quite difficult because of the large amount of complex data and other constraints. In recent years, artificial intelligence techniques are mostly used in portfolio optimization. Before, optimization problems used to be defined by the mathematical functions. Due to the lack of flexibility and disadvantages of such methods, new methods have been developed and inspired by events in nature. Optimization algorithms based on natural events are called heuristic algorithms. Heuristic algorithms are the algorithms that are inspired by natural

phenomena to accomplish any purpose or goal. There is a convergence to the optimum solution in the solution space, but no definite solution can be guaranteed in these algorithms. With the rise of the use of heuristics-based methods in problem solving, heuristics-based methods are widely used in quantitative decision making.

This book is structured into two parts. In the first part, the theoretical and mathematical background of portfolio construction and PSO method is mentioned and portfolio optimization cases solved by using the PSO method are given. The second part is about other application areas of PSO to give an idea and insight to the audience. Other than portfolio optimization, PSO applications in other fields such as renewable energies, operation and planning optimization, and image segmentation are included. The book totally contains 17 theoretical and empirical chapters. Chapters 1 through 3 introduce the audience to the theoretical background of utility theory and portfolio optimization. Chapters 4 through 7 discuss theoretical idea and detailed explanation of the PSO algorithm, the advantages and disadvantages of PSO, comprehensive literature review as a comparative study of PSO, and a complete description of the PSO family applied to the portfolio optimization problem. Chapters 8 through 10 give PSO portfolio optimization case studies for different stock markets. Chapters 11 through 17 focus on different applications of PSO apart from portfolio optimization.

The details of the chapters are explained as below:

In the first chapter, titled “Utility: Theories and model,” the aim is to look at utility theory from a broad perspective. The main hypothesis in the theory of decision is that the person who is in the position of deciding is entitled to be called the “economic man.” Also, the individual acts rationally. Thus, utility is the ability to satisfy (eliminate) human needs of goods and services. Expected utility theory forms the basis of traditional finance. Expected utility theory assumes that people choose risky or uncertain opportunities by comparing the expected benefits from them. The Allais and Ellsberg paradoxes criticize expected utility theory. Kahneman and Tversky (1979) present that the expected utility axioms are violated for more reasonable lottery alternatives than in the Allais paradox and put a link between finance and psychology. The prospect theory of Kahneman and Tversky forms the basis of behavioral finance.

In the second chapter, titled “Portfolio optimization,” Markowitz's mean-variance model, which is the main model of modern portfolio theory, is explained and mathematical representations are given. The subject is supported with mathematical notations by mentioning concepts such as portfolio risk and return, efficient frontier, utility theory, asset allocation, indifference curves, Sharpe ratio, and coefficient of variation.

In the third chapter, titled “Behavioral portfolio theory,” the aim is to explain behavioral portfolio theory in a theoretical way. The chapter starts with a definition of portfolio which is a financial asset that consists of various securities such as stocks and bonds, and derivative products, held by a particular person or group. Behavioral portfolio theory (BPT) emerged as a descriptive alternative to Markowitz's mean-variance portfolio theory. BPT connects two issues: the creation of portfolios and the design of securities. There are two versions of the BPT model. The first is the single

mental accounting (BPT-SA) in which the portfolio is integrated into one mental accounting and the multiple mental accounting (BPTMA).

The fourth chapter titled, “A comparative study on PSO with other metaheuristic methods,” is a comprehensive literature review as a comparative study of PSO algorithm with the most popular metaheuristic algorithms. The studies carried out between the years 2010 and 2020 on the comparison of PSO to some other metaheuristic algorithms are examined and evaluated in terms of the rates of these studies according to the following criteria: their publishing years, the metaheuristic algorithms that are compared to PSO, performance evaluation of the compared algorithms, examined metaheuristic algorithms with their inspirational approaches and their initial proposed studies, the field of subjects where the algorithms are applied in these studies, and the used databases in the examined studies. The intention of this chapter is to be useful for the researchers who want to conduct research on the PSO and other metaheuristic algorithms as it covers the essential and helpful analysis of the related research. Since it is an area of active research, there is no doubt that more metaheuristic algorithms and new applications will emerge in the future. Therefore, this chapter will be a handbook for the researchers who want to study this subject, and it will be beneficial in this respect.

The fifth chapter, titled “Mathematical model of particle swarm optimization: For numerical optimization problems,” is intended to be an introduction to the mathematical bases for the PSO algorithm as applied to numerical optimization problems. The work also covers some background on relevant topics and terms in computational intelligence, swarm intelligence, PSO, and optimization. An example of a single objective problem (DeJong’s test function 1) has been chosen from the literature. The basic PSO algorithm, equations, supporting preconditions, and parameter settings are explained in detail. An open library for PSO in Python was used to illustrate the function landscape and the trajectory of the solutions obtained via the PSO. A discussion for the advanced reader on variants of PSO is included. Problems like the portfolio optimization problem are multi-objective (MOP) in nature; hence, a brief discussion on the concepts concerning multi-objective optimization is presented for the advanced reader.

The sixth chapter, titled “Particle swarm optimization: The foundation,” lays the basic PSO foundation and introduces existing PSO variants for researchers who want to solve the portfolio optimization problem. It starts with the introduction of PSO, describing the advantages, disadvantages, and applications of PSO. Later, the basic PSO procedure and its parameter selection mechanisms are presented. The chapter also presents three popular applications of PSO in finance, including portfolio optimization. Finally, the chapter ends by introducing the existing PSO variants to solve the portfolio optimization problem.

In the seven chapter, titled “The PSO family: Application to the portfolio optimization problem,” a complete description of the PSO family applied to the portfolio optimization problem is provided. The combination with a model reduction method to find the set of selected assets provides a very flexible method to improve the portfolio management.

In the eighth chapter, titled “Constrained portfolio selection by particle swarm optimization: A literature review and numerical experiments,” the author’s purpose is to examine portfolio optimization models and applications of the PSO technique in solving these models. A constrained portfolio selection model has been developed, which is solved by the PSO technique as a metaheuristic approach using data from the Tehran Stock Exchange (TSE) as an emerging market to assess the developed model. In this case, the effects of three different risk measures of conditional value at risk (CVaR), variance, and semi-variance have been analyzed on the constructed portfolios. This analysis examines the PSO technique’s performance on the developed model utilizing the different risk measures from a portfolio metrics point of view, such as risk, return, and diversification index.

The ninth chapter, titled “Optimal portfolio selection with particle swarm algorithm: An application on BIST30,” aims to examine the optimum portfolio with minimum risk by using the PSO technique, for the stocks in the BIST-30 index. Logarithmic returns are calculated using the price data of the stocks. By using these returns, the optimum portfolio with minimum risk is created with PSO and nonlinear GRG (generalized reduced gradient) techniques. The empirical results obtained indicate that both methods give similar results.

In the tenth chapter, titled “Cardinality-constrained higher order moment portfolios using particle swarm optimization,” authors contribute in two ways to the literature on applying PSO to portfolio optimization. First, they go beyond the traditional mean-variance approach by optimizing the mean-variance-skewness-kurtosis approximation to the investor’s expected utility function. Second, they investigate improvement to the velocity equation in the special case of optimizing portfolios under a cardinality constraint. The improvement addresses the issue of early stagnation. The authors show the gain in a simulation experiment.

There are a large number of applications of PSO. The eleventh chapter, titled “Different applications of PSO,” tries to present a classified literature review for the applications of PSO in different fields. The applications are classified into different sections based on the area. The chapter also presents a table with references to multiple other applications over and above those covered in the chapter. References of some largely cited review papers dealing with the applications of PSO are also mentioned at the end of the chapter.

The twelfth chapter, titled “Particle swarm optimization in global path planning for swarm of robots,” provides a detailed insight of challenges encountered during the navigation of a collection of autonomous vehicles in a swarm environment, and provides a solution using the PSO algorithm for an optimized, collision-free path planning. The presence of obstacles and boundaries in the search space provides constraints on the positional variables required by objective function, making it a constrained optimization problem. The chapter explains various swarm intelligence algorithms and heuristic methods, gives detailed mathematical models and graphical explanations of PSO, specifically used for solving the problem, and finally shows its application with an open source software. This way the chapter also contributes by providing an example for solving constrained optimization problems similar to portfolio optimization. The chapter serves as a different application of PSO and

will be beneficial for researchers and students to further their research in the field of swarm intelligence.

In the thirteenth chapter, titled “Training multi-layer perceptron using hybridization of chaotic gravitational search algorithm and particle swarm optimization,” a novel hybridization strategy, namely chaotic gravitational search algorithm and particle swarm optimization (CGSAPSO), has been used to train a multilayer perceptron neural network for the classification task. The CGSAPSO utilizes the intensification power of PSO and the high exploration capability of CGSA to find the optimal regions of the solution space. The three classification datasets including XOR, Iris, and Balloon are employed for performance benchmarking. The experimental results clearly show that CGSAPSO provides better performance than peer algorithms. Actually, this chapter will be useful to the swarm intelligence researchers as they can apply CGSAPSO to other classification datasets including nowadays popular chest X-ray COVID-19 image dataset and other medical datasets. Besides, the hybridization power of CGSAPSO can be utilized for feature extraction and engineering optimization.

In the fourteenth chapter, titled “Solving hybrid flow shop scheduling problem with PSO: Solving hybrid flow shop scheduling problem with particle swarm optimization algorithm,” one version of hybrid flow shop scheduling problem that is frequently applied in real life is examined. The proposed problem includes dynamic job arrivals, setup times, and transportation times which are encountered in the real-life shop environment. It is in the NP-hard problem class and metaheuristics have been preferred for solving such problems. The results depict that PSO is highly effective for the proposed problem.

In the fifteenth chapter, “Constriction coefficient-based particle swarm optimization and gravitational search algorithm for image segmentation,” an image segmentation problem has been solved using a hybrid framework, namely construction coefficient-based particle swarm optimization and gravitational search algorithm (CPSOGSA). Kapur's entropy method is employed to find the optimal pixels. Besides, four standard benchmark images from the USC-SIPI image dataset are used for performance evaluation. The simulation results clearly convey that CPSOGSA provides optimal values for PSNR (peak to signal noise ratio) and mean and takes less computational time to find the feasible regions of the pixel space. This chapter will be beneficial and practically valuable for image processing professionals who want to utilize the efficiency and applicability of swarm intelligence algorithms like CPSOGSA in solving multilevel thresholding and image enhancement-related problems. Moreover, CPSOGSA can be employed for medical data analysis.

The sixteenth chapter, titled “An overview of the performance of PSO algorithm in renewable energies,” provides an expanded view of the uses of the PSO algorithm in the field of renewable energy systems which describes how the algorithm can be developed to cope with problems related to renewable energies to achieve desired goals. The PSO algorithm was used to solve many problems in the renewable energy systems, such as in optimal hybrid power systems, optimal sizing, and optimal net present cost. The renewable energy systems have several issues to discuss, such as

the cost of investment, the feasible technical criteria, optimal control, and ecological problems as well as the social effect. PSO algorithm uses are summarized around three main axes: economic, technical, and control, and it has proven best capabilities and high efficiency by reaching the optimum solutions with great convergence rate. Studies and research have proved that the PSO algorithm is one of the best algorithms used in the field of renewable energy. This is attributed to the algorithm's simplicity, high efficiency, and effectiveness compared to other algorithms and optimization methods.

Modern power systems have evolved in an increasingly highly complex system. The liberalization of the energy market and the introduction of distributed generation and, in particular, distributed renewable energy resources, have raised both opportunities and challenges that need to be tackled. Thus, complex issues related to the operation and planning of the distribution systems have emerged. Such issues involve many variables and refer to nonlinear objectives; thus, their optimization is significantly based on heuristic techniques, such as PSO. In the seventeenth chapter, "Application of PSO in distribution power systems: Operation and planning optimization," the implementation of PSO when contemplating various problems in power systems is presented, such as the optimal distributed generation placement, either alone or in conjunction with the optimal network configuration and the optimal schedule of electric vehicles. In this way, an apt example of the variety of problems for which PSO can be utilized and provide aid to important decisions in the field of power systems is provided.

Avcılar/Istanbul, Turkey

Burcu Adığüzel Mercangöz

Reference

- Tversky, A., & Kahneman, D. (1979). Prospect theory: An analysis of decision under risk. *Econometrica*, 47(2), 263–291.

Contents

Part I Applying Particle Swarm Optimization to Portfolio Optimization

1 Utility: Theories and Models	3
Murat Akkaya	
2 Portfolio Optimization	15
Burcu Adıgüzel Mercangöz	
3 Behavioral Portfolio Theory	29
Murat Akkaya	
4 A Comparative Study on PSO with Other Metaheuristic Methods	49
Serhat Yarat, Sibel Senan, and Zeynep Orman	
5 Mathematical Model of Particle Swarm Optimization: Numerical Optimization Problems	73
Ashwin A. Kadkol	
6 Particle Swarm Optimization: The Foundation	97
Dadabada Pradeep Kumar	
7 The PSO Family: Application to the Portfolio Optimization Problem	111
Lucas Fernández-Brillet, Oscar Álvarez, and Juan Luis Fernández-Martínez	
8 A Constrained Portfolio Selection Model Solved by Particle Swarm Optimization Under Different Risk Measures	133
Akbar Esfahanipour and Pouya Khodaee	

9 Optimal Portfolio Selection with Particle Swarm Algorithm: An Application on BIST-30	155
Burcu Adıgüzeli Mercangöz and Altaf Q. H. Badar	
10 Cardinality-Constrained Higher-Order Moment Portfolios Using Particle Swarm Optimization	169
Mulazim-Ali Khokhar, Kris Boudt, and Chunlin Wan	

Part II Different Applications of PSO

11 Different Applications of PSO	191
Altaf Q. H. Badar	
12 Particle Swarm Optimization in Global Path Planning for Swarm of Robots	209
Ritesh Kumar Halder	
13 Training Multi-layer Perceptron Using Hybridization of Chaotic Gravitational Search Algorithm and Particle Swarm Optimization	233
Sajad Ahmad Rather, P. Shanthi Bala, and Pillai Lekshmi Ashokan	
14 Solving Optimization Problem with Particle Swarm Optimization: Solving Hybrid Flow Shop Scheduling Problem with Particle Swarm Optimization Algorithm	263
Fatma Selen Madenoğlu	
15 Constriction Coefficient-Based Particle Swarm Optimization and Gravitational Search Algorithm for Image Segmentation	279
Sajad Ahmad Rather and P. Shanthi Bala	
16 An Overview of the Performance of PSO Algorithm in Renewable Energy Systems	307
Omar Hazem Mohammed and Mohammed Kharrich	
17 Application of PSO in Distribution Power Systems: Operation and Planning Optimization	321
Paschalis A. Gkaidatzis, Aggelos S. Bouhouras, and Dimitris P. Labridis	

Part 1

Applying Particle Swarm Optimization to Portfolio Optimization

Chapter 1

Utility: Theories and Models



Murat Akkaya

Abstract The aim of this study is to look at utility theory from a broad perspective. The main hypothesis in the theory of decision is that the person who is in the position of deciding is entitled to the “economic man.” Also, the individual acts rationally. Thus, utility is the ability to satisfy (eliminate) human needs of goods and services. Utility is basically a psychological concept and also is the basis of economics and finance. Three types of utility take place in the economics and finance literature: marginal utility, total utility, and average utility. In addition, two main approaches fall within utility comparison: cardinal utility theory and ordinal utility theory. Furthermore, expected utility theory forms the basis of traditional finance. Expected benefit theory assumes that people choose risky or uncertain opportunities by comparing the expected benefits from them. Allais and Ellsberg paradoxes criticize expected utility theory. Tversky and Kahneman (*Econometrica*, 47: 263–291, 1979) present that the expected utility axioms are violated for more reasonable lottery alternatives than in the Allais paradox and put a link between finance and psychology. The prospect theory of Tversky and Kahneman forms the basis of behavioral finance.

Keywords Utility · Utility theories · Expected utility · Behavioral finance

1.1 Introduction

The economic and financial system consists of individuals and institutions created by these individuals. Individual behaviors form the basis of this system. Human has a developed central nervous system. Individuals gather information about their natural/social environment and the conditions that make up this framework, through receptors and sensory nerves, and turn it into a decision or action that they think is

M. Akkaya (✉)
T.C. İstanbul Arel University, Istanbul, Turkey
e-mail: muratakkaya@arel.edu.tr

best for him in his brain. All of these actions are called “mind.” Man started to perceive the differences between the substances consumed and turned to enjoy this action during consumption process or get pleasure. Pleasure is a motivation that characterizes the utility that will be obtained from the energy consumption required for the survival of human life and motivates the organism in this process. The situation that is expressed with taste, fun, pleasure, pride, comfort, or such feelings is defined as benefit or utility. Thus, the word “utility” is used in two different meanings such as “usefulness” and “pleasure” in the history of economics.

Economists, like psychologists, are closely related to the behavior of individuals and require them to work together to uncover a model of behavior in the economic decision-making process and to reach some theoretical approaches. Human life passes by making decisions, and various decisions are made every day. Some of these are momentary and sincere and are the results of emotions. Some decisions are taken in the long term. The important factors in making decisions vary. When people make decisions, they try to choose the most appropriate among them by comparing the pros and cons of the options they can evaluate. The main hypothesis in the theory of decision is that the person who is in the position of deciding is entitled to the “economic man” feature. With this assumption, it is accepted that the person knows all the economic activities that can influence their own decisions in the economy, as well as the results. It is assumed that the individual also acts rationally, that is, he or she must keep maximization in her choices and decisions in order to maintain her standard of living.

Utility in finance means to generate a price for a financial asset called the indifference price, and utility functions are integrated to risk measures because risk determines returns, especially investor’s return maximization. Portfolio optimization is one of the financial applications of utility. Portfolio optimization is the process where an investor chooses the best portfolio that maximizes utility function or minimizes risk at investment at the same time.

1.2 Utility

Utility is the ability to satisfy (eliminate) human needs of goods and services. Utility is basically a psychological concept. The use of goods to meet the needs, on the other hand, reduces the severity of the need while also reducing the benefits provided from each unit. The severity of the commodity determines the utility, not the kind of need. In this case, the benefit will decrease as the amount of a good that people have increases because as the amount of that good increases, the severity of the need decreases. Also, utility is subjective. People consume the goods because they are beneficial. The utility of a good varies from individual to individual. For example, playing football very well for a football player may not be useful for someone who does not know how to play football. Even the benefit or utility of the same goods and services for the same person may be different at different times.

The term “utilita” is first defined by Galiani in 1750 as the capacity to form a felicity good. Bentham (1789) defines utility as the action in which goods give superiority, pleasure, goodness, or happiness to the beneficiary (Georgescu-Roegen, 1968). Also, Jevons (1871) defines utility as total of pleasure and prevented pain obtained in the use of an object. A rational choice occurs in hedonistic thought to balance pleasure and pain, or benefit and harm.

The following are the characteristics of utility¹:

- It does not possess ethical or moral significance.
- It is a psychological phenomenon.
- It is an individual and relative phenomenon.
- It is not the same with usefulness.
- It cannot be objectively measured.
- It depends on the intensity of desire.
- It differs from pleasure.
- It differs also from satisfaction.

Smith’s book, *The Wealth of Nations*, published in 1776, forms the necessary infrastructure to design benefit as an economic system based on human behavior. Smith (1937) defines utility as a motivation that leads people to action. A rational and consistent human protects his/her own benefits or utilities. Also, in his other book, *The Theory of Moral Values*, published in 1759 before *The Wealth of Nations* was published, Smith says that when nature shapes people for society, it has endowed it with a unique desire to please and dislike an original brother (brethren). Man is in a creation that seeks the happiness of others as well as himself. Man strives to maintain his own life. For this purpose, every decision has to take care of its own interests. This harmony carries its own internal contradiction. Thus, economic utilitarianism is devoted to philosophical utilitarianism because philosophical utilitarianism tries to find a set of rules that will ensure that individual actions are distinguished as true or false. Economic utilitarianism, on the other hand, tries to reach a social balance based on moral values and positive or negative emotions created by human action within the ethical system.

Individuals look for an option that will provide them the most benefit when making a choice; their purpose is to benefit maximization. There are six assumptions of the utility analysis put forward by Jevons, Menger (1871) and Walras (1874).

- Individuals are rational. The rational word in the economy is that people are in search of a combination of goods with their available resources and other limitations that will provide the highest level of satisfaction for them.
- The utility provided by the consumption of the goods can be measured. The theory of utility is also called *cardinal utility theory*. Here, the word “cardinal” is used to indicate that the utilities obtained from consumption can be explained with objective and quantitative measures.

¹<https://www.economicsdiscussion.net/utility/utility-meaning-characteristics-and-types-economics/13594>.

- The marginal utility of the last unit of a good consumed consecutively decreases as the amount of the good consumed increases.
- As the amount of good consumed increases, the total utility to a certain point also increases.
- Individuals' income is limited.
- Individuals are aware of the prices of all goods and services they can buy.

1.3 Types of Utility

Three types of utility take place in the economics and finance literature: marginal utility, total utility, and average utility.

1. *Marginal utility:* In the economics and finance literature, marginal utility is the benefit of the last unit consumed. Marginal utility shows the change that each additional unit of a good creates in the total benefit obtained from the previous consumed units. The word "marginal" means the last unit in mathematics. Utility is satisfaction, which eliminates the need.

The additional utility that the consumer or producer gets as he/she increases the amount of a product consumed consecutively is called marginal utility. Marginal utility is the difference that an additional unit of a good consumed creates in the total utility from the previously consumed unit. In summary, it is the benefit of the last consumed unit.

According to diminishing marginal utility, as the consumer consumes more than one product in a row, his satisfaction (pleasure from consumption after a certain period of time) will decrease. In other words, as the consumption amount of any commodity is increased, the pleasure gained as a result of the consumption of that commodity will gradually decrease.

2. *Total utility:* The utility obtained from all units of a good that a person consumes in a certain period is called total utility (TU). The total utility curve increases as the amount of consumption increases. However, when a certain point is reached, this increase stops and starts to decrease. This stems from the marginal utility law.
3. *Average utility:* Average utility refers to the utility in which the total unit of consumption of goods is divided by number of total units.

1.4 Functions of Utility

Two main approaches fall within utility comparison: cardinal unit theory and ordinal utility theory. Cardinal unit theory argues that utility is measurable. The ordinalist or ordinal utility theory suggests that the utility is an immeasurable phenomenon, and it would be sufficient for individuals to list the benefits they receive from consumption of goods.

1. *Cardinal utility theory*: Bentham (1789) assumes that the value defined as “utility” can be measured numerically (cardinal). According to Bentham, a standard utility scale that can be applied to all people can be developed.

Assumptions²:

- Individuals are rational. It aims to maximize its benefits under the income constraint.
- The utility of each good can be measured.
- The marginal utility of money is fixed. This assumption is necessary for money to become a measurement standard.
- As the people consume more than a good, marginal utility gradually decreases.
- The total utility of a basket of goods depends on the quantity of goods that makes up that basket: $U = f(x_1, \dots, x_n)$.

The mathematical acquisition of balance in the cardinal utility theory is as follows:

$$\text{Utility function : } U = f(x).$$

If the individual buys x as much as p_x unit price, he makes a total payment of (xP_x) . The purpose of the individual is to maximize the difference between the benefit received from the consumption of the good and the expenditure made.

$$\max U(x) - (xP_x).$$

For the maximum, the first-order partial derivative with respect to x is set to zero (first-order condition):

$$\begin{aligned} \frac{\partial U(x)}{\partial x} - \frac{\partial (xp_x)}{\partial x} &= 0 \rightarrow \frac{\partial U(x)}{\partial x} = p_x \\ \frac{\partial U(x)}{\partial x} &= p_x \quad \text{or} \quad U_x = p_x. \end{aligned}$$

The marginal utility of good x is MU_x or U_x .

2. *Ordinal utility theory*: Ordinal utility theory assumes that benefit is an immeasurable magnitude. The assumptions of the cardinal utility theory that are far from reality and based on coercion have been criticized by economists, and these economists have stated that the idea of numerical measurement of the concept of abstract benefit is unrealistic. Thereupon, a more appropriate approach, ordinal utility theory, has been developed.

Assumptions:

²https://abs.cu.edu.tr/Dokumanlar/2016/EAS223/833448774_mikro_iktisat_i.pdf.

- Individuals are rational. They try to maximize benefit when income and prices are data.
- Utility is ordinal. In other words, an individual puts the goods in a preference order according to the benefits (satisfaction) obtained from the goods consumed.
- Preferences are listed in terms of indifference curves, which are supposed to be convex by origin. Indifference curves have a negative and increasing slope. The negative sign of the indifference curve slope is called the marginal substitution rate. The theory of indifference curves is based on the reduced marginal substitution rate axiom.
- The benefit of the person depends on the amount of good consumed:

$$U = f(q_1, q_2, q_3, \dots, q_n).$$

One of the points that neoclassical economics emphasizes is the rationality principle borrowed from classical economics. This principle states that people are rational in measuring the benefits provided to them by different goods and in determining their needs in their pain and pleasure. This point of view is taken from Bentham's utilitarian philosophy. Simon (1947, 1957) is one of the first critics of the rational individual and the rational economic model with the proposition of limited rationality. Another researcher who emphasizes irrational behavior patterns is Allais (1953), a French economist. Allais argues that the decisions made cannot be linear and individuals act irrationally when choosing among possible alternatives, in situations where there is a lack of information and during evaluations that prevent stereotyped rational preferences. Langer (1975) states that irrational decisions emerge as a result of behavioral bias, which is illusion of control. This is an element that causes individuals to take high levels of risk. The framework describing the uncertainty and risk in economics and finance is the expected utility theory.

1.5 Expected Utility Theory

Expected utility theory, first formulated by Bernoulli (1954) and developed by John Von Neumann and Oskar Morgenstern in *Theory of Games and Economic Behavior*, forms the basis of traditional finance. According to this theory, human is a rational being. A rational person or an economic person (*homo economicus*) refers to a hypothetical person acting in his own interest, purposed to maximize the benefits while making his decisions, free from emotions. According to the expected benefit theory, people choose risky or uncertain opportunities by comparing the expected benefits from them (Kiyilar & Akkaya, 2016).

The basis of the theory of Von Neumann and Morgenstern is the maximization of the expected utility. The person with 100 USD has the chance to win 10%, and the expected utility is $0.10 \times 100 \text{ USD} = 10 \text{ USD}$. Von Neumann and Morgenstern

have applied this to optimal decision-making in the case of uncertainty. In the options of earning 100% with a rate of 25% and 1.000% with a rate of 10%, a rational individual will choose the second option ($0.25 \times 100 = 25$ USD, while $0.10 \times 1.000 = 100$ USD).

The event that enables Daniel Bernoulli (1700–1782), who first introduced the concept of expected benefit, to develop this concept is the St. Petersburg paradox. The St. Petersburg paradox is a problem that questions how much participation fee must be paid to participate in the game played with a coin. Daniel Bernoulli argues that people act to maximize expected utility rather than expected income. By suggesting a logarithmic function of the expected utility of the paradox, he proposes that the utility increase, which would follow an equal increase in income, would decrease. However, it could not provide a rational study on how to measure benefit (Schoemaker, 1982).

Von Neumann and Morgenstern (1944) assume that individuals have made a decision to purchase, taking into account the future utilities they will derive from the consumption of goods or services. Moreover, this utility has a predictable quality only according to certain possibilities. Therefore, utility has been replaced with the expected utility function, arguing that the utility function under risk status is not sufficient to explain consumption decisions. This work has taken its place in the literature as an important step in economics as a pioneer of modern demand theories based on the expected utility function, which replaces the traditional demand theories based on the utility function.

The expected utility theory has four goals (Bailey, 2002):

- Expected utility theory can be used to model the decision process that includes the risky selection.
- Expected utility theory is seen as predictive and positivist in economics and finance.
- This theory is optimal provided that the observed human behavior is modeled under appropriate conditions.
- Expected utility theory is a normative model and assumes that human behavior is generally semi-optimal.

Von Neumann and Morgenstern's anticipated utility theory has four basic principles (Hanson & Kysar, 1999):

- *Ordering*: An individual who has to choose between two alternatives may have to choose one or be uninterested in alternatives. All preferences are transitive. If a person prefers A over B and prefers B over C, he will choose A over C as well.
- *Continuity*: Each B preference is not different according to A and C preferences.
- *Independence*: An individual's preference will not change even when these two things are replaced by a significant game of chance. If someone prefers A over B, the probability of winning A with a 50% probability will prefer the probability of winning B with a 50% probability.

- *Invariance:* Different presentations of the same decision problem will result in the same choice. No matter how the problem is presented, the decision will not change.

Von Neumann and Morgenstern (1944) define the expected utility as the value found by multiplying a possible utility from a decision or event by the probability of the event. It is assumed that there is a utility function (U) consisting of the results (x) that each individual will achieve in the formulation. Let us assume that the probability of an action that will result in x is p and the probability of action b that will lead to the same result is q (Kiyilar & Akkaya, 2016).

$$p \cdot U(x) > q \cdot U(x) \text{ ise,}$$

that is, if the expected utility of action a is greater than the expected utility of action b , the decision maker will definitely prefer action a . In this case, it is assumed that decision makers know the probability of different events and take the decision that maximizes its utility.

The total expected utility to be obtained if the basket of goods or services is selected is calculated according to the following formula. The individual in the risk environment prefers the alternative with the highest expected utility calculated from this formula from the baskets of goods or services (m).

$$\phi(y)^i = \sum_{k=1}^n P\left(\frac{y^i}{k}\right) \phi\left(\frac{y^i}{k}\right) \quad i = 1, 2, \dots, m$$

The expected utility approach of Von Neumann and Morgenstern is different from Daniel Bernoulli's because Von Neumann and Morgenstern prove for the first time by rationalizing a rational choice based on the expected utility maximization. The expected utility theory assumptions are as follows.

- When people encounter a situation of uncertainty, using the Bayes theorem, they determine the “objective possibility” of the realization of this situation. While doing this process, they do not show bias about any option.
- Much is better than less. If A benefits more than B, the decision maker will definitely choose A over B.
- The decisions taken are consistent. If A benefits more than B, and B benefits more than C, the decision maker will prefer A if he/she chooses between A and C.
- After determining the probabilities of the uncertain events faced by people when making a decision and calculating a row within the utility function, the ultimate goal of the decision maker is to maximize his/her utility, and for this, he or she chooses the preference that provides that purpose in the options.
- This utility function is dish shaped. In other words, it shows that the “decreasing marginal utility rule” is valid.

The axioms of expected benefit maximization are:

- *Completeness*: If X and Y are two commodity baskets, X is at least as good as Y, or Y is as good as X, or both are valid.
- *Transitivity*: If X and Y are two baskets of goods, if X is at least as good as Y and Y is at least as good as Z, then X is at least as good as Z.
- *Independence*: Suppose that X, Y, and Z are three lots. If we confuse the two lots with the third, the order of preference of these two quotes is not dependent on the third and is independent of it.
- *Continuity*: Preferences are continuous.

The axioms above are necessary and sufficient to verify that the preferences listed according to their expected utility match exactly with the real preferences of individuals (Schoemaker, 1982) and expected utility maximization and decision-making approaches reached their peak with the work of Leonard Savage.

The expected utility theory is concerned with the issue of decision-making at risk. This theory is considered as the normative model in rational decision-making. Making decisions at risk can be seen as a choice between expectations and gambling. The individual prefers the option that gives him the highest benefit. This theory has three basic principles (Tversky and Kahneman 1979):

- *Expectation*: $U(x_1p_1, \dots, x_np_n) = p_1 u(x_1) + \dots + p_n u(x_n)$. Accordingly, the total utility (U) of an option is equal to the expected utility of the results.
- *Asset integration*: Expectations are accepted if the utility of the individual is more than the utility of the asset before the acceptance of this option, in the event that it is combined with the assets previously owned. So the basis of the utility function is the ultimate state of the individual rather than gains or losses.
- *Risk avoidance*: Individuals will prefer options with certain outcomes rather than risky options. According to this theory, the risk aversion is equal to the concave state of the utility function.

Expected utility theory states that individuals try to maximize the expected benefit of choices among risky options, and for this purpose, they compare the benefits of each result with their probabilities and prefer the highest weighted option. But the basis of the objections to the expected utility theory lies in the fact that the observed human behavior is different from the theory or assumptions. The first important criticism of the theory is brought by Allais (1953), who received the Nobel Prize in economics in 1988. The criticisms known as the “Allais paradox” reveal that individuals weigh the expected results of the lotteries and the possibilities related to these results. The utilities obtained in the expected utility theory are found by weighting with probabilities. People weigh more on realization than actual results, which is called certainty effect. This is a phenomenon opposite to the principle of independence in the expected utility theory. Ellsberg (1961) and many researchers have revealed with empirical studies that the expected utility axioms are systematically violated. In the Ellsberg paradox, selection is made between a known situation and an unknown situation. The Allais paradox has been ignored for a long time, suggesting that it is just an extreme example. However, Tversky and Kahneman (1979) show that the expected utility axioms were violated for more reasonable

lottery alternatives than in the Allais paradox. This article of Kahneman and Tversky, published in *Econometrica* in 1979, is accepted in the literature as the beginning of behavioral finance. Kahneman and Tversky's work in the field of judgment and decision-making started a new era in finance. These studies form a link between finance and psychology. The prospect theory of Kahneman and Tversky forms the basis of behavioral finance.

1.6 Conclusion

In traditional economics, utility is divided into two: ordinal utility, which cannot be measured, and cardinal utility, which can be measured. Economists (cardinal beneficiaries) who argue that the benefit can be measured accept that every good or group of goods can be measured in a certain utility unit. Cardinal utility theory is based on three basic concepts, namely, marginal utility, total utility, and diminishing marginal utility. The utility maximization assumption is based on Bentham (1789). In the original use of Bentham, utility is expressed as the experience of pain and pleasure and emphasizes what to do besides what we will do.

The expected utility theory is also widely used in modeling preferences related to risky alternatives in many areas where economic, financial, and uncertain decision processes are examined. However, this theory is far from unproblematic. Findings from experimental studies in economics have shown that the axioms of the expected utility theory are violated systematically in real life. In other words, this theory cannot adequately explain people's decision-making processes.

Utility has a hidden feature, and this concept finds its place as the utility experienced in the work of Daniel Kahneman in the field of behavioral economics. Although the concept of utility has changed since Bentham's era, it is used by many economists as the concept of decision utility. Future studies should be on the hidden side of utility.

Key Terms and Definitions

Utility: Utility is the ability to satisfy (eliminate) human needs of goods and services. The term "utilita" is first defined by Galiani in 1750 as the capacity to form a felicity good. Bentham (1789) defines utility as the action in which goods give superiority, pleasure, goodness, or happiness to the beneficiary (Georgescu-Roegen, 1968). Also, Jevons (1871) defines utility as total of pleasure and prevented pain obtained in the use of an object.

Cardinal utility: Bentham (1789) assumes that the value defined as "utility" can be measured numerically (cardinal). According to Bentham, a standard utility scale that can be applied to all people can be developed.

Ordinal utility: Ordinal utility theory assumes that benefit is an immeasurable magnitude.

Expected utility: The expected utility is the result obtained by multiplying the potential utility, which is the result of a decision under uncertainty, by the probability of the event taking place.

Expected utility theory: All individuals who follow a rational utility optimization process calculate the probability of occurrence for each event in the case of uncertainty by Bayesian methods. The expected utility is maximized by multiplying the expected probabilities from the events with the calculated probabilities.

References

- Allais, M. (1953). Le comportement de l'homme rationnel devant le risque: Critique des postulats et axiomes de l'école américaine. *Econometrica: Journal of the Econometric Society*, 1953, 503–546.
- Bailey, R. E. (2002). *Economics of financial markets* (pp. 15–20). Essex, UK: Department of Economics University of Essex.
- Bernoulli, D. (1954). Exposition of a new theory on the measurement of risk (Comentarii Academiae Scientiarum Imperialis Petropolitanae). *Econometrica*, 22, 22–36.
- Ellsberg, D. (1961). Risk, ambiguity, and the Savage axioms. *The Quarterly Journal of Economics*, 1961, 643–669.
- Georgescu-Roegen, N. (1968). Utility. *International Encyclopedia of the Social Sciences*, 16(1), 236–267.
- Hanson, J. D., & Kysar, D. A. (1999). Taking behavioralism seriously: The problem of market manipulation. *NYUL Rev*, 74, 630.
- Jevons, W. S. (1871). *The theory of political economy*. London: Macmillan.
- Kıylar, M., & Akkaya, M. (2016). *Davranışsal finans*. İstanbul: Literatür Yayıncılık.
- Langer, E. J. (1975). The illusion of control. *Journal of Personality and Social Psychology*, 32(2), 311.
- Menger, C. (1871). *1981. Principles of economics*. New York: New York University Press.
- Schoemaker, P. J. (1982). The expected utility model: Its variants, purposes, evidence and limitations. *Journal of Economic Literature*, 529–563.
- Simon, H. A. (1947). *Administrative behavior, a study of decision-making processes in administrative organization*. New York: The Macmillan Co.
- Simon, H. A. (1957). *Models of man: Social and rational*. New York: John Wiley and Sons, Inc..
- Smith, A. (1937). *The wealth of nations* [1776].
- Tversky, A., & Kahneman, D. (1979). Prospect theory: An analysis of decision under risk. *Econometrica*, 47(2), 263–291.
- Von Neumann, J., & Morgenstern, O. (1944). *Theory of games and economic behavior (commemorative edition)*. Princeton: Princeton University Press.
- Walras, L. (1874). *1954. Elements of pure economics*. London: Allen and Unwin.

Additional Reading

- Barberis, N., & Xiong, W. (2012). Realization utility. *Journal of Financial Economics*, 104(2), 251–271.
- Broome, J. (1991). Utility. *Economics and Philosophy*, 7(1), 1–12.
- Ellsberg, D. (1954). Classic and current notions of “measurable utility”. *The Economic Journal*, 64 (255), 528–556.

- Fishburn, P. C. (1968). Utility theory. *Management Science*, 14(5), 335–378.
- Fishburn, P. C. (1970). *Utility theory for decision making* (No. RAC-R-105). McLean, VA: Research Analysis Corp.
- Kahneman, D., & Snell, J. (1990). *Predicting utility*.
- Kahneman, D., & Thaler, R. H. (2006). Anomalies: Utility maximization and experienced utility. *Journal of Economic Perspectives*, 20(1), 221–234.
- Layard, R., Mayraz, G., & Nickell, S. (2008). The marginal utility of income. *Journal of Public Economics*, 92(8–9), 1846–1857.
- MacCrimmon, K. R., & Larsson, S. (1979). Utility theory: Axioms versus ‘paradoxes’. In *Expected utility hypotheses and the Allais paradox* (pp. 333–409). Dordrecht: Springer.
- Sen, A. (1991). Utility: Ideas and terminology. *Economics and Philosophy*, 7(2), 277–283.

Chapter 2

Portfolio Optimization



Burcu Adıgüzel Mercangöz

Abstract In portfolio management, it is aimed to create a portfolio that gives the best combination of risk and return among the assets in the market. There are different optimization techniques for creating an optimum portfolio depending on the risk and return variable. Particle swarm optimization (PSO) method is one of the important and useful techniques used in portfolio optimization in finance. In this chapter, Markowitz mean-variance model, which is the main model of modern portfolio theory, is explained, and mathematical representations are given. The subject is supported with mathematical notations by mentioning concepts such as portfolio risk and return, efficient frontier, utility theory, asset allocation, indifference curves, Sharpe ratio, and coefficient of variation.

Keywords Markowitz mean-variance model · Efficient frontier · Sharpe ratio · Coefficient of variation · Utility theory · Asset allocation · Indifference curves

2.1 Basic Concepts: Risk and Return

The purpose of portfolio management is to form a portfolio from different assets, and this portfolio gives the highest possible return at a given risk level. Portfolio managers try to form portfolios that allocate funds optimally to stocks, bonds, and other assets, so the portfolio of which is well diversified and provides the highest rate of return at the lowest possible level of risk or the lowest level of risk at the highest possible rate of return. The concept of risk and return is crucially important for decision-making in finance.

Since the main subject of this book is portfolio optimization with particle swarm optimization (PSO) technique, in the first chapter of the book, the mathematical models used in solving the portfolio optimization problem are discussed. The basics

B. A. Mercangöz (✉)
Istanbul University, Istanbul, Turkey
e-mail: burcua@istanbul.edu.tr

of Markowitz modern portfolio theory are explained as well as how effective portfolios are selected with the help of utility theory and indifference curves.

2.2 Risk and Return

The assets in a portfolio can provide two types of returns. The first one is the capital gain obtained by changing the price of the financial instrument in the market. In other words, capital gain/loss arises by the increase/decrease of an asset price. The second is the interest provided by fixed income instruments. In addition, dividends can be given by stocks.

Risk is the probability of the actual return of an investment to differ from the expected return. In other words, the risk represents potential negative impacts on securities.

When one measures a rate of return of an asset after it was earned, it means a historical return is measured. However, investors are more interested in the return that an investment is expected to earn in the future. The expected return is the return to be obtained from any asset after a certain period of time in a risky world. When it is mentioned about the expected return, this is a return on a risky asset expected in the future. Expected return can be measured using both ex post and ex ante variables. If the assumption is that past returns are the best estimates for the future return expectations, the expected rate of return may be measured by historical returns. It is called ex post returns, and the expected return $E(r)$ is calculated by taking the arithmetic average of a series of historical returns as below:

$$E(r) = \frac{\sum_{i=1}^n (r_i)}{n} \quad \text{Ex post return calculation} \quad (2.1)$$

The historical returns can be also calculated as logarithmic.

$$E(r_t) = \ln(p_t/p_{t-1}) \quad \text{Logarithmic historical return} \quad (2.2)$$

where p_t is the price of the asset at time t .

Besides, expected returns can be based on the possibilities of potential incomes/outcomes. The likelihood of certain outcomes determines the expected returns. This is calculated as follows:

$$E(r) = \sum_i r_i * p_i \quad \text{Ex ante return calculation} \quad (2.3)$$

r_i : The return in each outcome

p_i : Possibility of each outcome

Risk is defined as a deviation from expectations and mathematically measured by the standard deviation. However, there are different measurement methods of risk. For instance, lower partial moment, mean absolute deviation measures, range, and semi-variance are also used for risk measurements.

The variance of a variable is the expectation of the sum of the squared deviations from its mean; also, the square root of the variance represents the standard deviation of the variable. As with the return measurement, two data can be taken as a base for risk measurement: ex post and ex ante variables.

$$\text{Var}(r_i) = \sigma_i^2 = \frac{\sum [r_i - \bar{r}_i]^2}{n - 1} \quad \text{Ex post risk calculation} \quad (2.4)$$

$$\text{Var}(r_i) = \sigma_i^2 = [r_i - E(r_i)]^2 * p_i \quad \text{Ex ante risk calculation} \quad (2.5)$$

$$\text{Std}(r_i) = \sigma_i = [\text{Var}(r_i)]^{1/2} \quad \text{Standard deviation} \quad (2.6)$$

2.3 Markowitz Mean-Variance Portfolio Theory and Diversification

In traditional portfolio management, it is predicted that the risk level can be reduced by only increasing the portfolio size, regardless of the relationship between the returns of different assets in the portfolio. On the other hand, the Markowitz mean-variance model imposes that only increasing portfolio asset number is not enough to reduce the portfolio risk level; the direction and the degree of the relationship between the assets are also important (Markowitz, 1952). According to the Markowitz portfolio, by adding new assets to a portfolio, investors can reduce risk, but the correlation coefficients of assets must be lower than 1. However, the portfolio's expected rate of return is always estimated as the weighted average of the expected rate of returns of each asset. In that way, Markowitz is the first who shows that diversification reduces the variance of a portfolio (Adiguzel Mercangöz & Eroğlu, 2019; Adiguzel Mercangoz, 2019).

The model has some assumptions. It is assumed that investors in financial markets are risk-averse. The risk-averse investors are investors that choose the less risky choice when they are given two assets that have the same expected return. They are rational and homogeneous expectations. A rational investor prefers a portfolio that has better expected return between alternatives at a given risk level. (Chen et al., 2010)

According to the modern portfolio theory, portfolio return is the weighted average of all expected rate of returns, and the general formula of the expected return of a portfolio $E(r_p)$ for n assets is the following:

$$E(r_P) = \sum_{i=1}^n w_i * E(r_i) \quad \text{Portfolio return} \quad (2.7)$$

where:

$$\sum_{i=1}^n w_i = 1.0$$

n = the number of assets in a portfolio

w_i = the weight of each assets

r_i, r_P = the return on asset i and the return on portfolio p

$E()$ = expectation of a portfolio return or asset return

The variance of a portfolio, portfolio risk, is estimated by multiplying the weights and covariances of all assets in the portfolio:

$$\text{Var}(r_p) = \sigma_p^2 = \sum_{i=1}^n \sum_{j=1}^n w_i * w_j * \text{Cov}(r_i, r_j) \quad \text{Portfolio variance} \quad (2.8)$$

The calculation of a variance of a portfolio includes covariance term. Covariance (Cov) is the measure of how much two variables change together. In other words, it is a statistic that examines the changes of two random variables together. It is one of the important concepts in portfolio theory.

The covariance is estimated by taking the average of the cross product of their deviations.

$$\text{Cov}(r_i, r_j) = \frac{\sum_{i=1}^n (r_i - E(r_i)) * (r_j - E(r_j))}{n - 1} \quad \text{Portfolio covariance} \quad (2.9)$$

Covariance is also calculated by correlation coefficient:

$$\text{Cov}(r_i, r_j) = \rho_{ij} * \sigma_i * \sigma_j = \sigma_{ij} \quad \text{Portfolio covariance} \quad (2.10)$$

ρ_{ij} represents the correlation coefficient between returns of two assets: asset i and asset j .

The correlation coefficient ranges between +1 and -1. The sign of correlation coefficients represents the direction of the movements of two asset returns at a given moment. The sign of the correlation coefficient is more important than the magnitude of it. High positive sign shows that asset returns are moved in the same direction; that is, an increase/decrease in one return of an asset leads to an increase/decrease in the other asset return. Oppositely, a negative sign indicates that asset returns are moved in different direction. An increase/decrease in return of an asset leads to a decrease/increase in the other asset return (Adıgüzel Mercangöz & Eroğlu, 2019).

$$\text{Var}(r_p) = \sum_{i=1}^n \sum_{j=1}^n w_i * w_j * \rho_{ij} * \sigma_i * \sigma_j \quad \text{Portfolio covariance} \quad (2.11)$$

2.4 The Markowitz Efficient Frontier

According to the Markowitz mean-variance model, a single optimum portfolio is not determined. Rather, in this model, a set of effective portfolios that are optimum at a certain level of return and risk is determined. The reason that the total risk of a portfolio is not equal to the average of the risks of all assets in the portfolio is the covariance resulting from the different reactions of the assets in the portfolio.

Therefore, the “number of assets in the portfolio,” which is expressed as the basic factor determining the portfolio risk in the traditional portfolio theory before Markowitz, has been replaced with the “the covariance between returns of assets in the portfolio” in the modern portfolio theory.

Regarding the modern portfolio theory, as long as the correlation coefficient between two assets is less than 1, the standard deviation of a portfolio being formed of these two assets is less than the weighted average of the standard deviations of two assets.

In Markowitz portfolio theory, firstly, “effective portfolios” with higher expected return rates than others are determined at each risk level, and all investable assets are determined for a rational investor, and then the selection among these assets is determined by risk-expected return preferences (risk attitude). Given a certain level of risk, the entire set of portfolios whose expected return reaches its maximum value is called “the Markowitz efficient frontier.”

The main assumption of Markowitz is that investors are risk-averse. Therefore, they choose the portfolio that yields the highest expected returns at a given level of risk. This means that investors choose a portfolio that is on the efficient frontier. Where they choose a portfolio on the efficient frontier depends entirely on their risk attitudes.

Efficient frontier represents portfolios that give the highest expected rate of return at a certain risk level. When it is plotted risk and returns of each possible asset's portfolio combinations in the risk-return graph it can be gotten Fig. 2.1. The portfolio set that investors can invest in, consisting of all possible combinations of assets in the market, is known as opportunity set (feasibility set). The line along the upper edge of the opportunity set is called “efficient frontier.” This line indicates the portfolio set that gives the highest expected rate of return for each portfolio risk level. According to the theory, people choose portfolios at the efficient frontier because they are risk-averse. People have homogeneous expectations.

MVP refers to the “minimum variance portfolio,” which means that there are not any other portfolio that has a lower risk at this point. The efficient frontier starts at the MVP point and continues to give the set of portfolios with maximum return. Portfolios that are under the MVP point in the figure are inefficient; therefore, they have to be rejected out of hand.

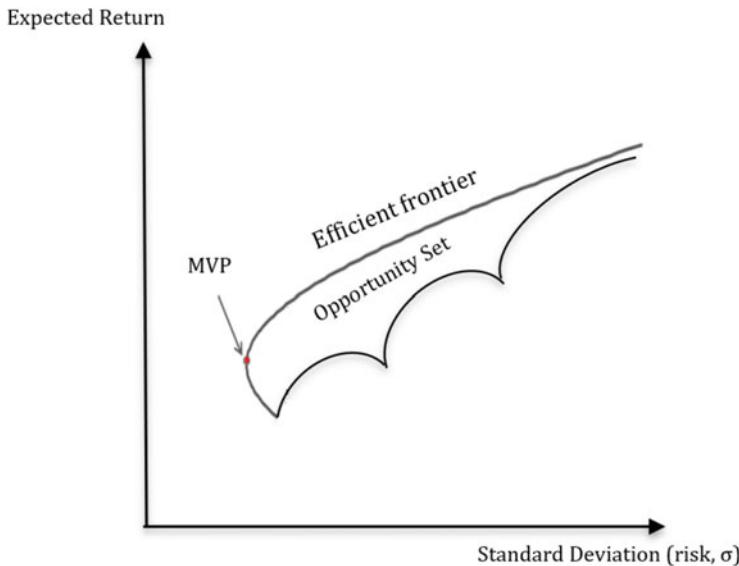


Fig. 2.1 Opportunity set and efficient frontier. Source: Drawn by the author

2.5 Different Types of Risks

According to the modern portfolio theory, risk can be decreased by diversification. However, only nonsystematic risk can be reduced by portfolio diversification. Investors face two groups of risks: systematic and nonsystematic risk. Unsystematic risk (unique risk) is an existing risk associated with a specific financial asset. Financial risk, operational risk, management risk, and sector risk are in the nonsystematic risk group and can be reduced by diversification.

If systematic risk (market risk) is a negative situation that may arise in economic, social, or political conditions, portfolio returns decrease since all assets in the market may be affected negatively. Therefore, it is not possible to eliminate market-specific risks that are not related to the asset. Inflation risk, currency risk, and interest rate risk are some examples of systematic risk.

When the number of assets in a portfolio is increased, only the diversifiable risk can be eliminated. Therefore, there should be only left market risk in a well-diversified portfolio.

While the asset numbers are increasing in a portfolio, their variances become completely insignificant. Only the covariance part remains. Thus, the portfolio variance becomes only the average covariance. This concept can be explained by covariance matrix in Table 2.1.

In this variance-covariance matrix, there will be:

Table 2.1 The number of “ n ” assets variance-covariance matrix

	1	2	3	...	n
1	$x_1^2 \sigma_1^2$	$x_1 x_2 \text{Cou}_{12}$	$x_1 x_3 \text{Cou}_{13}$...	$x_1 x_n \text{Cou}_{1n}$
2	$x_1 x_2 \text{Cou}_{12}$	$x_2^2 \sigma_2^2$	$x_2 x_3 \text{Cou}_{23}$...	
3	$x_1 x_3 \text{Cou}_{13}$	$x_2 x_3 \text{Cou}_{23}$	$x_3^2 \sigma_3^2$...	
...				...	
...				...	
...				...	
n	$x_1 x_n \text{Cou}_{1n}$				$x_n^2 \sigma_n^2$

$$n^2 - n = X_n * X_n * \text{Cou}_{nn}$$

$$N = X_n^{2*} \sigma_n^2$$

Then variance of portfolio will be:

$$\begin{aligned} & \left(\frac{1}{n}\right)^2 (n^2 - n) \text{Cou}_{nn} + \left(\frac{1}{n}\right)^2 n \sigma_n^2 \\ & \left(1 - \frac{1}{n}\right) \text{Cou}_{nn} + \left(\frac{1}{n}\right) \sigma_n^2 \end{aligned}$$

When n goes to infinity (∞), $1/n$ will be zero; then the above formula will be only Cou_{nn} . This means variance of portfolio will be just covariance when n goes to infinity.

While the asset numbers in a portfolio go toward infinity, the portfolio variance will consist of only covariance.

$$\text{Variance of portfolio} = \text{Covariance} \quad \text{when } \left(\begin{array}{l} n \rightarrow \infty \\ \text{goes} \end{array} \right)$$

The consequences of this inference are important. When the number of assets in a portfolio is increased, their variances become completely insignificant. Only the covariance (Cou) part remains. Thus, the portfolio variance becomes the average covariance ($\overline{\text{Cou}}$) (Fig. 2.2).

Total risk of individual asset = Portfolio risk + Diversifiable risk

$$\overline{\text{Var}} = \overline{\text{Cou}} + (\overline{\text{Var}} - \overline{\text{Cou}})$$

In Markowitz portfolio theory, investors are assumed as risk-averse and have homogeneous expectations. Therefore, they create portfolio depending on both expected rate of return and the standard deviation of return as a risk measure.

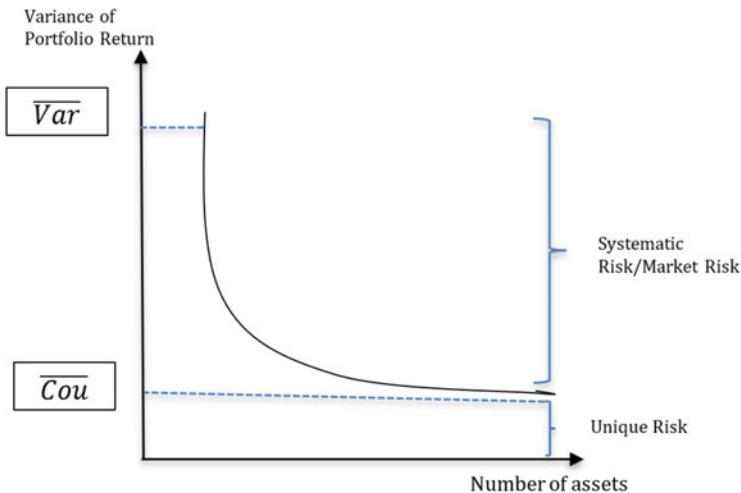


Fig. 2.2 Portfolio risk and number of asset relation. Source: Drawn by the author

Based on assumptions, the problem of portfolio selection is described as minimizing the standard deviation of the investment (portfolio risk) concerning a given portfolio return or maximizing the return concerning the standard deviation of the investment (portfolio risk). This means that the optimal portfolio is obtained by holding portfolio return constant and solving for the proportion elements (w_i) that minimize the standard deviation of the portfolio or, alternatively, holding portfolio standard deviation constant and solving for the proportion elements that maximize the expected return.

When portfolios contain “ n ” assets, the objection function and the constraints will be as follows (Kendal & Yan, 2005):

$$\text{Min } \sigma_p^2 = \sum_{i=1}^n \sum_{j=1}^n w_i * w_j * \rho_{ij} \sigma_i \sigma_j \quad \text{Minimizing portfolio variance} \quad (2.12)$$

The variables in the equation are previously defined in Eq. (2.10).

Subject to $w_1 + w_2 + \dots + w_n = 1$

$$\sum_i^n W_i = 1$$

$$0 \leq W_i \leq 1 \quad i = 1, 2, \dots, n$$

By using this minimization equation, the minimum portfolio variance is estimated for any certain level of portfolio return. While solving this minimization formula at a given point of return level, it should be subjected to the constraint that the sum of the weights to one.

One of the criteria used in selecting the portfolio with different expected returns and risk levels in portfolio management is the coefficient of variation. The coefficient of variation (CV) is defined as the risk undertaken for each unit return. The CV is estimated by dividing the standard deviation (risk) by return. A financial asset or a portfolio with a low coefficient of variation is preferred over another. The formula is shown as follows:

$$(\text{Minimization}) \quad \text{Coefficient of variation (CV)} = \frac{\sigma_p}{E(r_p)}.$$

Similar to the coefficient of variation, another criterion used in the selection of assets or portfolio is the Sharpe ratio. It is also called “reward to volatility.” The Sharpe ratio is calculated by dividing the expected return over the risk-free interest rate by the risk. An investor who is assumed to be indifferent to risk prefers the asset with the highest Sharpe ratio among different investment alternatives. The Sharpe ratio is calculated using the following formula:

$$(\text{Maximization}) \quad \text{Sharpe ratio (Reward to volatility)} = \frac{E(r_p) - r_f}{\sigma_p}$$

$$r_f = \text{risk - free rate of return}$$

After determining the most effective portfolios among all the assets available in the capital market, revealing the expected rate of return-risk preferences derived from the specific utility functions of the investors with the indifference curves answers the question of which portfolio is the optimal portfolio for each investor.

Therefore, the portfolio optimization problem can also be solved by minimizing the coefficient of variation or maximizing the Sharpe ratio.

2.6 Asset Allocation

Investors who have different risk attitudes will take different positions in a given opportunity set. An investor who avoids risk will take positions at lower points on the X-axis. That is, it will hold the portfolio with a lower standard deviation and also a lower expected return. An investor who is a risk-taker (risk lover) will take positions at higher points on the X-axis. Regardless of what kind of investors they are, they want to take positions with highest utility. Based on a specific risk level, the point that gives the maximum utility will be on the efficient frontier.

The expected utility of a portfolio depends on its expected return and risk level (Bodie, Kane, & Marcus, 2010).

$$U = E(r) - 0.0005 * A * \sigma^2$$

A: Coefficient of risk aversion

A: 0 => Risk neutral investors

Increasing A means higher level of risk aversion.

Utility is inversely proportional to risk but directly proportional to expected return of an asset:

$$E(r_c) = r_f + y * [E(r_p) - r_f]$$

$$\sigma_c^2 = y^2 * \sigma_p^2$$

If relevant variables are put into the utility formula:

$$U = E(r) - 0.0005 * A * \sigma^2,$$

this utility formula will be as follows:

$$(\max y) U = r_f + y * [E(r_p) - r_f] - 0.005 * A * y^2 * \sigma_p^2$$

According to this formula, there is a risky asset ratio that makes the maximum utility according to A (according to a certain risk attitude.)

For instance, at a given data of 7% risk-free rate, 22% standard deviation, and 15% expected return, one solves the formula for the $A = 4$; one can get $y = 0.41$ which shows the proportion of the risky assets that maximize the utility (Fig. 2.3).

The maximization problem is solved by taking the derivative of the statement as zero. Doing this and solving for y provides the optimal position in opportunity set for the risk-averse investors between all risky assets.

$$y^* = \frac{E(r_p) - r_f}{0.001 * A * \sigma_p^2}$$

This shows that the optimal risky portfolio is directly proportional to the risk premium offered by the risky asset and inversely proportional to the risk level.

$$y^* = \frac{\%15 - \%7}{0.001 * 4 * 22^2} = 0,41$$

$$E(r_c) = 7\% + 0.41 * [15\% - 7\%] = 10.28\%$$

$$\sigma_c = 0.41 * 22\% = 9.02\%$$

Risk premium for complete part => $E(r_c) - r_f = 10.28\% - 7\% = 3.28\%$

Reward to variability ratio => $\frac{3.28\%}{9.02\%} = 0.36$

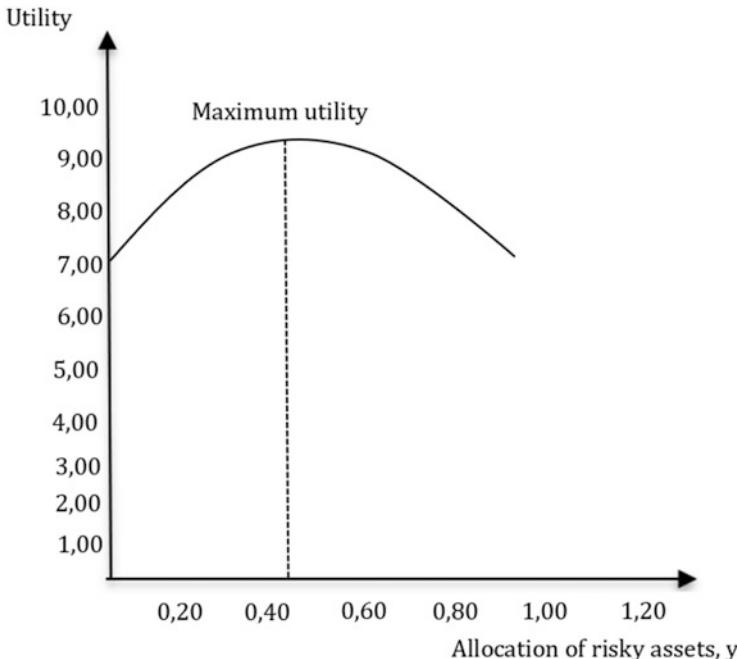


Fig. 2.3 Utility function. Source: Redrawn by the author (Bodie et al., 2010)

The utility formula allows us to draw the indifference curves. When it is solved the formula continuously for different risk levels, estimated expected returns will provide the same benefit. When all these combinations are given in the chart, indifference curves will appear (Fig. 2.4).

The risk-taker (risk lover) investor who is more willing to take a risk has shallower indifference curves than the risk-averse investor who is not so willing to take a risk. Shallower curves mean that the investor takes a high risk and so expects a high return to compensate for an increase in portfolio risk.

The point where the indifference curve touches the efficient frontier will give the effective portfolio that maximizes the utility.

2.7 Conclusion

The global financial markets offer many investment instruments options around the world for those who want to invest. It is crucially important for investors to create optimal portfolios between huge numbers of financial assets. Therefore, creating optimal portfolios using various methods has been studied for many years in finance. The selection of assets to be included in the portfolio and the proportion of these assets to be kept are a matter of portfolio management. A portfolio manager should

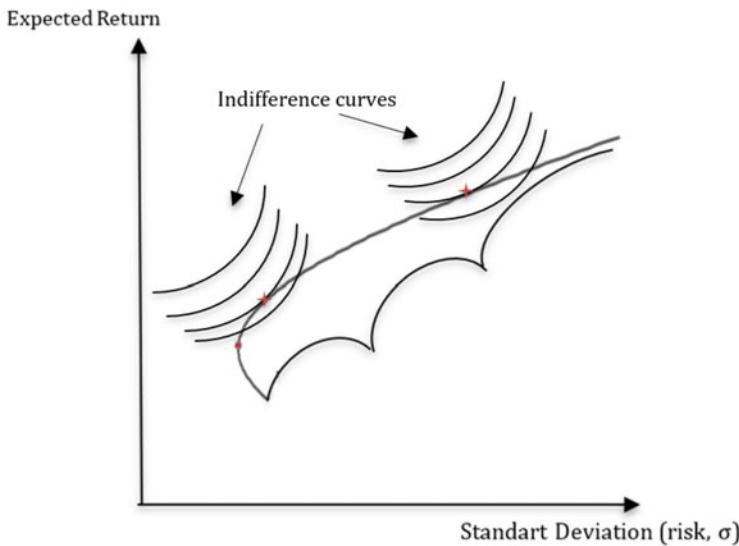


Fig. 2.4 Different indifference curves and efficient portfolio. Source: Drawn by the author

choose the optimum portfolio between all opportunity sets. There are many techniques and methods to create an optimal portfolio. PSO technique is one of those, which is used to determine an optimal portfolio.

Since the main subject of this book is portfolio optimization with PSO technique, this chapter focuses on the basic concepts of portfolio optimization. Mathematical calculation methods of portfolio risk and return are given, and the logic of diversification is explained. The criteria to be taken into consideration when choosing the optimum portfolio are mentioned.

According to the modern portfolio theory, the correlation coefficient between the expected returns of the assets in the portfolio should be less than one in order to reduce the risk with diversification. The risk-return area created by the portfolios consisting of possible combinations of all assets in the market constitutes an opportunity set. Investors will prefer a portfolio at the top line of this opportunity set, as they expect the highest return at a given risk level. In other words, portfolios that yield the highest returns at a certain risk level are effective portfolios, and this top line is called the efficient frontier. At what point investors will stand on the efficient frontier depends entirely on their risk attitude. These are the portfolios at the point where the indifference curves that maximize the utility are tangent to the efficient frontier. In summary, after determining the most effective portfolios among all the assets available in the market, revealing the expected rate of return-risk preferences derived from the specific utility functions of the investors with the indifference curves answers the question of which portfolio is the optimal portfolio for each investor.

References

- Adiguzel Mercangoz, B. (2019). Particle swarm algorithm: an application on portfolio optimization. In R. Jhuma, M. Anirban, K. D. Sedhan, & K. Goran (Eds.), *Metaheuristic approaches to portfolio optimization* (pp. 27–59). ABD: IGI Global.
- Adıgüzel Mercangöz, B., & Eroğlu, E. (2019). The genetic algorithm: an application on portfolio optimization. In R. Jhuma, M. Anirban, K. D. Sedhan, & K. Goran (Eds.), *Metaheuristic approaches to portfolio optimization* (pp. 154–178). ABD: IGI Global.
- Bodie, Z., Kane, A., & Marcus, A. J. (2010). *Investments* (10th ed.). Irwin: McGraw-Hill.
- Chen, W., Chung, H., Ho, K., & Hsu, T. (2010). Portfolio optimization models and mean covariance spanning tests. In C.-F. Lee et al. (Eds.), *Handbook of quantitative finance and risk management* (pp. 165–184). Berlin: Springer.
- Kendal, G., & Yan, S. (2005). *A particle swarm optimization approach in the construction of optimal risky portfolios*. 23. IASTED international multi conference artificial intelligence and applications journal, February 14–16 (pp. 140–145). Australia: Innsbruck.
- Markowitz, H. (1952). Portfolio selection. *The Journal of Finance*, 7(1), 77–91. March. 1952. www.jstor.org.proxy.lib.chalmers.se/stable/10.2307/2975974?origin=api (2012-1030).

Chapter 3

Behavioral Portfolio Theory



Murat Akkaya

Abstract The aim of this study is to explain behavioral portfolio theory in a theoretical way. The study starts with the definition of portfolio which is a financial asset that consists of various securities such as stocks and bonds and derivative products, held by a particular person or group. Also, portfolio management is the management of securities according to investors' returns and risk targets. There are two basic portfolio management theories in finance literature. The first is the traditional portfolio (simple diversification) approach based on the diversification of securities. The second is the modern portfolio theory, which is based on a more mathematical basis. Modern portfolio theory mathematically shows the measurement of the risk and return of a portfolio of two or more securities and the determination of optimal portfolios. Markowitz's mean-variance model is the first mathematical explanation of the idea of diversifying investments and is the cornerstone of many risk models developed such as capital asset pricing model and arbitrage pricing model in later years. The empirical studies reveal that investors do not act rationally as financial models assume and anomalies occur. Thus, behavioral finance tries to fill the gap in this area and states that investors should be considered "normal" rather than rational. Prospect theory developed by Kahneman and Tversky (1979), brings psychological explanations to finance issues. Mental accounting in behavioral finance prevents the rule of taking into account the correlation between the returns of an investment. Also, herd behavior of investors distorts the efficiency of the markets and leads to volatility in financial markets. When investors show herd behavior, they do not care about the information received and tend to imitate other investor behavior. Behavioral portfolio theory (BPT) emerged as a descriptive alternative to Markowitz's mean-variance portfolio theory. BPT connects two issues, the creation of portfolios and the design of securities. There are two versions of the BPT model: single mental accounting (BPT-SA) in which the portfolio is integrated into one mental accounting and multiple mental accounting (BPT-MA).

M. Akkaya (✉)
T.C. İstanbul Arel University, Istanbul, Turkey
e-mail: muratakkaya@arel.edu.tr

Keywords Asset pricing models · Behavioral finance · Behavioral portfolio theory

3.1 Introduction

Individual and corporate investors trade stocks, bonds, warrants, bills, repo, foreign currency, gold, etc., in financial markets. They create portfolios by investing in different assets. Portfolio means literally a wallet. Portfolio refers to all of the securities held by investors or used on behalf of the investor. In terms of finance, the portfolio refers to a cluster of securities and is defined as the new financial asset created by combining multiple financial assets. Portfolio is a financial asset that consists of various securities such as stocks and bonds and derivative products, held by a particular person or group. Various portfolios can be created according to investors' different returns and risk preferences. The important thing here is the preferences of the investor.

The rational investors try to achieve the highest return together with the risk level. The investor needs to consider the nonsystematic risk as well as the systematic risk during portfolio creation and evaluation. In this respect, the importance of portfolio management becomes apparent. Portfolio management is a system of continuing and updating the evaluation of the success of investors' goals, preferences, and constraints determined and resolved under this information by monitoring the portfolio. Also, portfolio management is the management of securities according to investors' returns and risk targets. The portfolio manager tries to provide the expected return to the portfolio owner or investor with the necessary diversification at an acceptable risk level. The aim of portfolio management is to reduce the risk through diversification. In finance and markets, this situation is defined as "not putting all of the eggs in one basket."

In the portfolio management process, requirements are:

- Determination of investment policies according to the investor's preferences.
- Development of investment strategies in market conditions.
- The values of investment instruments and the constant consideration of the investor.
- Making changes in portfolio composition according to changing conditions.
- To keep customers informed.

This process includes a sequential analysis when making decisions. In the portfolio selection process, as a result of the analyses made during the portfolio planning and investment analysis phase, the financial instruments to be included in the portfolio are selected. Portfolio management is a dynamic process. The success of the portfolio needs to be measured in certain periods. Thus, the achievement of the targets determined at the beginning of the investment process is evaluated. It is determined whether there is a need to make a change in the portfolio content.

There are two basic portfolio management theories in finance literature. The first is the traditional portfolio (simple diversification) approach based on the

diversification of securities. The second is the modern portfolio theory, which is based on a more mathematical basis. The traditional portfolio approach lasted until 1952 when Harry Markowitz's modern portfolio management approach emerged.

3.2 Traditional Portfolio Management Approach (Simple Diversification)

The traditional portfolio management approach (simple diversification) was widely used until the 1950s. Although this approach has no scientific basis, there is ease of implementation. Here, it aims to provide maximum return and distribute the risk. In the traditional portfolio management approach, securities selection and simple diversification are made according to different sectors. In addition, numerical methods are not included in the selection of securities. It is assumed that a diversification of securities of businesses in different sectors will have a positive effect (Kıyılar & Akkaya, 2016).

Traditional portfolio theory is based on simple diversification. The correlations between the returns of the securities in the portfolio are not considered, and it is foreseen that the risk level will be decreased by increasing the number of securities in the portfolio. The purpose of creating a portfolio is to distribute the risk. The returns on the securities that make up the portfolio will not move in the same direction, so the risk of the portfolio will be less than the risk of a single security. Based on this principle, the traditional portfolio theory is based on the principle of increasing the number of securities in the portfolio. The purpose of portfolio management is to get the highest utility from portfolio investments. According to the traditional portfolio approach, the utility can be obtained by the highest return from the securities portfolio at the highest acceptable risk level. The investors' risk attitude determines the content of the portfolios.

The traditional portfolio approach has many criticisms. This portfolio management has some drawbacks. These drawbacks are:

- Inclusion of securities in the portfolio that do not provide the required return regardless of risk.
- Managerial difficulties of the portfolio with many securities.
- Increasing analysis and research costs from the excess number of securities in the portfolio.
- Frequent purchase of small amounts of securities so that more commissions are paid.

This method has been criticized a lot due to the weakness of its scientific basis. In addition, quantitative information is not considered important in the selection of securities. The result of these criticisms is modern portfolio theory, based on mathematical and statistical methods.

3.3 Modern Portfolio Theory

Harry M. Markowitz presented the mean-variance model and modern portfolio theory and received a Nobel Prize for his work. Markowitz (1952) introduced the concept of risk mathematically (standard deviation of the return rate of an asset). Modern portfolio theory emphasizes that risk cannot be reduced by portfolio diversification and risk can be minimized at a certain return level through diversification in portfolio management. Modern portfolio theory is a comprehensive approach that includes issues such as indifference curves, efficient frontier, opportunity set, and optimal portfolio selection, along with the correlation relationship between the returns of securities. Modern portfolio theory mathematically shows the measurement of the risk and return of a portfolio of two or more securities and the determination of optimal portfolios. Correlations between securities returns should also be taken into account in diversification. The risk of a portfolio of securities such as A and B depends on the covariance and/or correlation coefficient between the returns of A and B. With Markowitz diversification, lower risk portfolios can be obtained compared to simple diversification without considering correlations.

Assumptions of modern portfolio theory (MPT) are as follows:

- The investor determines each investment by looking at the probability distributions of the expected return to be provided by the alternative.
- Investors try to maximize their expected utilities for a single period. Utility curves are compatible with decreasing marginal utility.
- Investors identify portfolio risk as a deviation from the expected return.
- Investors make their investment decisions based on expected returns and risks.
- Investors try to increase their returns at a certain risk level.

According to MPT, investors are rational and determine their portfolios according to the risk-return level. Rational investors will prefer the highest return at a certain risk level and the portfolio with the lowest risk at a certain return level. Modern portfolio theory shows ways to achieve maximum return at a certain risk level in order to create its own optimum portfolio. In this framework, the expected returns, standard deviations, and covariances among all shares must be calculated.

Markowitz has three important contributions to traditional theory.

1. In portfolio management, the sum of the parts is not equal to the whole.
2. While some portfolios of investors provide the same return, they are more risky; some portfolios are at the same risk level, and they do not prefer those portfolios because they provide less returns, so some portfolios are superior to others.
3. Many calculations can be made with the effective limit.

The slope of the indifference curve is different for investors. The investor determines the slope and location of the indifference curve according to the expected return and risk level. The fact that the slope of indifference curves is high indicates that the investor avoids risk. The low slope of indifference curves indicates that the investor takes more risks. The efficient frontier, on the other hand, is a point in a

two-dimensional diagram with the expected return rate and risk elements, which is risk on the X-axis and expected return rate on the y-axis. Possible portfolios on the efficient frontier are listed in terms of risk and return. Portfolios below the efficient frontier are accessible but not effective. There are portfolios that cannot be accessed above the efficient frontier.

The investor's utility functions comprise the risk and expected return rate. Modern portfolio theory allows the determination of optimum portfolios for different investors with utility functions. In line with the risk-return preferences of investors, effective portfolios can be created that provide more expected return opportunities at all risk levels. In choosing the optimal portfolio, the risk-expected return rate preferences derived from the investor's unique utility functions should be put forward by indifference curves. The utility function, the attitude to risk, determines which of the portfolios the investor chooses on the efficient frontier. Investors will choose the portfolio that will maximize their utility from these portfolios. Therefore, the investor invests in the portfolio at the point where the indifference curve is tangent to the efficient frontier. Portfolio is optimal where the indifference curves are tangent to the limit of effectiveness.

The expected return of a portfolio consists of the weighted average of the returns on the securities in the portfolio. The total weight of the portfolio must be 1 or 100%. The expected return of the portfolio is:

$$E(r_p) = \sum_{i=1}^n w_i \cdot E(r_i)$$

$E(r_p)$ Expected return of the portfolio

w_i Portfolio weight of securities

$E(r_i)$ Expected return of the security i

N Number of securities in the portfolio

The risk of a portfolio consists of the weighted average of the standard deviations of the securities in the portfolio as in the expected return of the portfolio. Thus, the risk of the portfolio is lower than the standard deviation of the securities that make up the portfolio. The following is the standard deviation of the portfolio:

$$\sigma_p = \sqrt{\sum_{i=1}^n \sum_{j=1}^n w_i w_j \sigma_{ij}}$$

σ_p Portfolio risk

W_i Weight of w_{i-i} securities in the portfolio

W_j Weight of the securities w_{j-j} in the portfolio

σ_{ij} Standard deviation of each security

The relation between the expected returns of the securities included in the portfolio is covariance. The covariance shows the direction of the two variables moving together or changing them. The positive covariance indicates that two securities move in the same direction at the same time. If the covariance is negative,

it shows that it moves in the opposite direction at the same time. If the two securities do not move independently, in the same or opposite direction, there is no covariance. Covariance is calculated as follows:

$$\text{Cov}(i, k) = \frac{1}{N-1} \sum_{t=1}^N (R_{it} - E(R_i))(R_{kt} - E(R_k))$$

$\text{COV}(i,k)$ Covariance between i and k securities

R_{it} The return of i securities in period t

R_{KT} Return of k securities in period t

$E(R_i)$ Expected return of i security

$E(R_k)$ Expected return of k securities

N The number of possible returns

The correlation coefficient also measures the degree of the relationship between the returns of the two securities. The correlation coefficient (+1) indicates that there is a complete and positive linear relationship between securities returns. Securities returns move together. Having a correlation coefficient (-1) indicates that there is a complete and negative linear relationship between securities returns. Securities returns move completely opposite. The correlation coefficient (0) indicates that there is no relationship between securities returns. Correlation coefficient is calculated as follows:

$$\rho_{ij} = \text{Cov}(r_i, r_j) / \sigma_i \sigma_j.$$

ρ_{ij} Correlation coefficient between i and j securities

$\text{Cov}(r_i, r_j)$ Covariance value between i and j securities

σ Security standard deviation

Markowitz's mean-variance model is the first mathematical explanation of the idea of diversifying investments and is the cornerstone of many risk models developed in later years. The efficient markets hypothesis emerged in the 1960s and became important in the last quarter of the twenty-first century. This model is based on "random walk," and price changes are random and cannot be predicted beforehand. Stock price changes have no historical connection, and these price series cannot be used to predict the future in a meaningful way. One of the most important assumptions of the efficient markets hypothesis is that any investor will not be able to generate abnormal returns by using any information because prices contain all available information. The effectiveness of a market depends on the following conditions (Fama, 1970):

- Prices should reflect the market balance with all available information.
- Prices should reflect the response to new information that is neutral or immediate, with little delay.

Since price movements in an efficient market will occur in line with the information received while the current price of a security reflects all available

information, it will be independent of consecutive price changes or consecutive quarterly returns. Fama divides market efficiency into three categories: weak form, semi-strong form, and strong form.

Sharpe developed the Markowitz method and introduced a simple model (single index model) in 1963. Sharpe (1964), Lintner (1975), and Mossin (1966) investigated the change of prices if savers invested in securities and especially stocks in accordance with the modern portfolio theory. As a result of these studies, capital asset pricing model (CAPM) was developed. Roll (1977) claimed that the model was inadequate and suggested that the model should not be used in portfolio management. In the same period, Steve Ross alternatively introduced the model known as arbitrage pricing theory (APT).

3.4 Index Models

The Sharpe model links the relationship only to the market index, rather than the relationship (correlation) between stock returns. This model attributes the returns of various stocks to only a basic factor. There is a linear relationship between all securities and the market, and this relationship can be expressed with a simple linear regression model.

The difference of the multi-index model is that it associates the returns of financial assets with more variables rather than market returns. These variables include interest, inflation, etc., such as independent macro variables. Arbitrage pricing model is the extension of multiple index model.

3.5 Capital Asset Pricing Model (CAPM)

CAPM is one of the models used in calculating equity cost, valuation of capital assets, and determining the optimal portfolio. The approach, which is accepted as the first of the asset pricing models, accepts the market portfolio as the only variable and tries to explain the returns of all risky financial assets with the returns of the market portfolio. It states that the systematic risk expressed as the model beta (β) coefficient is the only factor affecting returns.

The financial asset pricing model expresses the expected return of a security as a function of risk-free interest rate, systematic risk of investment, and market portfolio. In this model, the factor that determines how much the expected return will be than the risk-free interest rate is the beta (β) coefficient. The return on a stock must be as much as the risk premium than the risk-free rate return. Risk premium, on the other hand, is calculated by multiplying the difference between the index return on the market and the risk-free interest rate by beta. Returns of the market index are considered to be market returns.

Expected return = Risk – free interest rate + Risk premium.

$$\begin{aligned}\text{Expected return} &= \text{Risk – free interest rate} \\ &\quad + [\beta \times (\text{Market return} – \text{Risk – free interest rate})].\end{aligned}$$

CAPM tries to explain the risk-expected rate of return relationship of portfolios or assets mathematically. The expected return of an asset in the model consists of the systematic risk and risk premium of the asset. CAPM measures the risk beta coefficient and determines the expected rate of return for a security. The expected rate of return of an asset is a function of the risk-free rate of return and the risk premium.

Beta (β) is used in the calculation of systematic risk and also expressed as the ratio of the covariance between the return of an asset and the return of the market portfolio to the variance of the market return. The mathematical expression of the beta is as follows.

$$\beta_i = \text{COV}(R_i, R_M) / \sigma_M$$

The beta coefficient (β) indicates the tendency of the asset's rate of return to move with the market. It is used as a measure of the variability of the asset according to the market portfolio. Securities with a β value = 1 are in the middle risk group, and their returns are at the market portfolio level. Financial assets with a $\beta > 1$ have a high systematic risk, and the expected return will be high. If the β is < 1 , financial assets will have low systematic risk, and expected return will be low.

Capital asset pricing model is shaped as follows:

$$E(R_i) = R_f + \beta_i (E(R_m) - R_f).$$

(R_i) The rate of return expected from the investment

R_f Risk-free interest rate

$E(R_m)$ Expected return of the market

$(E(R_m) - R_f)$ Market risk premium

B Systematic risk criterion which refers to beta coefficient

The expected return-risk relationship in CAPM is based on a single factor called the market portfolio, and the variability in return rates is explained only by this factor. This understanding causes criticism and doubts directed at CAPM. Observing and experimental testing of the market portfolio is quite difficult. In addition, CAPM is accepted as a remote and limited theory to real-world conditions. Alternative forms of FVFM have been developed upon criticism. They are zero-beta CAPM, multiperiod CAPM, conditional CAPM, multi-beta CAPM, international CAPM, and consumption-based CAPM.

3.6 Arbitrage Pricing Theory (APT)

The arbitrage pricing theory (APT) is developed against the criticism brought to CAPM by Ross (1976). Arbitrage is a transaction based on obtaining risk-free profit by taking advantage of the price differences of similar financial assets, buying the relevant asset in the cheap market, and selling it in the expensive market. In APT, it is accepted that there is a positive relationship between return and risk where the return on securities is affected by factors in the industry and the market. The basis of the arbitrage pricing theory is the recognition of important systematic factors that affect the long-term average returns of financial assets.

The balance prices between the expected return and risk of the portfolio are calculated using the multiple factor model. The return of each stock or portfolio is specified as a function of multiple common risk elements for all stocks or portfolio. There are multiple factors in APT, and asset return rates are a linear function of these factors. These factors may be macroeconomic variables or company factors. The model does not explain what factors might affect the price of a financial asset.

The expected returns in APT are assumed to be in a linear relationship with a set of indices. Accordingly, it is assumed that the returns of the assets are derived by a linear “ k ” factor model.

$$R_i(t) = E[R_i(t)] + b_{i1}f_1(t) + b_{i2}f_2(t) + \cdots + b_{ik}f_k(t) + \epsilon_i(t).$$

$R_i(t)$ Return rate of the asset i at time t

$E[R_i(t)]$ Expected rate of return of the asset i at the beginning of time t

B_{ij} The sensitivity of the presence of i against the risk factor j ($j = 1, 2, \dots, k$)

F_j The value of the risk factor j at time t

ϵ_i The amount of nonsystematic risk

Ross (1976) shows that if the number of assets is large enough, the risk-return relationship will be as follows.

$$E(R_i) = R_f + b_{i1}[E(R_1) - R_f] + b_{i2}[E(R_2) - R_f] + \cdots + b_{ij}[E(R_j) - R_f].$$

3.7 Behavioral Finance

One of the assumptions of traditional financial theories assumes that all investors are rational. The results of the empirical researches determine that the investors could not do this even though they wanted to maximize their benefits and avoid risk. This is based on cognitive defects. The empirical studies reveal that investors do not act rationally as financial models assume and anomalies occur. Thus, behavioral finance tries to fill the gap in this area and states that investors should be considered “normal” rather than rational. In addition, it assumes that investors consider

variables other than risk and return in their investment decisions and that it is not the one that maximizes the utility, but at the very best, the decisions that the individual himself will be satisfied with.

The articles of Kahneman and Tversky published in *Econometrica* in 1979 are accepted in the literature as the beginning of behavioral finance. Kahneman and Tversky's work in the field of judgment and decision-making started a period in the field of finance. These studies form a link between finance and psychology. The prospect theory of Kahneman and Tversky forms the basis of behavioral finance. Prospect theory assumes that investors give different weights at different probability levels to earnings and losses, and losses are more important than earnings. Psychological factors such as investors' heuristics and the irrational arbitrage limit constitute behavioral finance (Kiyilar & Akkaya, 2016).

Behavioral finance deals with investor behavior and its impact on financial markets. It contributes to investor psychology and the inclusion of behavioral characteristics of investors in asset pricing models. Behavioral finance makes explanations on shaping investor behavior and investment decisions, how basic information is interpreted, and other financial factors in addition to financial factors in investor's decision (Barberis & Thaler, 2003).

Prospect theory developed by Kahneman and Tversky (1979) brings psychological explanations to finance issues. This theory consists of experimental research results and is a mathematically formulated theory against the expected utility theory. Prospect theory further incorporates human psychology into the economic model and forecasts. It is revealed by the findings in this theory that individuals can make irrational choices. In the prospect theory, as in the expected utility theory, investors are trying to maximize the utility. However, it is the most multiplexing under special expectations with new rules. Psychological factors in prospect theory cause people to deviate from rationality systematically. Individuals give different weights at different probability levels in terms of their regions of earnings and losses. In addition, common mistakes of many investors in the same direction cause investor sentiment (Kiyilar & Akkaya, 2016).

Behavioral finance models emphasize that the efficient market hypothesis is not valid. It states that markets are not fully active and there are deviations from market efficiency. Many empirical studies have been done on this subject in the last 20 years. Three models have emerged with these studies:

- The “representative agent” model of Barberis, Shleifer, and Vishny (1998) based on psychological findings.
- The models of Daniel, Hirshleifer, and Subrahmanyam (1998) based on overconfidence and biased self-attribution.
- Interactive relationship models of Hong and Stein (1999) and heterogeneous investors.

The prospect theory emphasizes the effects of investors' psychological and emotional tendencies on financial decisions. Investors keep them away from acting rationally by using their own shortcuts in the decision-making process. Investors use mental accounting in their financial decisions. As a result of psychological

tendencies, it causes the investor to think of the portfolio as a pyramid of assets. Each floor of the pyramid corresponds to the desire of the investor to meet its purpose. Individuals have individual mental calculations suitable for investment purposes and tend to take different levels of risk. Therefore, individuals select the assets that meet the expected return and risk of the mental account for each mental account. The basic need of individuals is security. Individuals first prefer safe beings to satisfy this mental calculation and then invest in assets that will satisfy mental calculations for high returns and risk levels that correspond to other layers of the pyramid (Shefrin & Statman, 2000). Investors build their decisions on beliefs that include the possibilities of uncertain events. These beliefs are expressed as “I think . . .” like “me.” In rare cases, numerical form or subjective possibilities are used. Three heuristics (cognitive shortcut) are used to make decisions under uncertainty: representation, availability, and anchor or correction. These incentives serve to make economic, easy, and effective decisions. However, it also leads to systematic estimation errors (Tversky & Kahneman, 1979).

Existing emotional factors also have an impact on investors' decisions and behavior. Cognitive factors are related to how people organize and use information, while emotional factors are related to the emotional aspect of individuals in the record of information (Shefrin & Statman, 2000). Ambition, fear, hope, pride, and regret affect investment decisions. Emotional factors that often appear in financial markets include avoiding regret, cognitive contradiction, hedonic correction, predisposition effect, easy money effect, risk avoidance, and break-even effect (Kiyilar & Akkaya, 2016).

Individuals avoid the actions they will regret and look for events that will make them proud. The effects of feelings of pride and regret are inverted. The sense of regret is an emotional state and a negative emotion resulting from the poor outcome of an investment decision. It is strongly felt by people. The sense of pride is manifested by the positive outcome of an investment decision and is a state of emotional pleasure. Pride is on the opposite side of regret. In order not to regret, people want to rationalize the situation and make changes in the cognitive process.

Herd behavior is also an important phenomenon regarding the effects of investor behavior on financial markets. In order for herd behavior to occur, it must be imitation, that is, investors must know the decisions of other investors. It is herd behavior that investors act together like a lot and also buy the same assets. Herd behavior of investors distorts the efficiency of the markets and leads to volatility in financial markets. When investors show herd behavior, they do not care about the information received and tend to imitate other investor behavior. In addition, mental accounting is the process of coding and evaluating the results briefly. Thaler (1999) through the concept of mental accounting defines the accounting system as recording and summarizing transactions and analyzing, auditing, and reporting the results. Markowitz's modern portfolio theory includes methods for reducing portfolio risk and effective diversification. According to the theory, it is the correlation between securities that is important in portfolio creation and risk distribution. Mental accounting in behavioral finance prevents the rule of taking into account the correlation between the returns of an investment, which is the most essential feature of

diversification, because instead of making an effective diversification in reducing portfolio risk, investors take into account the investments one by one and open separate mental accounts and evaluate them. Recording the result of a choice in a mental account affects the evaluation of the choice. Saving different transactions in different mental accounts and evaluating these accounts separately and independently may lead to wrong decisions in investment preferences and timing. How the results of financial decisions are evaluated is in the subject of mental accounting. Mental accounting causes investors to make irrational decisions.

The coexistence of this type of risk-avoiding and risk-seeking behavior patterns causes deviations from the utility function. In addition, the expected utility theory is replaced by the expectation theory, and the modern portfolio theory is also replaced by the behavioral portfolio theory.

3.8 Behavioral Portfolio Theory

The empirical studies on modern portfolio theory to investment portfolios in the real world present anomalies and problems. Modern portfolio theory is theoretical and tries to show how transactions take place in capital markets. Also, MPT does not give any advice on how to design investment portfolios. It is useful under certain principles, some of which investors know and some of which they are not familiar with. Thus, modern portfolio theory is descriptive, not prescriptive. Modern portfolio theory's assumption that all investors are always rational and they are wealth enhancers is clearly false. Also, as a result of these problems, when financial advisors try to communicate with their customers about their portfolios using modern portfolio theory structures, communication is largely stopped (Curtis, 2004). The main purpose of investors is to make a profit. Investors often focus on performance, forecast, market timing, and some other factors before investing. Investments made based on these factors provide average returns. Investors can sometimes experience dissatisfaction due to their belief that they can do better than an average return. The difference between expected and realized earnings causes this situation. Traditional models assume that the subjective probabilistic beliefs of the investors are objectively correct and indirectly the markets are efficient. In this environment, the purpose of the portfolio is to manage the risk profile of the investor's consumption flow based on the initial asset and stochastic labor income (Hoffmann, Shefrin, & Pennings, 2010).

Behavioral portfolio theory (BPT) emerged as a descriptive alternative to Markowitz's mean-variance portfolio theory. BPT connects two issues: the creation of portfolios and the design of securities (Shefrin & Statman, 2000). BPT by Shefrin and Statman gets roots from Roy's (1952) safety first approach. Roy does not use investor's portfolio risk but uses the probability of ruin. BPT is also an alternative portfolio choice model developed to better capture these different features. Also, the basis of BPT is in sharp contradiction with the fundamental points of modern portfolio theory. Thaler (1999) presents that individuals divide their current and

future assets into separate and nontransferable parts in the mental accounting approach. Individuals make different mental calculations that are handled separately and in different ways. BPT integrates the mental accounting structure of Kahneman and Tversky (1979) and Tversky and Kahneman (1992) and also helps investors to view their portfolios as a collection of subportfolios, each suitable for a particular mental accounting. Das, Markowitz, Scheid, and Statman (2011) combine important features of modern portfolio theory and behavioral portfolio theory and create a new mental accounting (MA) framework.

Pfiffelmann, Roger, and Bourachnikova (2016) clarify that “The three core features of BPT are: (1) investors seek to secure a minimal level of their wealth; (2) investors consider their portfolio as a collection of subportfolios, each of them being optimal for a given mental account; (3) investors do not behave rationally as they exhibit optimistic and pessimistic behaviors.”

The basis of behavioral portfolio theory presented by Shefrin and Statman (2000) is the prospect theory and the two-factor Lopes theory (1987). Then they compare it with modern portfolio theory. The prospect theory by Kahneman and Tversky (1979) begins with the observation that people facing complex problems turn these problems into simpler subproblems. Prospect theory divides the decision-making process into two processes, namely, the editing phase and valuation phase. The editing phase consists of a preliminary analysis of the preferences evaluated. These preferences can be expressed simply here. In the evaluation phase, the formed preferences are evaluated, and the expectation with the highest value is preferred. The function of the editing phase is to organize and reformulate preferences for easy evaluation and preference. Following the editing phase, it is assumed that the decision maker evaluates each corrected option and prefers the option with the highest value. The editing phase is followed by value function phase. The important assumption of the prospect theory is that the premise of value is not the final state of the assets owned, but the change in wealth. This assumption is also compatible with basic perception and judgment principles. The theory focuses on the evaluation of changes or differences rather than size. In the prospect theory, the value functions of gains and losses are different. Investors make decisions based on potential gains and losses based on a reference point. The risk trend of people varies according to the area where the value function is located. Individuals tend to take risks if they evaluate their investments in the area of earnings and if they evaluate in the loss zone. In many experiments, the utility function of the individual is concave in the gain area and convex in the loss area (Kiyilar & Akkaya, 2016).

Modern portfolio theory assumes that investors evaluate portfolios as a whole and consider the covariances between assets when creating their portfolios. Also, average variance investors not only care about their individual assets but also look for the expected returns and the variance of the total portfolio. Average variance investors have consistent attitudes toward risk, and they always avoid risk. Typical investors ignore covariances between assets, but some traders use procedures that help consider covariances both institutionally and individually (Shefrin & Statman, 2000). Also, BPT considers it as a probability weighting function rather than the actual probability distribution used in MPT (Bourachnikova, 2007). BPT shows that

investors are risk-averse and risk-seeking therewithal. Hoffmann et al. (2010) present that BPT emphasizes “the role of behavioral preferences in portfolio selection and proposes that individual investors’ portfolio choices and consequently return performance reflect characteristics such as aspirations, hope, fear, and narrow framing.”

The main assumption of BPT is that investors do not consider portfolios as a whole which is the main proposition of MPT. However, investors have separated mental account layers, in which each mental account layer is joined with a specific goal and certain attitude toward risk. Thus, investors’ attitudes and goals in risk differ across layers (Statman, 2004, 2008). Shefrin (2015) explains mental account layers. “The first mental account reflects investor’s aversion to risk at the lowest layer. The second is characterized by a moderate level of risk tolerance, which explains why a combination of stocks and bonds take place at the middle layer. The third account explains investor’s preferences to take more risks, that explains why stocks of a small number of companies, or even of a single company, account for dominant assets at the topmost layer.” The portfolios in BPT are similar to layered pyramids. Investors design their portfolios as an asset pyramid on the basis of layers. Layers are created in relation to specific goals and specific attitudes to risk. Layers are associated with different targets, and covariances between layers are ignored. The potential layer above is designed for a chance to get rich. BPT estimates that an increase in transaction costs will reduce the number of securities in each layer of the portfolio. An increase in the area allocated to one tier will increase the number of securities in that tier (Shefrin & Statman, 2000). Investors form layered portfolios that are protected against risks and form a condition to get profit simultaneously (Shefrin, 2015). The investors’ main goal is to form a portfolio that satisfies the goals qualified at each layer of the portfolio pyramid (Das et al., 2011).

Lekovic (2019) clarifies optimal portfolio. “MPT investors have only one efficient frontier to think of, whereas BPT investors have a number of efficient frontiers to consider—one for each mental account (Das et al., 2011). Therefore, rather than selecting a single and the most optimal portfolio, normal investors choose several optimal sub-portfolios—one per each layer of the portfolio pyramid. The optimal portfolio is constructed by combining optimal sub-portfolios. According to H. Shefrin and M. Statman (2000, 128), an MPT optimal portfolio is a combination of the market portfolio and risk-free securities, whereas a BPT optimal portfolio resembles a mix of bonds and state lottery tickets because it actually represents a combination of such optimal sub-portfolios.”

MPT assumes that the optimal portfolio varies depending on investor’s attitudes toward risk. But BPT explains the optimal portfolio variation between investors is due to the different levels of risk tolerance and also investors’ different needs, biases, preferences, shortcuts, and emotions. Thus, BPT’s behavioral-wants frontier does not match with the MPT’s mean-variance frontier, and the optimal portfolios differ in the optimal MPT and BPT portfolio.

The optimal portfolio in BPT is at preferred risk level that maximizes the all utilities of an investor as a sum of utilitarian, expressive and emotional utilities. Portfolio E as a rational investor is on mean-variance frontier. F is on behavioral-

wants frontier. E ignores investor sentiments and emotions. However, F reflects psychological and emotional benefits. Thus, F is below E. This is because lower expected return at the same level of risk usually comes from the realization of psychological and emotional benefits.

Behavioral portfolio is based on the assumption that the vast majority of investors make decisions based on emotions and shortcuts. Behavioral portfolio management (BPM) has two categories for financial market participants: emotional crowds and behavioral data investors (BDIs) or rational investors. Emotional crowds consist of investors who make decisions with anecdotal evidence and emotional responses to uncover events. Emotional investors make decisions quickly, with little or no effort and voluntary control, avoiding loss automatically. BDIs also make decisions using detailed and comprehensive analysis of available data. BPM is based on dynamic interaction between these two investor groups. Rational investors or BDIs react to distortions that arise by taking positions across the emotional crowd. However, they are not an important factor in keeping prices at the same level. As a result, the distortions that occur are measurable and permanent. BDIs can create portfolios that take advantage of these distortions simply or rationally as crowded investors move in another direction. Cases that trigger the response of crowded investors may be short-lived, but subsequent emotions are long-lived. As a result, price distortions are both measurable and permanent. This provides BDI traders the opportunity to identify distortions and create portfolios that benefit from them (Howard, 2014).

There are two versions of the BPT model: the single mental accounting (BPT-SA), where the portfolio is integrated into one mental accounting, and the multiple mental accounting (BPT-MA), where the portfolio is divided into multiple mental accounts. BPT traders, i.e., the investors in the Friedman-Savage puzzle, display a risk-avoiding and risk-seeking behavior at the same time. Portfolios in BPT-MA are similar to layered pyramids where each layer, for example, mental calculation, sustains a certain level of aspiration. This is because BPT-MA traders ignore covariance between layers and also can associate the position in one layer with another position in the same security in another layer.

Pfiffelmann et al. (2016) clarify that “in BPT, investors maximize their expected wealth (calculated with decision weights) subject to the constraint that the probability of failure to reach the threshold level A does not exceed a given level. This optimization program writes:

$$\max E(W) \text{ } u.c.p(W < A) <$$

where A is the aspiration level, the maximum probability of ruin, W is the final wealth distribution and a transformation function of probabilities. The literature proposes several ways to transform probabilities. We choose to use the specification proposed by Tversky and Kahneman (1992) in cumulative prospect theory. This model is detailed in Appendix A.” Pfiffelmann et al. (2016) observe that the optimal BPT portfolio is on the MPT’s efficient frontier in more than 70% of the cases.

3.9 Behavioral Asset Pricing Model

Shefrin and Statman (1994) develop behavioral asset pricing model (BAPM) for the price efficiency by the presence of noise traders and investigate the effects of noise traders on price efficiency, volatility, return anomalies, and noise trader existence. BAPM is an alternative to capital asset pricing model (CAPM), Fama-French three-factor model, and arbitrage pricing theory (APT). CAPM and efficient market hypothesis (EMH) can be useful for markets where rational investors and information traders take place. But in reality, such markets do not exist in the world. They are hypothetical. Also, various psychological and emotional factors lead investors to behave irrationally. Noise traders and their irrational decisions also affect the returns.

“BAPM model prices assets based not only on their utilitarian benefits, but also on expressive and emotional ones. Therefore, in line with the BAPM model, assets are worthy because they bring utilitarian benefits (low risk, high returns), as well as expressive (social responsibility, patriotism) and emotional (satisfaction, pride, excitement caused by trading) benefits” (Leković, 2019).

Statman (2017) compares investment asset pricing models with pricing models for meals, cars, films, and other products and services. The expected prices of a meal are same: utilitarian benefits, expressive and emotional benefits and cognitive and emotional errors which are the theoretical rationales. “Theoretical rationales in standard asset pricing models account only for wants for utilitarian benefits, whereas theoretical rationales in behavioral asset pricing models also account for wants for expressive and emotional benefits and the occurrence of cognitive and emotional errors.”

BAPM assumes that emotions and affects have a significant role in the pricing of financial assets because individuals invest the assets with a positive affect or avoid them with a negative affect. Also, the formation of mental schemas may lead to irrational decisions because mental schemas are the subjective experiences existing at the unconscious level and they affect the perception and reasoning. Emotions and mental schemas may cause the anomalies in financial markets.

Smart beta portfolios are portfolios in which the proportions of assets depart from their proportions in the market portfolio, aiming at higher ratios of expected returns to standard deviations than provided by that portfolio. A portfolio tilting toward small and value stocks is a smart beta portfolio because it assigns greater proportions to small and value stocks than their proportions in the market portfolio. So is a portfolio tilting toward small and value stocks and also toward momentum stocks and other stock characteristics associated with positive excess returns. We can see smart beta portfolios and indexes as reflections of behavioral asset pricing models in which differences between expected returns of portfolios depend not only on differences in their market-factor betas but also on differences in the betas of other factors. They include the small-large factor, value-growth factor, momentum factor, and other factors that distinguish portfolios with high expected returns from portfolios with low expected returns (Statman, 2017).

3.10 Solutions and Recommendations

Behavioral portfolio theory (BPT) emerged as a descriptive alternative to Markowitz's mean-variance portfolio theory. BPT connects two issues: the creation of portfolios and the design of securities. Behavioral asset pricing model (BAPM) is a new concept.

3.11 Future Research Directions

The aim of this study is to explain behavioral portfolio theory. The main assumption of BPT is that investors do not consider portfolios as a whole which is the main proposition of MPT. Behavioral asset pricing model (BAPM) is an alternative to capital asset pricing model (CAPM), Fama-French three-factor model, and arbitrage pricing theory (APT). This study aims to draw a general framework of behavioral portfolio theory, which is one of the behavioral finance models, and to indicate the points from which it differs from modern portfolio theory. This study focuses on the teachings of traditional and behavioral finance models in portfolio creation. Thus, empirical studies based on statistical analysis can be conducted where portfolio performances configured with behavioral perspective are compared with the performances of average variance portfolios.

3.12 Conclusion

Traditional finance theories consider market participants as "rational wealth maximizers" from the dates that emerged in the 1950s. In addition, according to the assumptions that form the basis of the modern portfolio theory, investors use only two parameters in their investment decisions. These are the expected return and standard deviation (risk). When the portfolio selection is considered in a rational framework, investors are considered to make completely rational decisions. In this context, investors try to choose financial assets that maximize the value of utility functions. This selection process becomes much easier by mean-variance approach. On the other hand, behavioral finance presents that emotion and psychology affect investors and therefore market anomalies occur due to the fact that investors are limited rational or normal. The effect of behavioral factors in the decision-making process plays an important role in the accuracy of the decisions made by investors.

Behavioral portfolio theory (BPT) emerges as a descriptive alternative to Markowitz's mean-variance portfolio theory and connects two issues: the creation of portfolios and the design of securities. The basis of behavioral portfolio theory is that the psychological (Kahneman & Tversky, 1979) and emotional characteristics (Lopes, 1987) of individuals affect their decisions. This theory assumes that

investors are normal and affected by some psychological prejudices during their decision-making. Behavioral portfolio theory reveals the effect of mental accounting bias. The concept of mental accounting was first used by Thaler ([1999](#)). BPT traders are at the same time avoiding risk and seeking risk, as in the Friedman-Savage puzzle.

Behavioral portfolio theory (BPT) and behavioral asset pricing model (BAPM) fill the gaps in theory and asset pricing models. BPT and BAPM focus on mental accounting, bounded rationality, and psychological and emotional utilities. Also, BPT and BAPM bring financial theories closer to reality.

Key Terms and Definitions

Portfolio: Portfolio is a financial asset that consists of various securities such as stocks and bonds and derivative products, held by a particular person or group. Various portfolios can be created according to investors' different returns and risk preferences. The important thing here is the preferences of the investor.

Portfolio management: Portfolio management is a system of continuing and updating the evaluation of the success of investors' goals, preferences, and constraints determined and resolved under this information by monitoring the portfolio. Also, portfolio management is the management of securities according to investors' returns and risk targets. The portfolio manager tries to provide the expected return to the portfolio owner or investor with the necessary diversification at an acceptable risk level. The aim of portfolio management is to reduce the risk through diversification.

Modern portfolio theory: Modern portfolio theory is a comprehensive approach that includes issues such as indifference curves, efficient frontier, opportunity set, and optimal portfolio selection, along with the correlation relationship between the returns of securities. Modern portfolio theory mathematically shows the measurement of the risk and return of a portfolio of two or more securities and the determination of optimal portfolios.

Behavioral finance: Behavioral finance deals with investor behavior and its impact on financial markets. It contributes to investor psychology and the inclusion of behavioral characteristics of investors in asset pricing models. Behavioral finance makes explanations on shaping investor behavior and investment decisions, how basic information is interpreted, and financial factors in addition to financial factors in investor's decision.

Behavioral portfolio theory: Behavioral portfolio theory (BPT) emerged as a descriptive alternative to Markowitz's mean-variance portfolio theory. BPT connects two issues: the creation of portfolios and the design of securities. BPT is also an alternative portfolio choice model developed to better capture these different features. BPT not only integrates the mental accounting structure of Kahneman and Tversky ([1979](#)) and Tversky and Kahneman ([1992](#)) but also enables investors to view their portfolios as a collection of subportfolios, each suitable for a particular mental accounting. Das et al. ([2011](#)) combine important features of modern portfolio theory and behavioral portfolio theory and create a new mental accounting (MA) framework.

Behavioral asset pricing model: Shefrin and Statman (1994) develop behavioral asset pricing model (BAPM) for the price efficiency by the presence of noise traders and analyze the effects of noise traders on price efficiency, volatility, return anomalies, and noise trader survival. BAPM is an alternative to capital asset pricing model (CAPM), Fama-French three-factor model, and arbitrage pricing theory (APT). CAPM and efficient market hypothesis (EMH) can be useful for markets where rational investors and information traders take place.

References

- Barberis, N., & Thaler, R. H. (2003). Advances. *Handbook of the Economics of Finance, 2003*, 1052–1114.
- Barberis, N., Shleifer, A., & Vishny, R. (1998). A model of investor sentiment. *Journal of Financial Economics, 49*(3), 307–343.
- Bourachnikova, O. (2007). Weighting function in the behavioral portfolio theory. *Working papers, CEB, 7*.
- Curtis, G. (2004). Modern portfolio theory and behavioral finance. *The Journal of Wealth Management, 7*(2), 16–22.
- Daniel, K., Hirshleifer, D., & Subrahmanyam, A. (1998). Investor psychology and security market under-and overreactions. *The Journal of Finance, 53*(6), 1839–1885.
- Das, S., Markowitz, H., Scheid, J., & Statman, M. (2011). Portfolios for investors who want to reach their goals while staying on the mean–variance efficient frontier. *The Journal of Wealth Management, 14*(2), 25–31.
- Fama, E. (1970). Efficient capital markets: A review of theory and empirical work. *Journal of Finance, 25*, 386.
- Hoffmann, A. O., Shefrin, H., & Pennings, J. M. (2010). Behavioral portfolio analysis of individual investors. Available at SSRN, 2010, 1629786.
- Hong, H., & Stein, J. C. (1999). A unified theory of underreaction, momentum trading, and overreaction in asset markets. *The Journal of Finance, 54*(6), 2143–2184.
- Howard, C. T. (2014). Behavioral portfolio management. *The Journal of Behavioral Finance, 4*(1), 39–61.
- Kahneman, D., & Tversky, A. (1979). Prospect theory: An analysis of decision under risk. *Econometrica, 47*(2), 263–292.
- Kıylar, M., & Akkaya, M. (2016). *Davranışsal Finans*. İstanbul: Literatür Yayıncılık.
- Leković, M. (2019). Behavioral portfolio theory and behavioral asset pricing model as an alternative to standard finance concepts. *Ekonomski Horizonti, 21*(3), 255–279.
- Lintner, J. (1975). The valuation of risk assets and the selection of risky investments in stock portfolios and capital budgets. In *Stochastic optimization models in finance* (pp. 131–155). New York: Academic Press.
- Lopes, L. L. (1987). Between hope and fear: The psychology of risk. *Advances in Experimental Social Psychology, 20*(3), 255–295.
- Markowitz, H. (1952). Portfolio selection. *Journal of Finance, 7*(1), 77–91.
- Mossin, J. (1966). Equilibrium in a capital asset market. *Econometrica: Journal of the Econometric Society, 1966*, 768–783.
- Pfiffelmann, M., Roger, T., & Bourachnikova, O. (2016). When behavioral portfolio theory meets Markowitz theory. *Economic Modelling, 53*, 419–435.
- Roll, R. (1977). A critique of the asset pricing theory's tests Part I: On past and potential testability of the theory. *Journal of Financial Economics, 4*(2), 129–176.
- Ross, S. A. (1976). The arbitrage theory and asset pricing model. *Journal of Economic Theory, 13*(3), 343–362.

- Roy, A. D. (1952). Safety first and the holding of assets. *Econometrica: Journal of the Econometric Society*, 1952, 431–449.
- Sharpe, W. F. (1964). Capital asset prices: A theory of market equilibrium under conditions of risk. *The Journal of Finance*, 19(3), 425–442.
- Shefrin, H., & Statman, M. (2000). Behavioral portfolio theory. *Journal of Financial and Quantitative Analysis*, 35(2), 127–151.
- Shefrin, H. (2015). The behavioral paradigm shift. *Revista de Administração de Empresas*, 55(1), 95–98.
- Statman, M. (2004). Behavioral portfolios: Hope for riches and protection from poverty. Pension design and structure. *New Lessons from Behavioral Finance*, 2004, 67–80.
- Statman, M. (2008). What is behavioral finance? *Handbook of Finance*, 2, 79–84.
- Thaler, R. H. (1999). Mental accounting matters. *Journal of Behavioral Decision Making*, 12(3), 183–206.
- Tversky, A., & Kahneman, D. (1992). Advances in prospect theory: Cumulative representation of uncertainty. *Journal of Risk and Uncertainty*, 5(4), 297–323.
- Tversky, A., & Kahneman, D. (1979). Prospect theory: An analysis of decision under risk. *Econometrica*, 47(2), 263–291

Additional Readings

- Black, F., Jensen, M. C., & Scholes, M. (1972). The capital asset pricing model: Some empirical tests. *Studies in the Theory of Capital Markets*, 81(3), 79–121.
- Dayala, R. (2012). A bridge between portfolio theory & behavioral finance, the market indifference curve. *The Market Indifference Curve (March 30, 2012)*.
- De Brouwer, P. J. (2009). Maslowian portfolio theory: An alternative formulation of the behavioural portfolio theory. *Journal of Asset Management*, 9(6), 359–365.
- Fama, E. F. (1965). The behavior of stock-market prices. *The Journal of Business*, 38(1), 34–105.
- Fama, E. F., & MacBeth, J. D. (1973). Risk, return, and equilibrium: Empirical tests. *Journal of Political Economy*, 81(3), 607–636.
- Harrington, D. R. (1983). *Modern portfolio theory, the capital asset pricing model, and arbitrage pricing theory: A user's guide*. Englewood Heights, NJ: Prentice Hall.
- Kahneman, D., Slovic, S. P., Slovic, P., & Tversky, A. (Eds.). (1982). *Judgment under uncertainty: Heuristics and biases*. London: Cambridge University Press.
- Muneer, S. (2012). Materialization of behavioral finance and behavioral portfolio theory: A brief review. *Journal of Economics and Behavioral Studies*, 4(8), 431–435.
- Shefrin, H. (2016). *Behavioral risk management: Managing the psychology that drives decisions and influences operational risk*. Berlin: Springer.
- Shefrin, H., & Statman, M. (1994). Behavioral capital asset pricing theory. *Journal of Financial and Quantitative Analysis*, 29(3), 323–349.
- Statman, M. (2017). Behavioral asset pricing: Asset pricing for normal people. *The Journal of Portfolio Management*, 44(1), 5–9.

Chapter 4

A Comparative Study on PSO with Other Metaheuristic Methods



Serhat Yarat, Sibel Senan, and Zeynep Orman

Abstract The research and development of metaheuristic methods are critical issues in computer science. In the past decade, metaheuristic algorithms have been used in many engineering applications such as optimization of engineering problems, telecommunications, information security, and image processing. Many metaheuristic algorithms such as particle swarm optimization (PSO), ant colony optimization (ACO), and genetic algorithm (GA) are recently becoming very popular.

There are many studies conducted in the literature on the comparison of PSO with other metaheuristic algorithms. In this chapter, various studies carried out between the years of 2010 and 2020 about the comparison of PSO with the other metaheuristic algorithms will be examined. The metaheuristic algorithms to be considered are simulated annealing (SA), genetic algorithm (GA), differential evolution (DE), ant colony optimization (ACO), artificial bee colony (ABC) algorithm, particle swarm optimization (PSO), tabu search (TS), harmony search (HS), firefly algorithm (FF), cuckoo search (CS), bat-inspired algorithm (BA), water wave optimization (WWO), clonal selection algorithm (CLONALG), chemical reaction optimization (CRO), sine cosine algorithm (SCA), glowworm swarm optimization (GSO), and grey wolf optimizer (GWO). This study aims to evaluate and analyze the covered papers according to several criteria such as (a) rates of studies according to publishing years, (b) the metaheuristic algorithms that are compared to PSO, (c) performance evaluation of compared algorithms, (d) the metaheuristic algorithms with their inspirational approaches and their initial proposed studies and years, (e) the field of subjects where the algorithms are applied in the reviewed studies, and (f) used databases in the examined studies.

This study is a comprehensive literature review of the comparison of PSO with the most popular metaheuristic algorithms. The intention of this review is to be useful for researchers who want to conduct a survey on this area of the subject as this chapter will cover the essential and helpful analysis of the related research.

S. Yarat · S. Senan · Z. Orman (✉)
Istanbul University-Cerrahpasa, Istanbul, Turkey
e-mail: ssenan@istanbul.edu.tr; ormanz@istanbul.edu.tr

Keywords Article swarm optimization (PSO) · Metaheuristic algorithms · Heuristic algorithms Optimization algorithms · Swarm intelligence (SI)

4.1 Introduction

Optimization algorithms can be considered among the exact and heuristic algorithms. When an algorithm is exact, it is guaranteed that the optimal solution will be found in a finite amount of time for the related optimization problem. However, for NP-hard (nondeterministic polynomial-time) optimization problems, it is often the case that there are some polynomial-time approximation algorithms, but the best-known algorithms require exponential time. On the other hand, heuristic algorithms can guarantee a solution that is close to the best solution in a reasonable amount of time. Hence, heuristic algorithms are mostly used when approximate solutions are sufficient. Metaheuristics are the problem-specific implementations of the heuristic optimization algorithms according to the guidelines expressed in such a heuristic framework.

Swarm intelligence (SI) refers to the artificial or natural systems composed of many self-organized individuals that have the collective behaviors of decentralized control. Swarm intelligence, which is a subset of artificial intelligence (AI), is becoming more and more critical as more complex problems require solutions that can be achieved in a less convenient but reasonable time. Swarm intelligence algorithms can be applied for various research fields such as engineering, industries, and social sciences. Specific algorithms for SI include particle swarm optimization (PSO), ant colony optimization (ACO), and artificial bee colony (ABC) algorithms. Each of these algorithms is a metaheuristic algorithm inspired by the natural, self-organized behavior of animals.

Particle swarm optimization (PSO) is a simple and effective optimization method, which has gained interest from many researchers in different fields. PSO is inspired by the movements of some animals living as a swarm, especially when they supply the basic needs such as finding food. These animals have also impact on the others in the swarm and satisfy both their and the swarm's objectives more easily. It is an optimization algorithm developed by Kennedy and Eberhart and introduced in 1995 (Eberhart & Kennedy, 1995).

This chapter will focus on the most popular metaheuristic algorithms. We aim to conduct a survey about the comparison of PSO with the other metaheuristic algorithms. Here are the metaheuristic algorithms that are taken into account: particle swarm optimization (PSO), simulated annealing (SA), genetic algorithm (GA), differential evolution (DE), ant colony optimization (ACO), artificial bee colony algorithm (ABC), tabu search (TS), harmony search (HS), firefly algorithm (FF), cuckoo search (CS), bat-inspired algorithm (BA), water wave optimization (WWO), clonal selection algorithm (CLONALG), chemical reaction optimization (CRO), sine cosine algorithm (SCA), glowworm swarm optimization (GSO), and grey wolf optimizer (GWO). There are also some other metaheuristic algorithms that

are not covered in this study (Eberhart & Kennedy, 1995; Gavrilas, 2010; Karaboğa, 2014; Lim & Leong, 2018; Sörensen, 2013).

The sections of this chapter are organized as follows: In the Literature Review section, various studies that are carried out between the years of 2010 and 2020 related to the comparison of PSO with the other metaheuristic algorithms are discussed. In the Results section, the results of the examined studies are evaluated in a general framework perspective. Evaluation results are summarized in the Conclusions section.

4.2 Literature Review

In this literature review, various research papers have been thoroughly studied, and particle swarm optimization (PSO) is compared with other metaheuristic methods within the framework of the previous studies. PSO is a population-based metaheuristic algorithm inspired by swarm intelligence (Eberhart & Kennedy, 1995; Qiang, Chen, & Li, 2015; Wang, Yang, & Mi, 2014; Yang & Deb, 2010). PSO has been designed for continuous optimization problems (Ozcan, 2016). There are many PSO-based algorithms and also hybrid algorithms obtained by combining PSO with other algorithms.

Hasan Ozcan studied differential evolution and PSO algorithms and compared them by using various benchmark functions (Ozcan, 2016). Both of these approaches gained high performance in the optimization of a physical system which included specific goals and complex constraints. These methods were determined to be used for solving challenging engineering problems with satisfying performance and ease of programming.

Assareh et al. presented the application of PSO and GA techniques for the estimation of oil demand in Iran (Assareh, Behrang, Assari, & Ghanbarzadeh, 2010). They also referenced some studies, which were using the metaheuristic methods for oil demand prediction in literature.

Yang et al. presented a comparison study for gradient-based algorithms and nature-inspired metaheuristic algorithms (FF, PSO, ACO, BCO, BA, CS, etc.) in terms of their advantages and disadvantages (Yang, Deb, Fong, He, & Zhao, 2016).

Dwivedi and Dikshit compared continuous GA and PSO algorithm in their study (Dwivedi & Dikshit, 2013). This comparison has shown that if the initial population and the maximum number of generations remained unchanged, then the performance of PSO algorithm was better than GA. PSO algorithm converged to a global optimum at a lesser number of iterations.

Arpan Kumar Kar presented a review study including bio-inspired algorithms such as PSO, GA, ACO, ABC, CS, FF, and BA and the scope of their applications (Kar, 2016). This study examined the problem domain descriptions and potential solution objectives of each considered metaheuristic algorithm.

Jia and Lichti aimed to use the heuristic optimization methods which are SA, GA, and PSO to solve the first-order design problem for a small-volume indoor network

and compare the performances of these algorithms (Jia & Lichti, 2017). The presented results indicated that PSO and GA gained similar solutions, while SA did not guarantee an optimal solution within the limited iterations.

Selvi and Umarani discussed the fundamentals of PSO and ant colony optimization (Selvi & Umarani, 2010). By presenting both the application effectiveness and the theoretical background of these algorithms, this study has demonstrated that they were one of the most successful methods that can be used in the metaheuristic field.

Azadeh et al. presented hybrid algorithms by combining PSO, GA, and artificial immune system (AIS) with various metaheuristic algorithms for forecasting the electricity energy consumption efficiently (Azadeh, Taghipour, Asadzadeh, & Abdollahi, 2014). In their study, the considered algorithms were compared with each other, and the AIS method with the clonal selection algorithm (CLONALG) has been indicated as the preferred approach as it obtained more accurate results than the other methods.

Yaghoubi and Akrami proposed an integrated approach for solving the optimization problem for a supply chain management system (Yaghoubi & Akrami, 2019). The proposed model used the ACO and PSO algorithms, and a Taguchi experimental design method was proposed to set the parameters' values to enhance the performance of these algorithms. The comparative results of this study showed that ACO is better than PSO in terms of speed convergence rates and the number of solutions iterations.

Yang analyzed the various metaheuristic algorithms, which were GA, SA, ACO, BA, PSO, HS, and FA for solving engineering optimization problems (Yang, 2010a). This study also covered some of the classic optimization methods.

Kulkarni and Desai analyzed the performances of ABC and PSO algorithms on multidimensional optimization of benchmark functions (Kulkarni & Desai, 2016). ABC and PSO algorithms obtained similar results when the unimodal functions were used. The ABC algorithm performed better than the PSO algorithm when multimodal functions were used.

Yang and Deb used the cuckoo search (CS) algorithm to solve engineering design optimization problems, including the design of springs and welded beam structures (Yang & Deb, 2010). The results of their study claimed that CS was more efficient than PSO and GA.

Ziyad Tariq Mustafa Al-Ta'i et al. used PSO and firefly (FF) algorithms for the design of a fingerprint authentication system (Al-Ta'i & Al-Hameed, 2013). According to the comparative results of their study, PSO performed better than the other algorithms in extracting features from fingerprints.

Pal et al. conducted computational experiments by using PSO and FF algorithms to find optimal solutions of noisy nonlinear continuous mathematical models (Pal, Rai, & Singh, 2012). In their study, firefly algorithm performed better than PSO algorithm, especially for higher levels of noise.

Mishra et al. proposed a model for a classification problem by using the bat algorithm to update the weights of a functional link artificial neural network (FLANN) classifier (Mishra, Shaw, & Mishra, 2012). The proposed model was compared with FLANN and PSO-FLANN. The simulation results indicated that

the proposed classification technique is superior and faster than FLANN and PSO-FLANN.

Ezgi Deniz Ulker proposed a novel PSO-HS-based algorithm (PH) by using the significant features of PSO and HS (Ülker, 2017). The experimental results demonstrated the effectiveness of the proposed algorithm, and it was claimed that PH obtained more efficient results than PSO and HS algorithms.

Sharma et al. presented a comparative study of three metaheuristic algorithms which were GA, PSO, and HS (Lim & Leong, 2018). They drew the effect of different parameters for each algorithm and presented a comparison in terms of variations and parameters.

Unler and Murat presented a modified discrete PSO algorithm for the feature selection problem (Unler & Murat, 2010). The proposed approach was compared with the TS and scatter search algorithms over public datasets. The obtained results showed that the proposed discrete PSO algorithm performed best for all performance criteria.

Rezk et al. studied the behavior performance of the two optimization techniques which were PSO and CS for extracting the global maximum power point (MPP) from the partially shaded photovoltaic power systems (Rezk, Fathy, & Abdelaziz, 2017). The obtained results indicated that CS- and PSO-based trackers have high accuracy and stability in extracting the global MPP regardless of the global MPP located compared with the traditional algorithms.

Sheijani and Izadi estimated the required time to develop a software by using metaheuristic algorithms (Sheijani & Izadi, 2019). It has shown that the PSO algorithm was superior to solve such problems when compared to other algorithms.

Adetunji et al. proposed two swarm-based algorithms which are PSO and WOA to solve optimal placement and the sizing of distributed generation (DG) units in the quest for transmission network planning (Adetunji, Hofsajer, & Cheng, 2020). This study showed the strengths and weaknesses of PSO and WOA in the implementation of optimal sizing of the DG units in transmission networks.

Hussain et al. used the PSO algorithm to solve software clustering problems (Hussain, Khanum, Abbasi, & Javed, 2015). PSO algorithm was tested on three different software test systems, and the results were compared with GA. The simulation results showed that the PSO approach had a fast convergence when compared to GA. However, it was also stated that more studies were required to find the optimum values of the PSO parameters.

Mousavirad et al. performed a benchmark of many population-based metaheuristic optimization algorithms which were WOA, imperialist competitive algorithm (ICA), CS, FA, BA, DE, PSO, GA, GWO, COA, biogeography-based optimization (BBO), teaching-learning-based optimization (TLBO), and gravitational search algorithm (GSA) for image thresholding in high-dimensional search spaces (Mousavirad, Schaefer, & Ebrahimpour-Komleh, 2019). In their study, the results demonstrated that ICA performed better than the other algorithms.

Hussein and Mousa proposed two nature-inspired metaheuristic schedulers based on PSO and ACO to model two different scheduling algorithms for load balancing Internet of things (IoT) tasks over the fog nodes (Hussein & Mousa, 2020). In their

study, the proposed algorithms were compared with the round-robin (RR) algorithm. The evaluation results showed that the proposed ACO-based scheduler presented an improvement in the response times of IoT applications when compared to PSO-based and RR algorithms.

Gholizadeh and Barati conducted a study to analyze the computational performances of PSO, HS, and FF for size and shape optimization of truss structures by examining their convergence behaviors (Gholizadeh & Barati, 2012). The obtained results showed that FF is superior to HS and PSO.

Padma et al. used the BA for optimal power flow analysis (Padma & Shiferaw, 2019). The obtained results were compared with MANPOWER, DE, and PSO algorithms. Their study claimed that the BA provided more effective results and obtained high-quality optimization.

Adnan and Razzaque presented a comparative study of PSO and CS algorithms (Adnan & Razzaque, 2013). These algorithms were applied to problem-specific distance functions. It was claimed in this study that the CS algorithm had the same effectiveness for finding the global optimal solution as the PSO algorithm. However, CS had a better computational efficiency by using fewer function evaluations.

Adrian et al. proposed the three metaheuristic algorithms, which were GA, PSO, and ACO, to solve the construction site layout problem (Adrian, Utamima, & Wang, 2014). In their study, various results were derived: (1) the three algorithms performed equally effective to search for the optimum solution, (2) the ACO was more efficient than the other two algorithms in terms of speed, and (3) all methods performed consistently to reach solution.

Babae and Sharifian analyzed the performance of DE, GA, and PSO-based calibration of a triaxial AMR magnetometer (Babae & Sharifian, 2018). The results of their study showed that all of these algorithms could be used for calibration alternatively. However, PSO was found to be faster, and it would produce results in the least error.

Calçada et al. examined the performance of GA and PSO for estimating the parameters for a yeast growth kinetic model (Calçada, Rosa, Duarte, & Lopes, 2010). In their study, the best objective function was obtained by PSO.

Civicioglu and Besdok analyzed the algorithmic concepts of CS, PSO, DE, and ABC algorithms (Civicioglu & Besdok, 2011). Comparative results revealed that the CS algorithm's problem-solving success was very close to the DE algorithm. The performances of the CS and PSO algorithms were closer to the performance of the DE algorithm than the ABC algorithm. The CK and DE algorithms provided more robust and precise results than the PSO and ABC algorithms.

Voratas Kachitvichyanukul studied three very similar evolutionary algorithms which were GA, PSO, and DE (Kachitvichyanukul, 2012). The qualitative comparisons of these algorithms were also presented.

Kawam and Mansour proposed CS algorithm for the training of a feedforward MLP (multilayer perceptron), and the obtained results were compared with those of PSO and guaranteed convergence particle swarm optimization (GCPSO) algorithms on four benchmark problems (Kawam & Mansour, 2012).

Cellular automata (CA) is a discrete and dynamic spatiotemporal modeling system that has been used for various problems. Metaheuristic methods such as GA, GSA, and PSO have been widely applied to CA modeling systems to generate more realistic simulation patterns. Feng et al. presented a comparative study of four CA models combining logistic regression (LR) and three metaheuristic algorithms (GA, GSA, and PSO) to simulate the land-use change in the Yangtze River Delta from 2005 to 2015 (Feng, Liu, & Tong, 2018). According to this study, the PSO, GSA, and GA algorithms showed similar optimization effect on CA transition rules under the same objective function, coefficient bounds, and default settings of the control parameters.

Hammouche et al. presented a comparison of six metaheuristic techniques (GA, PSO, DE, ACO, SA, and TS) to solve the multilevel thresholding problem (Hammouche, Diaf, & Siarry, 2010). Their experiment results showed that GA, PSO, and DE are much better in terms of precision, robustness, and time convergence than ACO, SA, and TS. Also, the DE was the most efficient one with respect to the quality of the solution, and the PSO converges the most quickly.

Various methods are used to solve the process parameter design problem, including metaheuristic algorithms. Tatjana Sibalija presented a review study, including a comparison of the performances of the most used metaheuristic algorithms (GA, SA, and PSO) in the optimization of industrial processes (Sibalija, 2020). In this study, three criteria have been taken into account to reach an optimum solution: (1) accuracy of an obtained optimum, (2) a number of objective function evaluations, and (3) sensitivity in respect to the algorithm-specific parameters tuning.

Financial decisions should be given to predict the financial crisis of an organization using its historical data. Uthayakumar et al. proposed an ACO-based financial crisis prediction (FCP) model that incorporated two phases: ACO-based feature selection (ACO-FS) algorithm and ACO-based data classification (ACO-DC) algorithm (Uthayakumar, Shankar, & Lakshmanaprabu, 2018). The comparisons of the developed ACO-FS and ACO-DC methods between GA, PSO, and GWO algorithms were given in the study. The study showed that the ACO-FS method has significantly better classification accuracy than GA, PSO, and GWO-based feature selection methods and the proposed ACO-FCP model was more competitive than conventional machine learning methods.

Kotti et al. presented a comparison between the PSO and ACO to solve analog circuit sizing problems (Kotti et al., 2011). In the study, performances of these algorithms were compared over two applications. Results of the study showed that the ACO algorithm offers better results in terms of robustness whereas PSO was faster and requires fewer algorithm parameters to handle.

Security assurance is an important part of an organization or company in the information and communications technology (ICT) industry to ensure that the network is secured without fear of intrusion. Oyinloye et al. presented a comparison of BA and PSO algorithm over a designed security assurance system (Oyinloye, Thompson, Bamisile, & Alademerin, 2020). The results showed that the PSO works better in high detection rate and false alarm rate.

Kumar et al. applied CS to determine optimal coefficients of a fractional delay-infinite impulse response (FD-IIR) filter (Kumar & Rawat, 2015). The simulation results of the proposed CS-based approach were compared with GA and PSO. The performance of the CS-based FD-IIR filter was determined to be superior to that achieved by GA and PSO.

SMOTE (Synthetic Minority Over-sampling Technique) is a well-known over-sampling technique to reduce the imbalanced dataset problem. SMOTE has two important key parameters: the first one is used for controlling the amount of over-sampling, and the other is used for specifying the area of the nearest neighbors. Since there are no default values known to be best for these parameters, it is important to develop methods for obtaining optimum parameters. Li et al. analyzed the BA and PSO algorithm to optimize these parameters for imbalanced dataset problem (Li, Fong, & Zhuang, 2015). The study showed that the BA gives better results than the traditional PSO.

Nayak et al. proposed a CRO-based pi-sigma neural network (PSNN) for data classification problem (Nayak, Naik, & Behera, 2015). In the study, the performance of the proposed CRO-PSNN was analyzed over various datasets and was compared with the performances of PSNN, GA-PSNN, and PSO-PSNN. Experimental results showed that the proposed method was robust and fast and also provided better classification accuracy than the others.

Nguyen and Truong successfully applied the CSA method for the distribution network reconfiguration problem to minimize the active power loss and voltage profile enhancement of power distribution systems (Nguyen & Truong, 2015). Numerical results showed that the proposed algorithm was able to find the most suitable solutions and gave better results than PSO.

Mihai Gavrilas provided basic information on various metaheuristic optimization techniques including PSO, GA, etc., and how to apply them to common optimization problems in power systems (Gavrilas, 2016).

Radfar et al. examined the inverse radiative boundary design problem in their study (Radfara, Amirib, & Arabsolghara, 2019). The GA and PSO and its modified forms such as PSO with constriction factor (PSO-CF), PSO with repulsion factor (PSO-RF), and PSO with adaptive inertia weight (PSO-AIW) algorithms were proposed to solve the related problem. The comparative results showed that the GA was fast and required less calculations to reach the objective function than the other techniques.

Rahaman and Kule used conventional sorting-based methods with GA and PSO to find a proper assignment of different functions in the nanoscale crossbar circuit that has defective cross-points (Rahaman & Kule, 2018). The proposed GA- and PSO-based methods performed more effectively than traditional sorting-based methods.

Ramadan et al. applied the PSO to find the near-optimal solutions for the capacitor allocation problem in distribution systems for the modified IEEE 16 bus distribution system connected to wind energy generation (Ramadan, Bendary, & Nagy, 2017). PSO technique was used to obtain the near optimum solution to the

considered problem. The application of PSO had better results in terms of power losses cost and voltage profile enhancement when compared to the results of GA.

Ramarao and Chandrasekaran used GA, PSO, DE, and SCA to determine the lightning channel-base-current (CBC) function parameters for representing the measured data given in IEC standard 62,305–1: 2010 for the severe case negative subsequent return stroke (NSS) (Ramarao & Chandrasekaran, 2019). The data obtained by SCA showed better performance compared to the data obtained by GA, PSO, and DE.

The correct parameter estimation plays a vital role in the accurate modeling of the battery. In the study of Sangwan et al., the parameters of two different models (first-order RC model and second-order RC model) for a proper representation of the correct nature of the battery were optimized using GA, PSO, ASMO, and DE algorithms (Sangwan, Sharma, Kumar, & Rathore, 2016). The DE algorithm had best performance for the first-order RC model, and the ASMO performed best for the second-order RC model. Overall, it has been seen that the DE algorithm was the most robust as well as computationally cheaper than the other algorithms.

Mohamed and Abdelsalam considered the constrained multicriteria multi-cloud provider selection problem that was formulated as an integer programming model. SA, GA, and PSO algorithms were used for solving it, and they were compared with the analytic hierarchy process (AHP) (Mohamed & Abdelsalam, 2020). It was found that the algorithms achieve better solutions compared with AHP solution.

In the study of Sukumar et al., the capacity of the battery energy storage system (BESS) in kWh was handled using different metaheuristic optimization techniques (Sukumar, Marsadek, Ramasamy, & Mokhlis, 2018). The different optimization techniques such as GWO, PSO, ABC, GSA, and GA were used to solve the BESS sizing problem, and a comparison of their performances was also carried out. According to the study, the GWO produced the most optimal solution compared to other optimization techniques.

Ahmid et al. presented the performance comparison of five metaheuristic algorithms to solve a discrete optimization problem for a case of a simply supported plate subjected to biaxial loading conditions (Ahmid, Dao, & Van Ngan, 2019). The results demonstrated the outperformance of the ACO algorithm over other algorithms, which confirmed the findings of previous studies. Additionally, the TS algorithm and the discrete-PSO (DPSO) algorithm performed poorly due to their limited exploration capability. Furthermore, GA and SA exhibited a high level of reliability but showed an expensive solution cost.

Yusup et al. examined the performance of the HS algorithm for feature selection phase of classification for different applications (Yusup, Zain, & Latib, 2019). According to the study, feature selection with HS performed better than other metaheuristic algorithms such as GA and PSO.

Asghari and Navimipour presented a survey paper for metaheuristic methods such as ACO, GA, and PSO in cloud computing systems (Asghari & Navimipour, 2015). The authors provided a discussion of covered methods in terms of their advantages and disadvantages by referring to the articles they reviewed. Obtained

results of the examined papers were also given to show the performances of each method.

Khan and Sahai presented a comparison of BA, GA, and PSO algorithms for training feedforward neural networks (Khan & Sahai, 2012). The experimental results showed that the BA outperforms all other algorithms.

Diao and Shen presented some experiments for feature selection techniques and heuristic search strategies (Diao & Shen, 2012). The results showed that the performance of HS was better compared with other algorithms such as GA and PSO.

Krishnnaveni and Arumugam proposed a technique to adjust the internal components of the HS algorithm automatically (Krishnnaveni & Arumugam, 2013). According to the study's results, the proposed technique had better performance for HS compared to PSO and GA.

Ramos et al. presented two distinct comparisons for the problem of feature selection in the context of nontechnical losses characteristics (Ramos, Nunes de Souza, Falcão, & Papa, 2012). The first one was comparing OPF with SVM-RBF, SVM-no Kernel, ANN-MLP, SOM, and NN. The second one was comparing PSO, HS, and gravitational search algorithm. The authors found that the HS-OPF performed best.

Das et al. proposed the HS algorithm-based feature selection method for handwritten Bangla word recognition problem (Das, Singh, Bhowmik, Sarkar, & Nasipuri, 2016). The HS-based feature selection method obtained a high accuracy rate and high classification accuracies compared to GA and PSO.

Dalla Vedova et al. made the comparison of GA, PSO, DE, and GWO on the fault detection and identification (FDI) task for an electromechanical actuator (EMA) (Mirjalili, Mirjalili, & Lewis, 2014). According to the result of the study, the PSO was the best optimization algorithm to detect multiple failures, achieving the higher PC value (more than 90%). Also, they emphasized it must be noted that a suitable algorithm for a given problem must be chosen case by case.

In the study of Wahab et al., a set of methods including GA, ACO, PSO, DE, ABC, GSO, and CS were compared to utilize 30 benchmark functions (Wahab, Nefti-Meziani, & Atyabi, 2015). The results of the study indicated the overall advantage of differential evolution (DE).

Kulkarni and Desai examined the performances of ABC and PSO algorithms on the multidimensional optimization of benchmark functions (Kulkarni & Desai, 2016). ABC and PSO algorithms exhibited similar efficiency for unimodal functions. On the other hand, the ABC algorithm was more efficient than the PSO algorithm for multimodal functions in terms of solution quality.

López-Camacho et al. analyzed GA, DE, and PSO algorithms for solving the docking problem (García-Nieto, Nebro, & Aldana-Monte, 2015). According to the results of the study, DE obtained the best performance over other algorithms.

Bashiri and Karimi presented a comparative study between heuristic and metaheuristic algorithms for quadratic assignment problem (QAP), which is an NP-hard combinatorial optimization problem. They used 3-Opt, greedy 3-Opt, and VNZ as the heuristic methods and TS, SA, and PSO as the metaheuristic methods for the problem (Bashiri & Karimi, 2010). The study's results showed that 3-Opt as a

Table 4.1 Rates of studies according to the year of publication

Year of publication [References]	No. of publications
2010 [Bashiri and Karimi (2010), Civicioglu and Besdok (2011), Hoang et al. (2010), Selvi and Umarani (2010), Sibalića (2020), Unle and Murat (2010), Yang (2010a), Yang and Deb (2010)]	8
2011 [Kachitvichyanukul (2012), Oyinloye et al. (2020)]	2
2012 [Diao and Shen (2012), Feng et al. (2018), Gholizadeh and Barati (2012), Kawam and Mansour (2012), Mishra et al. (2012), Padma and Shiferaw (2019), Pal et al. (2012), Ramos et al. (2012)]	8
2013 [Adrian et al. (2014), Al-Ta'i and Al-Hameed (2013), Krishnaveni and Arumugam (2013)]	3
2014 [Azadeh et al. (2014), Babaee and Sharifian (2018), Radfara et al. (2019)]	3
2015 [García-Nieto et al. (2015), Hussain et al. (2015), Li et al. (2015), Nayak et al. (2015), Nguyen and Truong (2015), Wahab et al. (2015)]	6
2016 [Das et al. (2016), Gavrilas (2016), Kulkarni and Desai (2016), Kuo et al. (2016), Lim and Leong (2018), Mohamed and Abdelsalam (2020), Ozcan (2016)]	8
2017 [Jia and Lichti (2017), Medani et al. (2017), Ramarao and Chandrasekaran (2019), Rezk et al. (2017), Ülker (2017)]	5
2018 [Ahmid et al. (2019), Calçada et al. (2010), Hammouche et al. (2010), Kotti et al. (2011), Ramadan et al. (2017)]	5
2019 [Adnan and Razzaque (2013), Asghari and Navimipour (2015), Mirjalili et al. (2014), Mousavirad et al. (2019), Rahaman and Kule (2018), Sangwan et al. (2016), Sheijani and Izadi (2019), Yaghoubi and Akrami (2019), Yusup et al. (2019)]	9
2020 [Adetunji et al. (2020), Hussein and Mousa (2020), Kumar and Rawat (2015), Sukumar et al. (2018), Uthayakumar et al. (2018)]	5

Source: authors' own creation

heuristic method and TS as a metaheuristic method obtained better results than the others.

Medani et al. applied the whale optimization algorithm (WOA) for the optimal reactive power dispatch (ORPD) problem (Medani, Sayah, & Bekrar, 2017). The obtained results were compared to those of PSO, PSO with time-varying acceleration coefficients (PSO-TVAC), and some other methods. According to the results of the study, the WOA algorithm was very effective in terms of fast convergence to the global optimum.

Kuo et al. used PSO and GA for item clustering in synchronized zoning systems (Kuo, Kuo, Chen, & Zulvia, 2016). According to the study, the experimental results showed that PSO and GA performed better than the existing algorithms in the literature for the item assignment problem. Also, the results showed that PSO had better performance than GA.

Hoang et al. presented a protocol using the HS for optimization of energy consumption of the network (Hoang, Yadav, Kumar, & Panda, 2010). In the study, the comparison of HS with the low-energy adaptive clustering hierarchy

Table 4.2 Studies with the metaheuristic algorithms that are compared to PSO

Algorithm	Studies
Simulated annealing (SA)	Ahmid et al. (2019), Bashiri and Karimi (2010), Feng et al. (2018), Hammouche et al. (2010), Jia and Lichti (2017), Mohamed and Abdelsalam (2020), Sheijani and Izadi (2019), Sibalija (2020)
Genetic algorithm (GA)	Adrian et al. (2014), Ahmid et al. (2019), Asghari and Navimipour (2015), Azadeh et al. (2014), Babaee and Sharifian (2018), Bashiri and Karimi (2010), Calçada et al. (2010), Das et al. (2016), Diao and Shen (2012), Dwivedi and Dikshit (2013), Feng et al. (2018), García-Nieto et al. (2015), Hammouche et al. (2010), Hoang et al. (2010), Hussain et al. (2015), Jia and Lichti (2017), Kachitvichyanukul (2012), Khan and Sahai (2012), Krishnaveni and Arumugam (2013), Kulkarni and Desai (2016), Kumar and Rawat (2015), Kuo et al. (2016), Lim and Leong (2018), Mirjalili et al. (2014), Mohamed and Abdelsalam (2020), Mousavirad et al. (2019), Nayak et al. (2015), Radfara et al. (2019), Rahaman and Kule (2018), Ramadan et al. (2017), Ramarao and Chandrasekaran (2019), Sangwan et al. (2016), Sheijani and Izadi (2019), Sibalija (2020), Sukumar et al. (2018), Uthayakumar et al. (2018), Yang and Deb (2010), Yusup et al. (2019)
Differential evolution (DE)	Asghari and Navimipour (2015), Babaee and Sharifian (2018), Civicioglu and Besdok (2011), García-Nieto et al. (2015), Hammouche et al. (2010), Kachitvichyanukul (2012), Mirjalili et al. (2014), Mousavirad et al. (2019), Ozcan (2016), Padma and Shiferaw (2019), Ramarao and Chandrasekaran (2019), Sangwan et al. (2016)
Ant colony optimization (ACO)	Adrian et al. (2014), Ahmid et al. (2019), Asghari and Navimipour (2015), Bashiri and Karimi (2010), Hammouche et al. (2010), Hussein and Mousa (2020), Kotti et al. (2011), Selvi and Umarani (2010), Uthayakumar et al. (2018), Yaghoubi and Akrami (2019)
Artificial bee colony algorithm (ABC)	Civicioglu and Besdok (2011), Kulkarni and Desai (2016), Sukumar et al. (2018), Yang (2010a)
Tabu search (TS)	Ahmid et al. (2019), Bashiri and Karimi (2010), Hammouche et al. (2010), Unle and Murat (2010)
Harmony search (HS)	Das et al. (2016), Diao and Shen (2012), Gholizadeh and Barati (2012), Hoang et al. (2010), Krishnaveni and Arumugam (2013), Lim and Leong (2018), Ramos et al. (2012), Ülker (2017), Yusup et al. (2019)
Firefly algorithm (FF)	Al-Ta'i and Al-Hameed (2013), Gholizadeh and Barati (2012), Mousavirad et al. (2019)
Cuckoo search (CS)	Adnan and Razzaque (2013), Civicioglu and Besdok (2011), Kawam and Mansour (2012), Kumar and Rawat (2015), Mousavirad et al. (2019), Nguyen and Truong (2015), Rezk et al. (2017), Yang and Deb (2010)
Bat-inspired algorithm (BA)	Khan and Sahai (2012), Li et al. (2015), Mishra et al. (2012), Mousavirad et al. (2019), Oyinloye et al. (2020), Padma and Shiferaw (2019)
Whale optimization algorithm (WOA)	Adetunji et al. (2020), Medani et al. (2017), Mousavirad et al. (2019)

(continued)

Table 4.2 (continued)

Algorithm	Studies
Clonal selection algorithm (CLONALG)	Azadeh et al. (2014)
Sine cosine algorithm (SCA)	Ramarao and Chandrasekaran (2019)
Chemical reaction optimization (CRO)	Nayak et al. (2015)
Grey wolf optimizer (GWO)	Mirjalili et al. (2014), Mousavirad et al. (2019), Sukumar et al. (2018)
Glowworm swarm optimization (GSO)	Wahab et al. (2015)

Source: authors' own creation

(LEACH), PSO, and GA as well as the traditional K-means and fuzzy c-means (FCM) clustering algorithms showed that the HS gave better results to reduce energy consumption and to improve the network lifetime.

4.3 Results

This chapter presents a survey research for the studies carried out between the years of 2010 and 2020 on the comparison of PSO with other metaheuristic algorithms. These studies are evaluated in terms of rates of studies according to the year of publication, the metaheuristic algorithms that are compared to PSO, the performance evaluation of compared algorithms, the metaheuristic algorithms with their inspirational approaches and their initial proposed studies and years, the field of subjects where the algorithms are applied in the reviewed studies, and used databases in the examined studies.

Table 4.1 presents the rates of the reviewed studies according to their year of publication. This analysis denotes that the years 2010, 2012, 2016, and 2019 are the most intense years of the conducted studies on the comparison of PSO with other metaheuristic algorithms.

In Table 4.2, the reviewed studies are examined according to the metaheuristic algorithms that are compared to PSO. The genetic algorithm (GA) is the most used algorithm with PSO in the literature.

Table 4.3 shows the performance evaluation of compared algorithms in the examined studies. According to Table 4.3, the HS algorithm performs better than PSO and other metaheuristic algorithms in all reviewed studies except in (Gholizadeh & Barati, 2012). In general, the PSO algorithm provides better results when compared to other algorithms in the reviewed studies.

Table 4.4 illustrates the metaheuristic algorithms with their inspirational approaches and their initial proposed studies and years.

Table 4.3 Compared algorithms and their performance results

References	Algorithms compared	According to the study, which one performs better?
Ozcan (2016)	PSO vs. DE	PSO
Dwivedi and Dikshit (2013)	PSO vs. GA	PSO
Yang and Deb (2010)	PSO and GA vs. CS	CS
Yang (2010a)	PSO vs. ABC	The ABC algorithm obtained high accuracy results, whereas the PSO algorithm obtained optimal results in a shorter time
Kulkarni and Desai (2016)	PSO vs. ABC	ABC
Selvi and Umarani (2010)	PSO vs. ACO	ACO
Mishra et al. (2012)	PSO-FLANN and FLANN vs. BA	BA
Jia and Lichten (2017)	PSO vs. SA and GA	GA
Al-Ta'i and Al-Hameed (2013)	PSO vs. FF	PSO
Pal et al. (2012)	PSO vs. FF	FF
Azadeh et al. (2014)	PSO, GA, vs. AIS, CLONALG	CLONALG
Ülker (2017)	PSO-HS-based algorithm vs. PSO and HS	PSO-HS-based algorithm
Yaghoubi and Akrami (2019)	ACO vs. PSO	ACO
Unle and Murat (2010)	PSO vs. TS and scatter search algorithms	PSO
Rezk et al. (2017)	PSO vs. CS	CS
Sheijani and Izadi (2019)	PSO vs. SA and GA	PSO
Adetunji et al. (2020)	PSO vs. WOA	Both of the algorithms perform better than each other for different metrics for power system networks
Hussain et al. (2015)	PSO vs. GA	PSO
Mousavirad et al. (2019)	PSO vs. WOA, CS, FA, BA, DE, GA, GWO, ICA, BBO, TLBO, and GSA	ICA
Hussein and Mousa (2020)	PSO vs. ACO	ACO
Gholizadeh and Barati (2012)	PSO vs. HS and FF	FF
Padma and Shiferaw (2019)	BA vs. DE and PSO	BA

(continued)

Table 4.3 (continued)

References	Algorithms compared	According to the study, which one performs better?
Adnan and Razzaque (2013)	PSO vs. CS	CS has the same performance for finding the true global optimum with PSO, whereas PSO is better than CS in terms of computational performance
Adrian et al. (2014)	PSO vs. GA and ACO	ACO
Babaee and Sharifian (2018)	PSO vs. DE and GA	PSO
Calçada et al. (2010)	PSO vs. GA	PSO
Civicioglu and Besdok (2011)	PSO vs. CS, DE, and ABC	CS and DE
Kachitvichyanukul (2012)	PSO vs. GA and DE	The algorithms perform better than each other based on qualitative comparisons
Kawam and Mansour (2012)	CS vs. PSO	CS
Feng et al. (2018)	PSO vs. GA and generalized simulated annealing (GSA)	The PSO, GSA, and GA algorithms perform similar optimization results
Hammouche et al. (2010)	PSO vs. GA, DE, ACO, SA, and TS	The PSO, GA, and DE have much better performance than the other ones
Sibalija (2020)	PSO vs. GA and SA	The performances of the algorithms differ according to various criteria considered in the study
Uthayakumar et al. (2018)	PSO vs. ACO, GA, and GWO	ACO
Kotti et al. (2011)	PSO vs. ACO	ACO is better than PSO in terms of robustness, whereas PSO is better than ACO in terms of speed
Oyinloye et al. (2020)	BA vs. PSO	The comparison between BAT and PSO shows that PSO works better in accuracy and has high detection rate and false alarm rate
Kumar and Rawat (2015)	PSO vs. CS and GA	CS
Li et al. (2015)	PSO vs. BA	BA
Nayak et al. (2015)	Chemical reaction optimization (CRO) vs. GA and PSO	CRO
Nguyen and Truong (2015)	PSO vs. CS	CS
Radfara et al. (2019)	GA vs. PSO, PSO-CF, PSO-RF, and PSO-W	GA
Rahaman and Kule (2018)	PSO and GA vs. traditional sorting-based methods	PSO and GA
Ramadan et al. (2017)	PSO vs. GA	PSO

(continued)

Table 4.3 (continued)

References	Algorithms compared	According to the study, which one performs better?
Ramarao and Chandrasekaran (2019)	PSO vs. GA, DE, and sine cosine algorithm (SCA)	SCA
Sangwan et al. (2016)	PSO vs. GA, ASMO, and DE	DE
Mohamed and Abdelsalam (2020)	PSO vs. GA and SA	SA and PSO are robust; they reached almost the same best solution despite the initial solution
Sukumar et al. (2018)	PSO vs. GA, ABC, GWO, and gravitational search algorithm (GSA)	GWO
Ahmid et al. (2019)	PSO vs. GA, SA, ACO, and TS	ACO
Yusup et al. (2019)	HS vs. PSO and GA	HS
Asghari and Navimipour (2015)	PSO vs. GA, ACO, greedy algorithm	PSO
Khan and Sahai (2012)	BA vs. PSO and GA	BA
Diao and Shen (2012)	HS vs. PSO and GA	HS
Krishnaveni and Arumugam (2013)	HS vs. PSO and GA	HS
Ramos et al. (2012)	HS vs. PSO	HS
Das et al. (2016)	HS vs. PSO and GA	HS
Mirjalili et al. (2014)	PSO vs. GA, DE, and GWO	PSO
Wahab et al. (2015)	GA, PSO, CS, DE, ACO, ABC, GSO, and other evolutionary algorithms	DE, closely followed by PSO
Kulkarni and Desai (2016)	PSO vs. ABC	ABC is better than PSO in terms of optimization. On the other hand, PSO is better than ABC in terms of speed
García-Nieto et al. (2015)	PSO vs. GA and DE	DE
Bashiri and Karimi (2010)	TS vs. PSO, ACO, SA, GA	TS
Medani et al. (2017)	PSO vs. WOA	WOA
Kuo et al. (2016)	PSO, GA vs. an existing algorithm for synchronized zoning system	PSO
Hoang et al. (2010)	HS vs. PSO and GA	HS

Source: authors' own creation

Table 4.4 Metaheuristic algorithms with their inspiration sources and initial proposed studies

Algorithm	Inspiration	Study	Year
PSO	Bird flock	Eberhart and Kennedy (1995)	1995
ACO	Ant colony	Dorigo, Birattari, and Stutzle (2006)	2006
ABC	Honeybee	Basturk and Karaboga (2006)	2006
CS	Cuckoo birds	Yang and Deb (2009)	2009
BA	Bat herd	Yang (2010b)	2010
FF	Firefly	Yang (2010c)	2010
GA	Natural selection and the survival of the fittest	Holland (1975)	1975
DE	Evolutionary concept	Storn and Price (1995)	1995
TS	Tabu	Glover (1986)	1986
WWO	Shallow water wave theory	Zhang, Zhang, Zhang, and Zheng (2015)	2015
CLONALG	Clonal selection	De Castro and Von Zuben (2000)	2000
SA	Analogy with annealing in solids	Kirkpatrick, Gelatt, and Vecchi (1983)	1983
HS	The principle of improvising music	Geem, Kim, and Loganathan (2001)	2001
SCA	The cyclic form of sine and cosine trigonometric functions	Mirjalili (2016)	2016
GWO	Gray wolves	Mirjalili et al. (2014)	2014
GSO	Glowworms	Krishnanand and Ghose (2009)	2005

Source: authors' own creation

Table 4.5 shows the field of subjects where the algorithms are applied in the reviewed studies.

Table 4.6 shows the used datasets in the examined studies. It can be concluded that the UCI dataset is the most preferred dataset.

4.4 Conclusions

This chapter is a comprehensive literature review of the comparison of particle swarm optimization (PSO) with other metaheuristic algorithms for optimization. The studies carried out between the years of 2010 and 2020 on the comparison of PSO to some other metaheuristic algorithms are examined by covering some of the most known metaheuristic algorithms related to optimization. In this aspect, this study will be a handbook for researchers who want to research this subject, and it will be very beneficial in this field.

Table 4.5 Application fields of the examined references

Application field	References
Optimization in mathematical and engineering problems	Adnan and Razzaque (2013), Asghari and Navimipour (2015), Kulkarni and Desai (2016), Li et al. (2015), Nguyen and Truong (2015), Ozcan (2016), Pal et al. (2012), Rahaman and Kule (2018), Sibalija (2020), Ülker (2017), Yang (2010a), Yang and Deb (2010)
Estimation and prediction	Assareh et al. (2010), Calçada et al. (2010), Uthayakumar et al. (2018)
Modeling and design problems	Dwivedi and Dikshit (2013), Feng et al. (2018), Gholizadeh and Barati (2012), Jia and Lichti (2017), Kumar and Rawat (2015), Radfara et al. (2019), Rezk et al. (2017), Sangwan et al. (2016), Sukumar et al. (2018)
Control systems	Selvi and Umarani (2010)
Classification/clustering	Kuo et al. (2016), Mishra et al. (2012), Nayak et al. (2015)
Fingerprint authentication	Al-Ta'i and Al-Hameed (2013)
Power system design	Adetunji et al. (2020), Azadeh et al. (2014), Gavrilas (2016), Medani et al. (2017), Ramadan et al. (2017), Ramarao and Chandrasekaran (2019)
Location-routing problem	Yaghoubi and Akrami (2019)
Feature selection	Das et al. (2016), Diao and Shen (2012), Krishnaveni and Arumugam (2013), Ramos et al. (2012), Unle and Murat (2010), Yusup et al. (2019)
Scheduling problem	Sheijani and Izadi (2019)
Software architecture	Hussain et al. (2015)
Multilevel thresholding problem	Hammouche et al. (2010), Mousavirad et al. (2019)
IoT	Babaee and Sharifian (2018), Hussein and Mousa (2020)
Power flow problem	Padma and Shiferaw (2019)
Construction site layout problem	Adrian et al. (2014)
Training neural networks	Kawam and Mansour (2012), Khan and Sahai (2012)
Analog circuit sizing problems	Kotti et al. (2011)
Information security	Oyinloye et al. (2020)
Cloud computing	Ahmid et al. (2019), Mohamed and Abdelsalam (2020)
Fault detection and identification	Mirjalili et al. (2014)
Molecular flexible docking problems	García-Nieto et al. (2015)
Quadratic assignment problem (QAP)	Bashiri and Karimi (2010)
Wireless sensor networks	Hoang et al. (2010)

Source: authors' own creation

According to the results obtained from the studies in the literature, it can be mentioned that the PSO algorithm has an important impact in various application fields. PSO has supported the other methods, and by using it in combination with other methods, these studies have provided better results for solving optimization

Table 4.6 Datasets used in the examined studies

Datasets	References
The breast cancer dataset	Mishra et al. (2012)
The data collected from students at the Institute of Computer Science in Sulaymaniyah City	Al-Ta'i and Al-Hameed (2013)
The data of GDP	Azadeh et al. (2014)
The University of California (UCI) Machine Learning Repository-UCI dataset	Diao and Shen (2012), Kawam and Mansour (2012), Krishnaveni and Arumugam (2013), Unle and Murat (2010)
www.mpsplib.com	Sheijani and Izadi (2019)
The Berkeley segmentation dataset: namely, 12,003, 181,079, 175,043, 101,085, 147,091, 101,087, and 253,027	Mousavirad et al. (2019)
IOT data	Hussein and Mousa (2020)
Standard (cancer first dataset, diabetes first dataset, heart first dataset, thyroid first dataset) and e-learning datasets	Khan and Sahai (2012)
An experimental data	Calçada et al. (2010)
The Open Spatial Data Sharing Project (http://ids.ceode.ac.cn)	Feng et al. (2018)
Qualitative bankruptcy dataset, Analcat data, bankruptcy dataset, Australian Credit dataset, German Credit dataset, Polish dataset	Uthayakumar et al. (2018)
Sick euthyroid dataset	Li et al. (2015)
Monk2 dataset, Pima dataset, Bupa dataset, New Thyroid, Wine dataset, Balance dataset, Heart dataset, Hayes-Roth dataset	Nayak et al. (2015)
Two labeled datasets obtained from a Brazilian electric power company	Ramos et al. (2012)
Dataset of 1020 Bangla handwritten words	Das et al. (2016)
The PDB database	García-Nieto et al. (2015)
Datasets from QAPLIB	Bashiri and Karimi (2010)

Source: authors' own creation

problems. There are many examples where it is sometimes used in conjunction with other metaheuristic algorithms.

References

- Adetunji, K. E., Hofsajer, I., & Cheng, L. (2020). Optimal DG allocation and sizing in power system networks using swarm-based algorithms. <https://arxiv.org/abs/2002.08089>. (19 Feb).
- Adnan, M. A., & Razzaque, M. A. (2013). *A comparative study of particle swarm optimization and cuckoo search techniques through problem-specific distance function: International Conference of Information and Communication Technology*. Depok: ICoICT.

- Adrian, A. M., Utamima, A., & Wang, K.-J. (2014). A comparative study of GA, PSO, and ACO for solving construction site layout optimization. *KSCE Journal of Civil Engineering*, 19, 520–527. <https://doi.org/10.1007/s12205-013-1467-6>.
- Ahmid, A., Dao, T.-M., & Van Ngan, L. É. (2019). Comparison study of discrete optimization problem using meta-heuristic approaches: A case study. *International Journal of Industrial Engineering and Operations Management (IJIEOM)*, 1(2), 97–109.
- Al-Ta'i, Z. T. M., & Al-Hameed, O. Y. A. (2013). Comparison between PSO and firefly algorithms in fingerprint authentication. *International Journal of Engineering and Innovative Technology (IJEIT)*, 3, 1.
- Asghari, S., & Navimipour, N. J. (2015). Review and comparison of meta-heuristic algorithms for service composition in cloud computing: Majlesi. *Journal of Multimedia Processing*, 4, 4.
- Assareh, E., Behrang, M. A., Assari, M. R., & Ghanbarzadeh, A. (2010). Application of PSO (particle swarm optimization) and GA (genetic algorithm) techniques on demand estimation of oil in Iran. *Energy*, 35(12), 5223–5229. <https://doi.org/10.1016/j.energy.2010.07.043>.
- Azadeh, A., Taghipour, M., Asadzadeh, S. M., & Abdollahi, M. (2014). Artificial immune simulation for improved forecasting of electricity consumption with random variations: Journal homepage: www.elsevier.com/locate/ijepes. *Electrical Power and Energy Systems*, 55, 205–224.
- Babaei, M., & Sharifian, S. (2018). Calibration of triaxial magnetometers for IoT applications using metaheuristic methods. *4th Iranian Conference on Signal Processing and Intelligent Systems (ICSPIS)*.
- Bashiri, M., & Karimi, H. (2010). *An analytical comparison to heuristic and meta-heuristic solution methods for quadratic assignment problem: The 40th International Conference on Computers & Industrial Engineering*. New York: IEEE. <https://doi.org/10.1109/ICCIE.2010.5668262>.
- Basturk, B., & Karaboga, D. (2006). An artificial bee colony (ABC) algorithm for numeric function optimization. In *Proceedings of the IEEE swarm intelligence symposium* (pp. 12–14). New York: IEEE.
- Calçada, D., Rosa, A., Duarte, L. C., & Lopes, V. V. (2010). *Comparison of GA and PSO performance in parameter estimation of microbial growth models: A case-study using experimental data*. New York: IEEE Congress on Evolutionary Computation. <https://doi.org/10.1109/CEC.2010.5586489>.
- Civicioglu, P., & Besdok, E. (2011). A conceptual comparison of the Cuckoo-search, particle swarm optimization, differential evolution, and artificial bee colony algorithms. *Artificial Intelligence Review*, 39, 315–346. <https://doi.org/10.1007/s10462-011-9276-0>.
- Das, S., Singh, P. K., Bhowmik, S., Sarkar, R., & Nasipuri, M. (2016). A harmony search based wrapper feature selection method for holistic bangla word recognition. *Procedia Computer Science*, 89, 395–403. Twelfth International Multi-Conference on Information Processing (IMCIP).
- De Castro, L.N., & Von Zuben, F.J. (2000). The clonal selection algorithm with engineering applications. In: *GECCO 2002—Workshop Proceedings*, pp. 36–37.
- Diao, R., & Shen, Q. (2012). Feature selection with harmony search. *IEEE Transactions on Systems, Man, and Cybernetics—Part-B: Cybernetics*, 42, 6.
- Dorigo, M., Birattari, M., & Stutzle, T. (2006). Ant colony optimization. *IEEE Computational Intelligence Society*, 1, 28–39.
- Dwivedi, R., & Dikshit, O. (2013). A comparison of particle swarm optimization (PSO) and genetic algorithm (GA) in second-order design (SOD) of GPS networks. *Journal of Applied Geodesy*, 7, 135–145. <https://doi.org/10.1515/jag-0045>.
- Eberhart, R. C., & Kennedy, J. (1995). A new optimizer using particle swarm theory. In *Proceedings of the 6th international symposium on micromachine and human science, Nagoya, Japan, Mar 13–16, 1995* (pp. 39–43). New York: IEEE.
- Feng, Y., Liu, Y., & Tong, X. (2018). Comparison of metaheuristic cellular automata models: A case study of dynamic land-use simulation in the Yangtze River Delta: Journal homepage:

- www.elsevier.com/locate/ceus. *Computers, Environment and Urban Systems*, 70, 138–150. <https://doi.org/10.1016/j.compenvurbsys.2018.03.003>.
- García-Nieto, J., Nebro, A. J., & Aldana-Monte, J. F. (2015). Solving molecular flexible docking problems with metaheuristics: A comparative study-Esteban López-Camacho María Jesús García Godoy. *Applied Soft Computing*, 28, 379–393. <https://doi.org/10.1016/j.asoc.2014.10.049>.
- Gavrilas, M. (2010). *Heuristic and metaheuristic optimization techniques with application to power systems: Power system Department "Gheorghe Asachi" Technical University of Iasi 21–23 D. Mangeron Blvd., 700050, Iasi ROMANIA Conference Paper October*.
- Gavrilas, M. (2016). Heuristic and metaheuristic optimization techniques with application to power systems. In *Proceedings of the 12th WSEAS international conference on Mathematical methods and computational techniques in electrical engineering*. Athens: WSEAS.
- Geem, Z. W., Kim, J. H., & Loganathan, G. V. (2001). A new heuristic optimization algorithm: Harmony search. *Simulation*, 76(2), 60–68.
- Gholizadeh, S., & Barati, H. (2012). A comparative study of three metaheuristics for optimum design of trusses. *International Journal of Civil Engineering*, 3, 423–441.
- Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research*, 5, 533–549. [https://doi.org/10.1016/0305-0548\(86\)90048-1](https://doi.org/10.1016/0305-0548(86)90048-1).
- Hammouche, K., Diaf, M., & Siarry, P. (2010). A comparative study of various meta-heuristic techniques applied to the multi-level thresholding problem: Journal homepage: www.elsevier.com/locate/engappai. *Engineering Applications of Artificial Intelligence*, 23, 676–688.
- Hoang, D. C., Yadav, P., Kumar, R., & Panda, S. K. (2010). A robust harmony search algorithm based clustering protocol for wireless sensor networks: IEEE International Conference on Communications Workshops. New York: IEEE. <https://doi.org/10.1109/ICCW.2010.5503895>.
- Holland, J. (1975). *Adaptation in natural and artificial systems*. Ann Arbor, MI: University of Michigan Press.
- Hussain, I., Khanum, A., Abbasi, A. Q., & Javed, M. Y. (2015). A novel approach for software architecture recovery using particle swarm optimization. *The International Arab Journal of Information Technology*, 12, 1.
- Hussein, M. K., & Mousa, M. H. (2020). Efficient task offloading for IoT-based applications in fog computing using ant colony optimization. *IEEE Access*, 8, 2975741. <https://doi.org/10.1109/ACCESS.2020.2975741>.
- Jia, F., & Lichti, D. (2017). A comparison of simulated annealing, genetic algorithm and particle swarm optimization in optimal first-order design of indoor TLS networks: ISPRS annals of the photogrammetry, remote sensing and spatial information sciences, Volume IV-2/W4, ISPRS geospatial week 2017, 18–22 September, Wuhan, China.
- Kachitvichyanukul, V. (2012). Comparison of three evolutionary algorithms: GA, PSO, and DE. *Industrial Engineering & Management Systems*, 11(3), 215–223. [https://doi.org/10.7232/items.11.3.215](https://doi.org/10.7232/-items.11.3.215). ISSN 1598-7248|EISSN 2234-6473.
- Kar, A. K. (2016). Bio inspired computing—A review of algorithms and scope of applications. *Expert Systems with Applications*, 59, 20–32. <https://doi.org/10.1016/j.eswa.2016.04.018>.
- Karaboga, D. (2014). *Yapay Zekâ Optimizasyon Algoritmaları*. Ankara: Nobel Akademik Yayıncılık.
- Kawam, A. A. L., & Mansour, N. (2012). Metaheuristic optimization algorithms for training artificial neural networks. *International Journal of Computer and Information Technology*, 1, 2.
- Khan, K., & Sahai, A. (2012). A comparison of BA, GA, PSO, BP, and LM for training feed-forward neural networks in e-learning context. *I. J. Intelligent Systems and Applications*, 7, 23–29. <https://doi.org/10.5815/ijisa.07.03>. Published Online June 2012 in MECS (<http://www.mecs-press.org/>).
- Kirkpatrick, S., Gelatt, D., Jr., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 220(4598), 671–680.

- Kotti, M., Benhala, B., Fakhfakh, M., Ahaitouf, A., Benlahbib, B., Loulou, M., & Mecheqrane, A. (2011). *Comparison between PSO and ACO techniques for analog circuit performance optimization*. Conference: *The International Conference on Microelectronics (ICM)*. New York: IEEE. <https://doi.org/10.1109/ICM.2011.6177367>.
- Krishnanand, K., & Ghose, D. (2009). A glowworm swarm optimization based multi-robot system for signal source localization. *Design and Control of Intelligent Robotic Systems*, 177, 49–68.
- Krishnaveni, V., & Arumugam, G. (2013). Harmony search-based wrapper feature selection method for 1-nearest neighbour classifier. *Proc. Int. Conf. on Pattern Recognition Informatics and Mobile Engineering PRIME*, 2013, 24–29.
- Kulkarni, V. R., & Desai, V. (2016). *ABC and PSO: A comparative analysis*: IEEE International Conference on Computational Intelligence and Computing Research (ICCIC). New York: IEEE. <https://doi.org/10.1109/ICCIC.2016.7919625>.
- Kumar, M., & Rawat, T. K. (2015). Optimal fractional delay-IIR filter design using cuckoo search algorithm. *ISA Transactions*, 59, 39–54. <https://doi.org/10.1016/j.isatra.2015.08.007>.
- Kuo, R. J., Kuo, P. H., Chen, Y. R., & Zulvia, F. E. (2016). Application of metaheuristics-based clustering algorithm to item assignment in a synchronized zone order picking system. *Applied Soft Computing*, 46, 143–150. <https://doi.org/10.1016/j.asoc.2016.03.012>.
- Li, J., Fong, S., & Zhuang, Y. (2015). *Optimizing SMOTE by metaheuristics with neural network and decision tree*: 3rd International Symposium on Computational and Business Intelligence. Bali: ISCBI.
- Lim, S. M., & Leong, K. Y. (2018). *A brief survey on intelligent swarm-based algorithms for solving optimization problems*. London: IntechOpen. <https://doi.org/10.5772/intechopen.76979>.
- Medani, K. B. O., Sayah, S., & Bekrar, A. (2017). Whale optimization algorithm based optimal reactive power dispatch: A case study of the Algerian power system. *Electric Power Systems Research*, 163(Part B), 696–705. <https://doi.org/10.1016/j.epsr.2017.09.001>.
- Mirjalili, S. (2016). SCA: A sine cosine algorithm for solving optimization problems. *Knowledge-Based Systems*, 96, 809–818.
- Mirjalili, S., Mirjalili, S. M., & Lewis, A. (2014). Grey Wolf optimizer. *Advances in Engineering Software*, 69, 46–61.
- Mishra, S., Shaw, K., & Mishra, D. (2012). A new metaheuristic bat inspired classification approach for microarray data. *Procedia Technology*, 4, 802–806.
- Mohamed, A. M., & Abdelsalam, H. M. (2020). A multicriteria optimization model for cloud service provider selection in multi-cloud environments. *Software: Practice and Experience*, 50, 925–947. <https://doi.org/10.1002/spe.2803>.
- Mousavirad, S. J., Schaefer, G., & Ebrahimpour-Komleh, H. (2019). *A benchmark of population-based metaheuristic algorithms for high-dimensional multi-level image thresholding*. Conference: IEEE Congress on Evolutionary Computation (CEC). New York: IEEE. <https://doi.org/10.1109/CEC.2019.8790273>.
- Nayak, J., Naik, B., & Behera, H. S. (2015). A novel chemical reaction optimization based higher order neural network (CRO-HONN) for nonlinear classification. *Ain Shams Engineering Journal*, 6(3), 1069–1091. <https://doi.org/10.1016/j.asej.2014.12.013>.
- Nguyen, T. T., & Truong, A. V. (2015). Distribution network reconfiguration for power loss minimization and voltage profile improvement using cuckoo search algorithm. *International Journal of Electrical Power & Energy Systems*, 68, 233–242. <https://doi.org/10.1016/j.ijepes.2014.12.075>.
- Oyinloye, O. E., Thompson, A. F., Bamisile, M. O., & Alademerin, D. S. (2020). Security assurance system using bat algorithm associated with particle swarm optimization. *International Journal of Computer Science and Information Security (IJCSIS)*, 18, 3.
- Ozcan, H. (2016). Comparison of particle swarm and differential evolution optimization algorithms considering various benchmark. *Journal of Polytechnic*, 20(4), 899–905.
- Padma, K., & Shiferaw, Y. (2019). A solution to optimal power flow problem using metaheuristic bat algorithm. *National Scientific Conference on Emerging Technology (ET)*, 3(3), 87–91.

- Pal, S. K., Rai, C. S., & Singh, A. P. (2012). Comparative study of firefly algorithm and particle swarm optimization for noisy nonlinear optimization problems. *I.J. Intelligent Systems and Applications*, 10, 50–57. <https://doi.org/10.5815/ijisa.2012.10.06>.
- Qiang, Y., Chen, L., & Li, B. (2015). Ant colony optimization applied to web service compositions in cloud computing. *Computers and Electrical Engineering*, 41, 18–27. <https://doi.org/10.1016/j.compeleceng.2014.12.004>.
- Radfara, N., Amirib, H., & Arabsolghara, A. (2019). Application of metaheuristic algorithms for solving inverse radiative boundary design problems with discrete power levels. *International Journal of Thermal Sciences*, 137, 539–551. <https://doi.org/10.1016/j.ijthermalsci.2018.12.014>.
- Rahaman, H., & Kule, M. (2018). *Defect tolerant approaches for function mapping in nano-crossbar circuits: Fourth International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN)*. Bengaluru: ICRCICN.
- Ramadan, H. S., Bendary, A. F., & Nagy, S. (2017). Particle swarm optimization algorithm for capacitor allocation problem in distribution systems with wind turbine generators. *International Journal of Electrical Power & Energy Systems*, 84, 143–152. <https://doi.org/10.1016/j.ijepes.2016.04.041>.
- Ramarao, G., & Chandrasekaran, K. (2019). Representation of severe negative subsequent return stroke by optimization-based channel-base-current function parameters. *Materials Today: Proceedings*, 11(Part 3), 1079–1087. <https://doi.org/10.1016/j.matpr.2018.12.042>.
- Ramos, C. C. O., Nunes de Souza, A., Falcão, A. X., & Papa, J. P. (2012). New insights on nontechnical losses characterization through evolutionary-based feature selection. *IEEE Transactions on Power Delivery*, 27, 1.
- Rezk, H., Fathy, A., & Abdelaziz, A. Y. (2017). A comparison of different global MPPT techniques based on meta-heuristic algorithms for photovoltaic system subjected to partial shading conditions: Journal homepage: www.elsevier.com/locate/rser. *A Renewable and Sustainable Energy Reviews*, 74, 377–386.
- Sangwan, V., Sharma, A., Kumar, R., & Rathore, A. K. (2016). *Estimation of battery parameters of the equivalent circuit models using meta-heuristic techniques: 1st IEEE International Conference on Power Electronics, Intelligent Control and Energy Systems (ICPEICES)*. New York: IEEE.
- Selvi, V., & Umarani, R. (2010). Comparative analysis of ant colony and particle swarm optimization techniques. *International Journal of Computer Applications*, 5, 4.
- Sheijani, O. S., & Izadi, A. (2019). Time optimization during software implementation for timely delivery using meta-heuristic algorithms. *International Journal of Machine Learning and Computing*, 9, 5.
- Sibalija, T. (2020). Metaheuristic algorithms in industrial process optimization: Performance, comparison, and recommendations. In *Intelligent technologies and applications* (pp. 270–283). https://doi.org/10.1007/978-981-15-5232-8_24.
- Sörensen, K. (2013). Metaheuristics—the metaphor exposed. *International Transactions in Operational Research*, 22, 3–18. <https://doi.org/10.1111/itor.12001>.
- Storn, R., & Price, K. (1995). *Differential evolution-a simple and efficient adaptive scheme for global optimization over continuous spaces*, Technical Report TR-95-012. Berkeley, CA: International Computer Science Institute.
- Sukumar, S., Marsadek, M., Ramasamy, A., & Mokhlis, H. (2018). Grey Wolf optimizer based battery energy storage system sizing for economic operation of microgrid. In *IEEE International Conference on Environment and Electrical Engineering and IEEE Industrial and Commercial Power Systems Europe (EEEIC/I&CPS Europe)*. New York: IEEE. <https://doi.org/10.1109/EEEIC.2018.8494501>.
- Ülker, E. D. (2017). *A PSO/HS based algorithm for optimization tasks computing conference (18–20 July 2017, London, UK)*.
- Unle, A., & Murat, A. (2010). A discrete particle swarm optimization method for feature selection in binary classification problems: Journal homepage: www.elsevier.com/locate/eswa. *European Journal of Operational Research*, 206, 528–539.

- Uthayakumar, J., Shankar, N. M. K., & Lakshmanaprabu, S. K. (2018). Financial crisis prediction model using ant colony optimization. *International Journal of Information Management*, 50, 538–556. <https://doi.org/10.1016/j.ijinfomgt.2018.12.001>.
- Wahab, M. N. A., Nefti-Meziani, S., & Atyabi, A. (2015). A comprehensive review of swarm optimization algorithms. *PLoS One*, 10(5), e0122827. <https://doi.org/10.1371/journal.pone.0122827>.
- Wang, D., Yang, Y., & Mi, Z. (2014). A genetic -based approach to web service composition in geo distributed cloud environment. *Computers and Electrical Engineering*, 43, 129–141. <https://doi.org/10.1016/j.compeleceng.2014.10.008>.
- Yaghoubi, A., & Akrami, F. (2019). *Proposing a new model for location—routing problem of perishable raw material suppliers with using meta-heuristic algorithms: Journal homepage: www.cell.com/heliyon*.
- Yang, X. S. (2010a). *Engineering optimization: An introduction with metaheuristic applications*. New Jersey: John Wiley & Sons. <https://doi.org/10.1002/9780470640425>.
- Yang, X.-S. (2010b). A new metaheuristic bat-inspired algorithm. In *Proceedings of the workshop on nature inspired cooperative strategies for optimization (NICSO)* (pp. 65–74). Berlin: Springer.
- Yang, X.-S. (2010c). Firefly algorithm, stochastic test functions and design optimisation. *International Journal of Bio-Inspired Computation*, 2, 78–84.
- Yang, X. S., & Deb, S. (2010). Engineering optimization by cuckoo search. *International Journal of Mathematical Modelling and Numerical Optimisation*, 1(4), 330–343.
- Yang, X.-S., & Deb, S. (2009). Cuckoo search via Lévy flights. In *Proceedings of the world congress on nature & biologically inspired computing* (pp. 210–214). London: NaBIC.
- Yang, X.-S., Deb, S., Fong, S., He, X., & Zhao, Y.-X. (2016). Swarm intelligence to metaheuristics: Nature-inspired optimization algorithms. *Computer*, 49, 52–59. <https://doi.org/10.1109/MC.2016.292>.
- Yusup, N., Zain, A. M., & Latib, A. A. (2019). *A review of harmony search algorithm-based feature selection method for classification*. *Journal of Physics: Conference Series*, Volume 1192, *The 2nd International Conference on Data and Information Science 15–16 November 2018, Bandung, Indonesia*.
- Zhang, B., Zhang, M.-X., Zhang, J.-F., & Zheng, Y.-J. (2015). A water wave optimization algorithm with variable population size and comprehensive learning. In D.-S. Huang, V. Bevilacqua, & P. Premaratne (Eds.), *Intelligent computing theories and methodologies* (pp. 124–136). Cham: Springer.

Chapter 5

Mathematical Model of Particle Swarm Optimization: Numerical Optimization Problems



Ashwin A. Kadkol

Abstract The Particle Swarm Optimization (PSO) algorithm was put forth by Kennedy and Eberhart in the year 1995. It is widely known for the ease with which it can be implemented and its simple approach. It is a multi-agent parallel search metaheuristic technique aimed at global optimization for numerical optimization problems. It has roots in artificial life techniques like swarm intelligence, fish schooling, etc. This chapter aims to introduce the mathematical bases for the algorithm and illustrates a few pictorial aids to understand the technique better. It is intended to serve as an introduction to spark the interest of the reader. Readers wishing to learn more about the applications of PSO and its variants to multi-objective, constrained, dynamic optimization problems and other advanced topics are recommended to consider the various references at the end of the chapter.

Keywords Artificial Intelligence · Computational Intelligence · Swarm Intelligence · Evolutionary computation · Metaheuristics · Population heuristics · Bio-Inspired Algorithms

5.1 Introduction

Artificial Intelligence or AI is a broad umbrella term that comprises various techniques. Some of the techniques that fall within the purview of AI are Machine Learning, symbolic AI, and Computational Intelligence (CI). This is an excerpt from a very popular textbook on Artificial Intelligence (AI), (Russel & Norvig, 2018): “We define AI as the study of agents that **receive** percepts from the environment and **perform actions**. Each such agent implements a **function** that **maps percept sequences to actions**, and we cover different ways to represent these functions, such as production systems, reactive agents, real-time conditional planners, neural

A. A. Kadkol (✉)
General Electric Research, Bangalore, India

networks, and decision-theoretic systems.” In the context of this chapter, we will focus on CI. Per the IEEE Computational Intelligence Society, “it is the theory, design, application and development of biologically and linguistically motivated computational paradigms. There are three traditional pillars of CI—Neural Networks, Fuzzy Systems and Evolutionary Computation (EC)” (IEEE CIS, 2019). Within CI, here, we will focus on EC. As defined by the IEEE Computational Intelligence Society, evolutionary computation is defined as “Using the biological evolution as a source of inspiration, evolutionary computation (EC) solves optimization problems by generating, evaluating and modifying a population of possible solutions. EC includes genetic algorithms, evolutionary programming, evolution strategies, genetic programming, swarm intelligence, differential evolution, evolvable hardware, multi-objective optimization and so on” (IEEE CIS, 2019). The main EC metaphor relates the natural evolution phenomenon to a type of problem-solving (trial and error) (Eiben & Smith, 2015). A simple analogy is given in the previously mentioned book to better understand the methods under EC. Any given environment is constituted of individuals who strive for survival and reproduction. The environment in which these individuals exist determines the fitness of the individuals or the chances of survival and/or reproducing. This is compared with a stochastic trial-and-error type of problem-solving process, where a collection of possible solutions exists. How well these possible solutions solve the problem determines chances of them being retained for use as seeds for construction of other possible solutions. This analogy is further simplified in the book (Eiben & Smith, 2015) with the numerical problem (which is intended to be solved) likened to the environment, the individuals in the said environment to a possible solution for the numerical problem, and finally the individual’s fitness to the quality of the proposed solution.

Swarm Intelligence falls under EC. Per (Shi, 2006), swarm Intelligence is defined as “the collective behaviors of simple individuals, interacting with each other and their respective environment.” The elements of the swarm/population (or agents as popularly referred to) follow some simple rules (Millonas, 1994)—“proximity” (population to be capable of carrying out simple space/time computations), “quality” (population to be responsive to environmental quality factors), “diverse response” (population to not restrict its movements to very narrow channels), “stability” (population to not change its behavioral mode based on changes to its environment), and “adaptability” (population to be capable to change its behavioral mode, in case computational expense is justified). The individuals are also referred to as agents, and these agents’ behaviors are summarized well in (Zhang, Wang, & Ji, 2015) as decentralized, local, and to some extent random. The interactions between the abovementioned agents/population elements resulted in the emergence of some global “intelligent” patterns which were hitherto unknown to the individual population elements/agents. Some popular examples of Swarm Intelligence are demonstrated by colonies of ants, flocks of birds, herds of animals, growth of bacteria, swarms of bees, and schools of fish. Inspired from these emerged the Ant Colony Optimization (Dorigo, 1992), Particle Swarm Optimization (Kennedy & Eberhart, 1995), Bacterial Foraging Optimization (Muller, Airaghi, Marchetto, & Koumoutsakos, 2000; Passino, 2002), Artificial Bee Colony (Basturk & Karaboga,

2007), and Glowworm Swarm Optimization (Krishnanand & Ghose, 2005) among several others.

Another classification of the EC algorithms in literature is under metaheuristics. Heuristics are defined in (Martí, Pardalos, & Resende, 2018) succinctly as strategies that use information that is accessible and loosely applicable to control problem-solving. Metaheuristics on the other hand are higher-level algorithmic frameworks providing a set of strategies to develop heuristics (Sørensen, Sevaux, & Glover, 2018). Both these are very old techniques and are called by various names in literature. In (Sørensen et al., 2018), an insightful categorization of the techniques in metaheuristics to five periods in time (prior to 1940, 1940–1980, 1980–2000, 2000 to now, and the future) has been shown.

This chapter will cover the mathematical bases for the PSO algorithm. The next section covers background to origins of the PSO algorithm. First, any assumptions or preconditions will be called out. Following this, the algorithm structure and pseudocode will be elaborated. Also included is a pictorial representation of the algorithm to elucidate the dynamics of the particles as they move through the decision space. Explanation of parameters, initialization/selection, tuning, boundary conditions, convergence criteria, and other aspects will be discussed. This is followed by a brief discussion on topics for an advanced reader around multi-objective optimization with constraints, PSO variants, and the applicability to the portfolio optimization problem. The definitions of keywords are consolidated toward the later part of this chapter.

5.2 Background

The PSO can be called a multi-agent (or population-based) parallel search method for numerical optimization of nonlinear functions. It has a strong link to A-life (fish schooling, bird flocking) (Shi, 2006). The authors (Kennedy & Eberhart, 1995) drew inspiration from simulations of bird flocking by (Heppner & Grenander, 1990; Reynolds, 1987). The former work took inspiration from the aesthetics of birds flocking, while the latter work took inspiration by the rules that helped the behavior of grouping, regrouping, and sharing information. It is a Bio-Inspired Algorithm which draws inspiration from flocks of birds in flight or schools of fish swimming together. In the words of Kennedy and Eberhart, who invented the technique, “So why, after all, did we call our first paradigm a “particle swarm”? Well, to tell the truth, our very first programs were intended to model the coordinated movements of bird flocks and schools of fish. As the programs evolved from modeling social behavior to doing optimization problems, at some point the two-dimensional plots we used to watch the algorithms perform ceased to look much like bird flocks or fish schools and started looking more like swarms of mosquitoes. The name came as simply as that” (Eberhart, Shi, & Kennedy, 2001).

Another perspective to explain the rationale of using the term “particle swarm” is offered by (Eberhart et al., 2001; Kennedy & Eberhart, 1995) via the “five basic principles of swarm intelligence” (Millonas, 1994)—“proximity,” “quality,”

“diverse response,” “stability,” and “adaptability” as discussed in the previous section. They also debate on the word “particle” and mention that members of the population are essentially massless, volumeless mathematical abstractions which undergo movements (hence, the analogies to velocity and acceleration terms make sense).

Next, a brief look at optimization in context of numerical functions is in order. The term “optimization” is defined by (Eberhart et al., 2001) as the “process of adjusting a system to get the best possible outcome.” The same authors discuss the various spaces of optimization. The *parametric space* encompasses the allowed values of all the parameters that can be used to test the function. The *function space* is the result of some operations on the parameters in parametric space. Real-world problems have multidimensional function spaces (i.e., multi-objective optimization). One sample problem is the optimal choice of a cellular/mobile phone, for instance, where price, aesthetics, connectivity (number of SIMs), and so on being the various functions that we’d like to optimize for. The *fitness space* is single dimensional and consists of the levels of success with which some combinations of parameters can optimize the values in function space. The goal of the optimization is to find parameters that aid in maximization of fitness. If the goal is single objective, then the function and fitness spaces could be the same (Eberhart et al., 2001). For the cell phone example, the fitness is determined by the combination of criteria which are met. This fitness helps in your decision to buy the device. Of course in real-world problems like these, not only are the optima many (multi-objective); there may be multiple solutions of varying optimality to the same problem (multimodality), several constraints may exist on the values that the parameters can take, and the parameters may be many. In such cases, the fitness is addressed in terms of a landscape which is multidimensional. In case of multimodal function space, there is also the possibility of the search landing in a *local optimum*, but the *global optimum* may exist elsewhere. The parameter space is also called population space, and the function space is also called as decision space.

The PSO searches for the optimal values in the real number space (Eberhart et al., 2001). The numerical optimization function that is intended to be optimized has one or more independent variables (see Eq. (5.1) which represents the DeJong’s test function 1 (Jong, 1975) or the sphere function in two dimensions). This is illustrated by Fig. 5.1. Each combination of independent variables can be thought to represent the state (say position in an n-dimension parameter space) of a possible solution to the optimization problem. The algorithm starts with an initial set of guesses like other population heuristics—called particles (or individuals of the swarm) (see Fig. 5.2). These particles continuously tend to fly toward each other, influence each other, and are constantly attempting to tend toward a more optimal state. The function space is illustrated by Fig. 5.1. The optimality of the solution is measured by the “fitness function.” As defined in the previous passage, notice that this is a single-objective case and the function space and fitness space are the same. This function maps the points via a numerical function that takes on real numbers. Particles move with the objective to attain an optimal state in the sense of the fitness function. This search is symbiotic, cooperative in nature (Engelbrecht, 2007). The

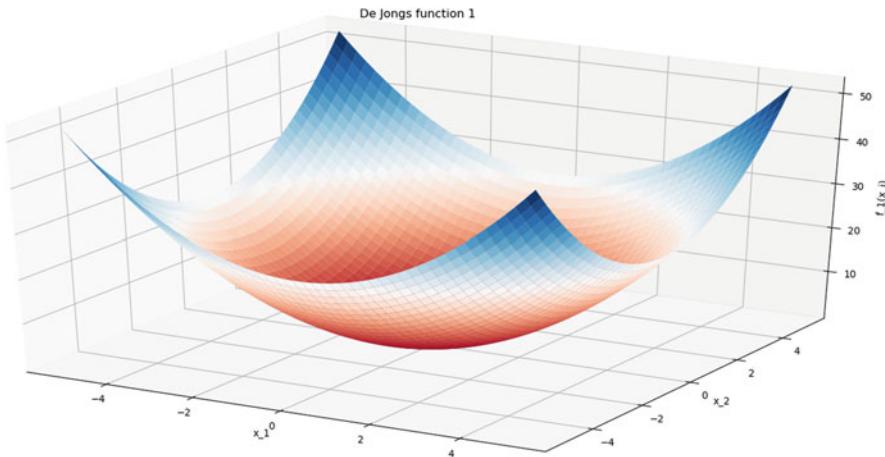


Fig. 5.1 Illustration of DeJong's function 1 (Jong, 1975), with $n = 2$ in (Eq. 5.1) above. Source: Author's own creation using Python language

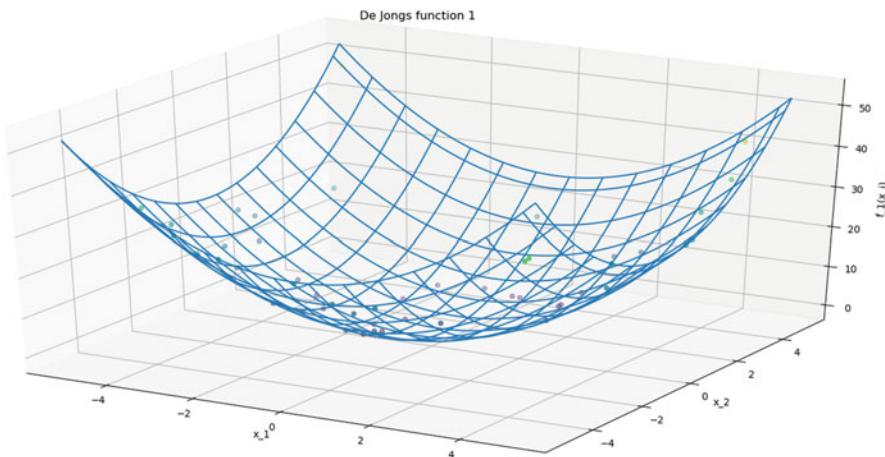


Fig. 5.2 Illustration of DeJong's function 1 with the fitness of initial set of particles in function space. Source: Author's own creation using Python language

two spaces—population space and decision space—are heterogeneous (some regions are preferred over others). The intent is to use DeJong's test function 1 (f_1) as an example to help understand the above concepts. The function is as shown in Eq. (5.1), and function space is illustrated in Fig. 5.1. The x_i constitutes the population space, and the function f_1 takes the x_i to decision space (or function space). The goal is to find the optimum (minima) of this function, i.e., $f_1 = 0$, which is evident at $\bar{x} = (x_1, x_2) = (0,0)$:

$$f_1(\vec{x}) = \sum_{i=1}^n x_i^2; -5.12 \leq x_i \leq 5.12 \quad (5.1)$$

This section covered topics around what a particle is and what the analogy of movements does mean mathematically. The assumptions and notations necessary to formally denote the particle, its position, perturbation in population space (parametric space), and corresponding movement along decision space (function space) are treated in detail in the next section.

5.3 PSO Algorithm and Mathematics

5.3.1 Key PSO Equations for Particle Motion

A particle's position can be represented by a vector $x_i^d(t)$. Here, the subscript i denotes the i th particle among " n " particles. The superscript " d " denotes the number of dimensions of the vector. As one would imagine, there are multiple variants of PSO in terms of how the particles tend toward one another in order to move toward the optimum. The "full model" is the version that the authors (Kennedy & Eberhart, 1995) originally proposed (see section below). Other variants are suggested in (Kennedy, 1997)—the "social-only" or global best model, the "cognition-only" or local best model, and finally, the "selfless" model. In the "full model," as would be evident below, every particle uses the personal best, global best, and its current position to decide where it should go next (with intent to achieve the best solution in terms of fitness). In the *social-only* model, all particles tend toward *only* the global best particle to hit the optima. This means that there is no interest toward the beliefs that lead to the individual bests of the particles. In the *cognition-only* model, each particle tends only toward its own hitherto best particle. Another variant (Engelbrecht, 2007) is where the particle tends toward its own best and an immediate neighborhood of the particle (as defined by some distance measure). A more detailed and pictorial treatment of the topic is in a subsequent subsection on neighborhood topologies (ring, wheel, and star).

5.3.1.1 Full Model of PSO

The particles' position update and transformations/perturbations/velocity updates can be represented by the Eqs. (5.2) and (5.3), respectively. These are the two equations which govern the PSO:

$$v_i^d(t+1) = \omega v_i^d(t) + c_p r_1 (\text{pbest}_i^d(t) - x_i^d(t)) + c_g r_2 (\text{gbest}^d(t) - x_i^d(t)) \quad (5.2)$$

$$x_i^d(t+1) = x_i^d(t) + v_i^d(t+1) \quad (5.3)$$

where $x_i^d(t)$ and $x_i^d(t+1)$ are the positions (along d th dimension) of the i th particle at time “ t ” and “ $t+1$,” respectively; $i = 1, 2, \dots, n$ and $d = 1, 2, \dots, M$; $v_i^d(t)$ and $v_i^d(t+1)$ are the velocities of the i th particle (along d th dimension) at time “ t ” and “ $t+1$,” respectively; $pbest_i^d(t)$ is the position of the personal best particle for the i th particle (along d th dimension) at time t ; $gbest^d(t)$ is the global best particle’s position (along d th dimension) at time t ; c_p and c_g are the personal and global best acceleration constants, respectively, which help accentuate the focus on personal best and global best, respectively; r_1 and r_2 are uniform random numbers between 0 and 1 to add a stochastic property to the flight of the particles; and ω is the inertia weight that is used to obtain a trade-off between exploration globally (toward global best) and local exploitation (toward personal best).

Notations above are adapted from (Daneshyari & Yen, 2011; Kadkol, 2010; Zhang et al., 2015).

The n particles fly through the decision space (or parametric space), and the fitness of each particle (proximity to the best objective) is measured via an objective function that maps the decision space to function space.

5.3.1.2 Pictorial Representation of Particle Movement via PSO (Full Model)

The illustration in Fig. 5.3 shows the movement of a particle through the population space (or parametric space) and the influence of the personal and global best particles on the particle.

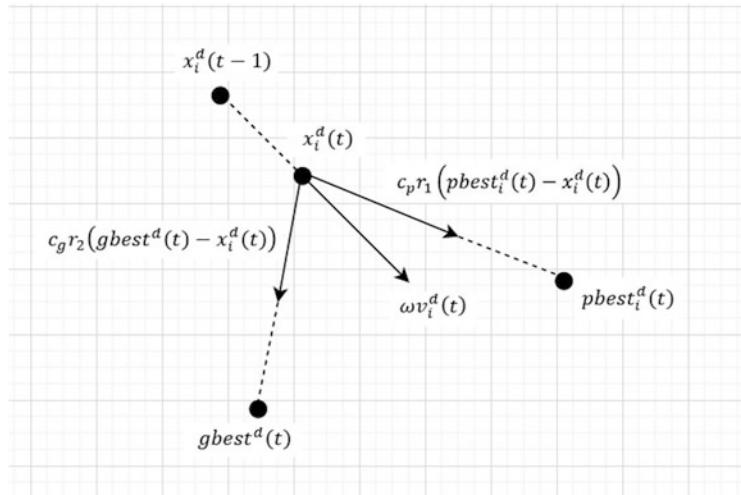


Fig. 5.3 Particle movement in PSO in the population space/parametric space. Source: Adapted from (Daneshyari & Yen, 2008, 2011) and redrawn by the author

Equations (5.2) and (5.3) have been illustrated in vector notation (for individuals seeking to implement this using a programming language/be able to understand better):

$$\begin{aligned}
 & \begin{pmatrix} v_1^1(t+1) & \cdots & v_1^d(t+1) \\ \vdots & v_2^2(t+1) & \vdots \\ v_N^1(t+1) & v_N^2(t+1) & v_N^M(t+1) \end{pmatrix} = \\
 & \omega \begin{pmatrix} v_1^1(t) & \cdots & v_1^d(t) \\ \vdots & v_2^2(t) & \vdots \\ v_N^1(t) & v_N^2(t) & v_N^M(t) \end{pmatrix} + c_p r_1 \left\{ \begin{pmatrix} \text{pbest}_1^1(t) & \cdots & \text{pbest}_1^d(t) \\ \vdots & \text{pbest}_2^2(t) & \vdots \\ \text{pbest}_N^1(t) & \text{pbest}_N^2(t) & \text{pbest}_N^M(t) \end{pmatrix} \right. \\
 & \quad \left. - \begin{pmatrix} x_1^1(t) & \cdots & x_1^d(t) \\ \vdots & x_2^2(t) & \vdots \\ x_N^1(t) & x_N^2(t) & x_N^M(t) \end{pmatrix} \right\} \\
 & + c_g r_2 \left\{ \begin{pmatrix} \text{gbest}_1^1(t) & \cdots & \text{gbest}_1^d(t) \\ \vdots & \text{gbest}_2^2(t) & \vdots \\ \text{gbest}_N^1(t) & \text{gbest}_N^2(t) & \text{gbest}_N^M(t) \end{pmatrix} \right. \\
 & \quad \left. - \begin{pmatrix} x_1^1(t) & \cdots & x_1^d(t) \\ \vdots & x_2^2(t) & \vdots \\ x_N^1(t) & x_N^2(t) & x_N^M(t) \end{pmatrix} \right\} \tag{5.4}
 \end{aligned}$$

$$\begin{aligned}
 & \begin{pmatrix} x_1^1(t+1) & \cdots & x_1^d(t+1) \\ \vdots & x_2^2(t+1) & \vdots \\ x_N^1(t+1) & x_N^2(t+1) & x_N^M(t+1) \end{pmatrix} \\
 & = \begin{pmatrix} x_1^1(t) & \cdots & x_1^d(t) \\ \vdots & x_2^2(t) & \vdots \\ x_N^1(t) & x_N^2(t) & x_N^M(t) \end{pmatrix} + \begin{pmatrix} v_1^1(t+1) & \cdots & v_1^d(t+1) \\ \vdots & v_2^2(t+1) & \vdots \\ v_N^1(t+1) & v_N^2(t+1) & v_N^M(t+1) \end{pmatrix} \tag{5.5}
 \end{aligned}$$

5.3.1.3 Variant Example: Local Best PSO

The local best version of the PSO particles has the information only of their own and nearest neighbors' best individuals (Engelbrecht, 2007). In global best PSO, particles tend toward the stochastic mean of the $\text{pbest}_i^d(t)$ and $\text{gbest}^d(t)$; here, they move toward the $\text{pbest}_i^d(t)$ and the $\text{lbest}^d(t)$ (the neighborhood best particle). The size of the neighborhood is a parameter that can be chosen. The particles' position update and transformations/perturbations/velocity updates can be represented by the Eqs. (5.6) and (5.7), respectively:

$$v_i^d(t+1) = \omega v_i^d(t) + c_p r_1 (\text{pbest}_i^d(t) - x_i^d(t)) + c_l r_2 (\text{lbest}^d(t) - x_i^d(t)) \quad (5.6)$$

$$x_i^d(t+1) = x_i^d(t) + v_i^d(t+1) \quad (5.7)$$

where $x_i^d(t)$ and $x_i^d(t+1)$ are the positions (along d th dimension) of the i th particle at time “ t ” and “ $t+1$,” respectively; $i = 1, 2, \dots, n$ and $d = 1, 2, \dots, M$; $v_i^d(t)$ and $v_i^d(t+1)$ are the velocities of the i th particle (along d th dimension) at time “ t ” and “ $t+1$,” respectively; $\text{pbest}_i^d(t)$ is the position of the personal best particle for the i th particle (along d th dimension) at time t ; $\text{lbest}^d(t)$ is the neighborhood’s best particle position (along d th dimension) at time t ; c_p and c_l are the personal and neighborhood best acceleration constants, respectively, which help accentuate the focus on personal best and local best, respectively; r_1 and r_2 are uniform random numbers (0,1) to add a stochastic property to the flight of the particles; and ω is the inertia weight that is used to obtain a trade-off between exploring/tending toward the neighborhood best and local exploitation (tending toward personal best). Notations above are adapted from (Daneshyari & Yen, 2011; Kadkol, 2010; Zhang et al., 2015).

5.3.2 Algorithm Structure and Pseudocode

This section gives a brief about the algorithm and pseudocode with the global best PSO model and is adapted from (Zhang et al., 2015). This pseudocode has been outlined with a minimization problem in perspective.

Step 1. Initialization step.

1. For every particle “ i ” in the “ n ” element population space, do the following:

- (a) Initialize particle’s position using a uniform random number such that it lies within (LB, UB), where LB and UB are the lower and upper bounds of the population space/parametric space.
- (b) Set $\text{pbest}_i^d(t)$ to initial position: $\text{pbest}_i^d(0) = x_i^d(0)$, where $x_i^d(0)$ minimizes function f .
- (c) Set gbest to the minimal value of the swarm: $\text{gbest}^d(0) = x_i^d(0)$ leading to the minimum value of fitness function (assuming it’s a minimization problem).
- (d) Initialize velocity: $v_i^d(t)$ based on a uniform random number that lies between ($-\text{range}$, range), where $\text{range} = |\text{UB} - \text{LB}|$.

Step 2. Repeat until some termination condition is met.

1. For every particle i in the n element population space, do the following:

- (a) Pick uniform random numbers: $r_1 \& r_2 \in U(0, 1)$.
- (b) Update the particle’s velocity (Eq. 5.2).
- (c) Update the particle’s position (Eq. 5.3).

- (d) If $f[x_i^d(t)] < f[\text{pbest}_i^d(t)]$, do the following:
- Update best known particle position: $\text{pbest}_i^d(t) = x_i^d(t)$.
 - If $f(x_i^d(t)) < f(\text{gbest}^d(t))$, update the swarm's best known position: $\text{gbest}^d(t) = x_i^d(t)$.
- (e) Move to the next iteration (time step), i.e., $t + 1$.
- Step 3. Output the global best solution ($\text{gbest}^d(t)$).

5.3.3 Neighborhood Topologies

The interaction between particles within the swarm is where each particle learns from others (the local best, personal best, global best). This learning is determined by how the particles are connected to each other. This can be likened to social networks (Engelbrecht, 2007). There are multiple topologies that drive the particle movements in the population space. The PSO algorithm performance is quite strongly influenced by the structure of the social network or neighborhood topology. The connectivity between particles, the grouping of particles into clusters (particles with common neighbors), and the distances between such clusters impact the PSO algorithm performance (Engelbrecht, 2007). The nature of the function space would help decide the topology. Three topologies are discussed briefly here. Figure 5.4 illustrates the particle interactions in a ring topology—local best PSO model with a particle neighborhood of two particles. This can traverse big areas of the population space, but convergence to best solution is slow (Eberhart & Kennedy, 1995; Kennedy & Eberhart, 1995).

Fig. 5.4 Ring topology (simple case of Local Best PSO model). Source: Adapted from (Shi, 2006; Talukder, 2011) and redrawn by the author

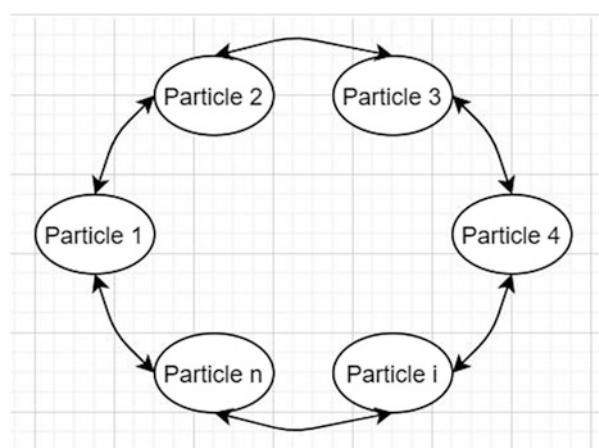


Figure 5.5 shows the particle interactions in a PSO model that resembles a governance-type model (with each particle interacting only with one particle, also called a focal). As and when this focal particle finds better solutions from across all particles, it communicates the same with the rest of the population. This type of interaction could slow down the transfer of information to the entire swarm.

Figure 5.6 illustrates the particle interactions in a star topology—global best PSO/full PSO model where every particle interacts with every other particle and hence any particle tends toward the best particle found in the entire swarm (Engelbrecht, 2007). This implementation leads to a fast convergence due to this

Fig. 5.5 Wheel topology.

Source: Adapted from (Shi, 2006; Talukder, 2011) and redrawn by the author

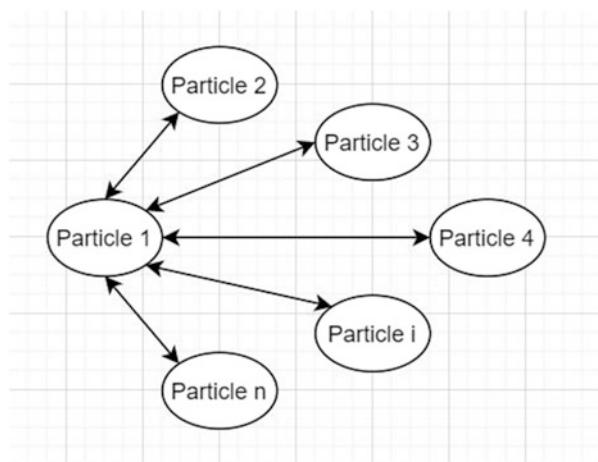
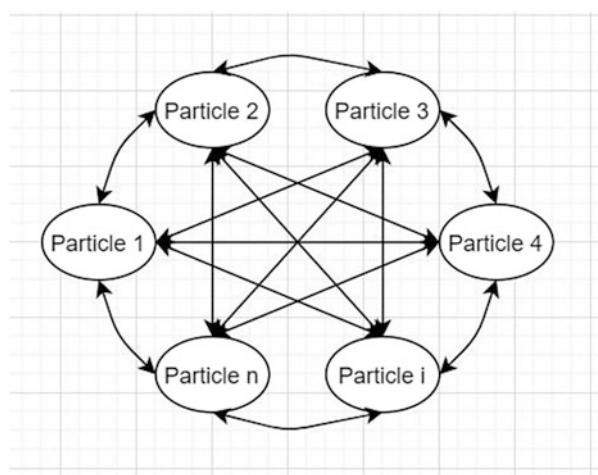


Fig. 5.6 Star topology (simple case of the global best PSO model). Source: Adapted from (Shi, 2006; Talukder, 2011) and redrawn by the author



information exchange; however, the downside is being caught in local minima as the tendency to explore is lesser. This topology works very well for unimodal functions (Engelbrecht, 2007).

Other spatial neighborhoods exist (Engelbrecht, 2007), and an example was proposed by Suganthan, where neighborhoods are established on the bases of spatial distance between particles (Suganthan, 1999) in Eqs. (5.8) and (5.9).

Two particles, l and m , are said to be in each other's neighborhood provided:

$$\frac{\|\vec{x}_l - \vec{x}_m\|}{d_{\max}} < \xi \quad (5.8)$$

where d_{\max} is the largest distance between the two particles and ξ is defined as:

$$\xi = \frac{3i + 0.6i_{\max}}{i_{\max}} \quad (5.9)$$

where i is the current iteration count and i_{\max} is the maximum number of iterations. This topology/strategy ensures smaller initial neighborhoods, and as the algorithm approaches the i_{\max} , it approaches the global best particle. The diversity and exploration are increased this way while also avoiding premature convergence (Engelbrecht, 2007).

5.4 Choice of PSO System Parameters and Tuning

5.4.1 V_{\max}

The vital PSO parameters are V_{\max} & c_g , c_p , and these are typically initialized at the start of an iteration and remain unchanged through the iteration (Shi, 2006).

$$V_{\max}.$$

The updates to the velocity are by design of the algorithm, stochastic in nature, and hence particle perturbations and the path could vary widely and could lead to an undamped oscillatory nature. To curtail this, one approach is to have a clamp as under:

- If $v_i^d(t) > V_{\max}$, then $v_i^d(t) = V_{\max}$
- Else if $v_i^d(t) < -V_{\max}$, then $v_i^d(t) = -V_{\max}$

(Shi, 2006) illustrates the need and the effects of doing this with the assumption of $d = 1$.

5.4.2 Control Parameter/Acceleration Constants (c_g , c_p , c_b , r_1 , r_2)

The acceleration constants (c_g , c_p) are also referred as trust parameters (Engelbrecht, 2007), and they determine whether particles trust themselves or their neighbors and also the rate at which the particles can tend toward the global/local/personal best particles. A small value causes the particle to follow a wide trajectory to tend toward the bests in many iterations. A large value causes the particle to follow a tighter trajectory toward the bests in fewer iterations, but the velocity explosion/clamping occurs faster. A value of zero means that the particle's trajectory is largely determined by its momentum component. The above was assuming both (c_g , c_p) are equal. If $c_g > 0$ and $c_p = 0$, all particles are tending toward the global best particle. Likewise, if $c_g = 0$ and $c_p > 0$, then all particles move on their own independently and rely on their own best values from the past. The choice of these parameters depends on the nature of the search space (unimodal, multimodal). Please refer to (Clerc & Kennedy, 2002; Engelbrecht, 2007) for a detailed treatment of the topic.

5.4.3 Inertia Weight (ω)

The inertia weight was first proposed to reduce the velocities with time (or iterations) in order to control the exploration and swarm more accurately and efficiently (Shi & Eberhart, 1999). Large inertia weights cause better exploration, while smaller weights cause a focus on a smaller region. Most often, the weight is set high initially and reduced with time (Engelbrecht, 2007).

5.4.4 Swarm Size (n)

Population size/number of particles in swarm needs to be chosen, keeping in mind certain trade-offs. Large size of swarm causes better initial diversity of particles, allowing wider search attained in lesser time (or iterations) in population space. The downsides are increased need for computation, more complexity of operations, and so on. Empirical studies recommend a swarm size between 10 and 30. Another view which is more practical is that the swarm size should be catered to the problem on hand based on cross-validation techniques. Refer to (Engelbrecht, 2007) for a more detailed treatment of the topic.

5.4.5 Neighborhood Size

This defines and sets context for the social interactions within the swarm. There is a trade-off as conceivable between small and large neighborhoods with respect to rate of convergence and optimality. Large neighborhoods have more interactions and faster convergence but are less reliable to converge to optimal solutions and more likely to get trapped in a local minimum. The reverse is true for small neighborhoods (Engelbrecht, 2007). To ensure the best of both worlds are achieved, we could start with a small neighborhood and move to larger neighborhood with time (or iterations) (Suganthan, 1999).

5.4.6 Number of Iterations

This is problem-specific. Too many iterations/epochs lead to too much computation time, and too few iterations/epochs lead to early search termination and hence inferior solutions.

5.4.7 Constriction Coefficient

(Clerc & Kennedy, 2002) came up with the constriction coefficient χ using which one could eliminate the need for the V_{\max} and ω (see Eq. 5.10 below). The proposed update to velocity update equation is:

$$v_i^d(t+1) = \chi [v_i^d(t) + c_p(pbest_i^d(t) - x_i^d(t)) + c_g r_2(gbest^d(t) - x_i^d(t))] \quad (5.10)$$

$$\text{where } \chi = \frac{2}{|4-\phi-\sqrt{\phi^2-4\phi}|} \quad \& \quad \phi = c_g + c_p.$$

This has been designed to hasten the process of convergence (Das, Abraham, & Konar, 2008). For large values of ϕ , there's a larger focus on convergence toward the personal or global bests. This is apt during exploration phase. For smaller values of ϕ , there's lesser perturbation and more apt during exploitation phase.

5.5 Initialization and Convergence Criteria

5.5.1 Initialization

Normally, the particles are initialized to cover the population space uniformly. This initial diversity of the particles influences the efficiency of the PSO (Engelbrecht,

2007). Also, depending on the nature of the function, the choice of particles decides the convergence and quality of solutions. An implementation of the algorithm with asymmetrical initialization shows how well the algorithm can escape local minima and truly explore the population space effectively.

Particle initial velocities can be set to zero in keeping with the physical analogy; however if a non-zero value must be set, it's recommended to set lower values to avoid the velocities from increasing beyond the bounds too quickly (Engelbrecht, 2007).

5.5.2 *Convergence*

PSO algorithms are executed until a convergence criterion is met. This could be a fixed number of iterations or a certain number of fitness function computations, or the optimal values found are within a tolerance value, or particle position updates don't change much across iterations, or the objective functions don't change much (Engelbrecht, 2007).

5.5.3 *Neighborhood Size*

The full model of PSO (global best-based) is the local best PSO with the entire swarm as the neighborhood. This technique is susceptible to local minima. There is a trade-off between local minima and convergence. Many neighborhoods with a smaller size ensure the avoidance of being trapped in local minima, whereas too large a neighborhood causes delayed convergence (Engelbrecht, 2007).

5.6 Applying PSO to DeJong's Test Function 1

The PSO was applied to DeJong's test function 1 (see Eq. 5.1). The population size (number of particles) was set to 50. For the sake of clearer illustration, the limits of the parametric space were set to $(-2,2)$ and the limits of the function space to $(-0.1,1)$. Figure 5.7 illustrates the function space on two separate runs. Each run constituted an initialization of the 50 particles, application of PSO iteratively and observation of function space, and convergence of solutions to minima. On run 1, Fig. 5.7a–c represent the visualization (at snapshots in time) in three dimensions of the function landscape (the horizontal axes represent the parametric values, i.e., $\vec{x} = (x_1, x_2)$, and the vertical axis represents the fitness value or function value). On run 2, Fig. 5.7d–f represent the visualization (at snapshots in time) in two

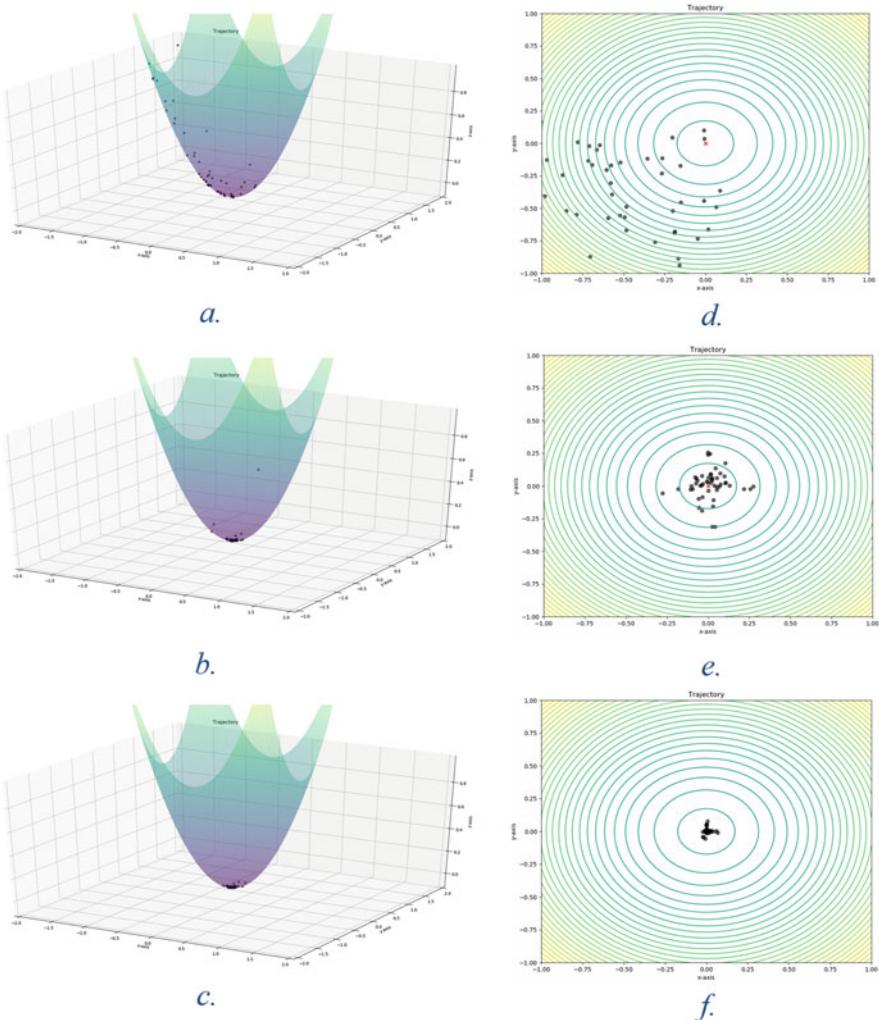


Fig. 5.7 Images (a–c) constitute snapshots taken during run 1 and are a visualization of 50 particles in function space and their trajectory as they move toward the optimum $\vec{x} = (0, 0)$. Images (d–f) constitute snapshots taken during run 2 and are a visualization of 50 particles in function space (represented via a contour plot) as they move toward the optimum (red "X" at $\vec{x} = (0, 0)$). Source: Author's own creation using the Python language package in literature (Miranda et al., 2017)

dimensions, where the two axes represent the parametric values, i.e., $\vec{x} = (x_1, x_2)$ and the function landscape is illustrated via the contours.

The function values (or fitness) of the 50 particles are represented by the black dots and the optimal value by the red cross mark. The particle fitness trajectory using PSO algorithm in the function space of DeJong's test function 1 is evident when the plots for each run are viewed in sequence (e.g., as a, b, and c). This implementation

of PSO was enabled by a Python language package in literature (Miranda, 2018; Miranda et al., 2017). An empirical study of this function and several others is available via (Shi & Eberhart, 1999).

5.7 Discussion and Further Reading

The chapter thus far has focused on the mathematical model of PSO for single-objective optimization problems, the nuances of the core algorithm, and tunable parameters. Some of the known challenges of the vanilla PSO are being trapped in local minima (Shi & Eberhart, 1999) and computational complexity. Here, we will focus on the former as the latter has been addressed by the increased compute capabilities and by parallelism of compute over the years. For the problem of being trapped in local minima or premature convergence, this can be imagined with the context of a multimodal functional space (i.e., many crests and troughs). There are some techniques to avoid this and ensure an improved trade-off between exploitation and exploration. A suggestion from literature has been to employ a self-adapting methodology for inertia weight (ω), based on the problem and the fitness landscape (Shi & Eberhart, 1999); a fuzzy local search-based method to adapt the inertia weight (ω) has been proposed (Shi & Eberhart, 2001). There are approaches around usage of neighborhood-based methods (Kennedy, 1999; Kennedy & Mendes, 2002; Mendes, Kennedy, & Neves, 2004), which in some sense delay the exploitation but enhance the exploration. A natural selection-based method was used to temper the influence of the personal and global bests on the velocity in (Angeline, 1998). A velocity-based reinitialization method has been proposed in (Binkley & Hagiwara, 2008), where the velocity is used to assess swarm stagnation to reinitialize the entire swarm population while retaining the hitherto best individuals. In (Clerc, 1999) “no-hope” convergence criterion and “re-hope” methodology is used to reinitialize the swarm on the basis of some gradient estimates of the objective function. Other variants include where entire swarm is reinitialized in dynamic environments (Eberhart & Shi, 2001) and where the global best is checked periodically to observe any changes to detect changes in the environment (Hu & Eberhart, 2002). Yet another approach is in (Clerc, 2006), where the number of iterations without any advancement in the particle’s local best is used as an indicator of stagnation to reinitialize the swarm. In (Chen, Sun, Wei, & Tang, 2011), the authors have used the sigmoid function to improve the velocity update equation inspired by techniques from Reinforcement Learning. Please see prior section on “Choice of PSO System Parameters and Tuning,” where a constriction-coefficient-based method to alter the velocity was seen (Clerc & Kennedy, 2002). In prior section on “Neighborhood Topologies,” forming neighborhoods on the bases of spatial distance between particles (Suganthan, 1999) was seen. Many more such techniques have been devised to address this, and an introduction of the trends in PSO that have been devised to overcome some of the problems, as well as to extend the scope of the type of problems the original PSO was designed to solve, is available for the

advanced reader (Zhang et al., 2015). The authors have systematically categorized the variants under the following categories:

- Modifications to the core PSO algorithm (chaotic PSO, fuzzy PSO, quantum-behaved PSO, etc.)
- Hybridization of the PSO with some other approaches like Genetic Algorithms, Artificial Immune Systems, Ant Colony Optimization, Simulated Annealing, etc.
- PSO variants by nature of problems in other fields: multi-objective, constrained, discrete, etc.
- Parameter selections and convergence analysis.
- Parallel implementations, use of compute power (GPUs).

Many real-world problems are multi-objective in nature, including the problem of **portfolio optimization** that this book addresses. Refer to (Kuo & Hong, 2013) for a treatment on the topic of fusion of Genetic Algorithms with PSO to the topic of Portfolio Optimization. A combination of categories two and three above is seen in (Daneshyari & Yen, 2008, 2011), where Cultural Algorithm-based PSO has been implemented to solve for Multi-objective Optimization problem (MOP) with constraints. In some scenarios, the MOPs are such that the objective functions change with time (dynamic MOP) (Farina, Deb, & Amato, 2004) and the PSO algorithm implementation has to factor this (Kadkol, 2010) and (Kadkol & Yen, 2012). A perspective on multi-objective optimization problem is shown below (in Eqs. 5.11 and 5.12) and is adapted from (Hu, Eberhart, & Shi, 2003; Kadkol, 2010):

$$\min_{x \in R^n} \vec{f}(\vec{x}) = [f_1(\vec{x}), f_2(\vec{x}), \dots, f_r(\vec{x})], \quad (5.11)$$

where $f_1, f_2, f_3, \dots, f_r$ are each of the r objective functions and $\vec{x} = [x_1, x_2, \dots, x_n] \in R^n$ are the n decision variables. The constraints are given by:

$$\begin{aligned} g_j(x) &\leq 0, \text{ for } j = 1, 2, \dots, m \text{ and } h_j(x) = 0, \text{ for } j = m + 1, M \\ &= 2, \dots, p. \end{aligned} \quad (5.12)$$

The bounds on the decision space (or parametric space) are $x_i^{\text{MIN}} \leq x_i \leq x_i^{\text{MAX}}$ for $i = 1, 2, \dots, n$.

The solutions to the aforesaid MOPs (fitness space) include attempting to simultaneously optimize all the r objectives (function space or objective space) in an n dimensional space (decision space or parametric space). These solutions are called as Pareto optimal solutions. The objectives in the function space have p constraints (m inequality and $p-m$ equality constraints). The concept of Pareto optimality is the key here. Some definitions are shown below in this regard as adapted from (Hu et al., 2003; Zitzler, 1999), which have a more elaborate treatment of the topic. The definitions that follow have been made from a minimization perspective:

Definition 1: Feasible Set, X_f Is the set of parametric vectors \vec{x} which satisfy constraints $g_j(\vec{x})$ and $h_j(\vec{x})$. It is defined as under:

$$X_f = \left\{ \vec{x} \in X \mid g_j(x) \leq 0, \text{ for } j = 1, 2, \dots, m \text{ and } h_j(x) = 0, \text{ for } j = m + 1, M = 2, \dots, p \right\}$$

The feasible region in the function space is denoted as $Y_f = f(X_f)$.

Definition 2: Pareto Dominance Any given parametric vector $\vec{x} = [x_1, x_2, \dots, x_n] \in R^n$ is said to dominate $\vec{x}' = [x_1, x_2, \dots, x_n] \in R^n$ strongly if and only if $\vec{f}(\vec{x}) < \vec{f}(\vec{x}')$. \vec{x} is said to be weakly dominating \vec{x}' iff.

$\vec{f}(\vec{x}) \leq \vec{f}(\vec{x}')$ and \vec{x} is said to be indifferent to \vec{x}' iff $\vec{f}(\vec{x}) \neq \vec{f}(\vec{x}') \wedge \vec{f}(\vec{x}') \neq \vec{f}(\vec{x})$.

Definition 3: Pareto Optimality/Pareto Optimal Set P^* For the said multi-objective optimization problem, a given solution $\vec{x}^* \in X_f$, (where X_f is the feasible solution space), is said to be Pareto optimal if and only if there is no $\vec{x} \in X_f$ that dominates \vec{x}^* .

Definition 4: Pareto Front (PF) The solutions which are Pareto optimal are part of the Pareto front. It can be defined as under:

$$PF = \left\{ \vec{f}(\vec{x}^*) \mid \vec{x}^* \in P^* \right\}$$

5.8 Conclusion

PSO is one of the powerful Computational Intelligence techniques that has a strong analogy to nature, is very useful in a wide variety of numerical optimization problems, and is quick to set up. It has applications in many disciplines including Portfolio Optimization which is inherently a multi-objective, dynamic optimization problem with constraints. There are many variants of PSO aimed to handle practical issues of multiple objectives and modalities, with constraints. The References section has many authors' work with variants of PSO for the single-objective case as well as the multiple-objective case with constraints and dynamic functions. This chapter has attempted to address the topic of PSO and the relatively simple

mathematics necessary to understand the technique keeping in mind a reader who is new to this area.

Key Terms and Definitions

Artificial Intelligence: “We define AI as the study of agents that **receive** percepts from the environment and **perform actions**. Each such agent implements a **function** that **maps percept sequences to actions**, and we cover different ways to represent these functions, such as production systems, reactive agents, real-time conditional planners, neural networks, and decision-theoretic systems.” (Russel & Norvig, 2018).

Computational Intelligence: “Computational Intelligence (CI) is the theory, design, application and development of biologically and linguistically motivated computational paradigms. Traditionally the three main pillars of CI have been Neural Networks, Fuzzy Systems and Evolutionary Computation.” (IEEE CIS, 2019).

Evolutionary Computation: “Using the biological evolution as a source of inspiration, evolutionary computation (EC) solves optimization problems by generating, evaluating and modifying a population of possible solutions. EC includes genetic algorithms, evolutionary programming, evolution strategies, genetic programming, swarm intelligence, differential evolution, evolvable hardware, multi-objective optimization and so on.” (IEEE CIS, 2019).

Swarm: “A swarm is a population of interacting elements that is able to optimize some global objective through collaborative search of space. Interactions that are relatively local (topologically) are often emphasized. There is a general stochastic (or chaotic) tendency in a swarm for individuals to move toward a center of mass in the population on critical dimensions, resulting in convergence to an optimum.” (Eberhart et al., 2001).

Swarm Intelligence: “The collective behaviors of simple individuals, interacting with each other and their respective environment.” (Shi, 2006).

Metaheuristics: “A metaheuristic is a high-level framework, a set of concepts and strategies that blend together, and offer a perspective on the development of optimization algorithms.” (Sørensen, Sevaux, & Glover, 2019).

Population Heuristics: Heuristic techniques that rely on multiple initial guesses/initializations as a “population” to compute function evaluations as opposed to a single guess. The heuristic algorithms have strong analogies to social systems with populations with different levels of fitness (Beasley, 2002).

Bio-Inspired Algorithms: Algorithms that fall under the Computational Intelligence category (above) and are used interchangeably.

Acknowledgments This body of work did not receive any specific grant from any funding agency in the public, commercial, or not-for-profit sectors. The author would like to acknowledge his family and GE Research for enabling this work.

References

- Angeline, P. (1998). Using selection to improve Particle Swarm Optimization. In *IEEE international conference on evolutionary computation proceedings*. Anchorage, AK, USA: IEEE.
- Basturk, B., & Karaboga, D. (2007). Artificial Bee Colony (ABC) optimization algorithm for solving constrained optimization problems. In O. C. P. Melin (Ed.), *Proceedings of the foundations of fuzzy logic and soft computing* (pp. 789–798). New York, NY, USA: Springer.
- Beasley, J. E. (2002). Population heuristics. In P. A. Pardalos (Ed.), *Handbook of applied optimization* (pp. 138–157). Oxford: Oxford University Press.
- Binkley, K. J., & Hagiwara, M. (2008). Balancing exploitation and exploration in Particle Swarm Optimization: Velocity-based reinitialization. *Transactions of the Japanese Society for Artificial Intelligence*, 23(1), 103–111.
- Chen, F., Sun, X., Wei, D., & Tang, Y. (2011). Tradeoff strategy between exploration and exploitation for PSO. In *Seventh International Conference on Natural Computation* (pp. 1216–1222). Shanghai, China: IEEE.
- Clerc, M. (1999). The swarm and the queen: Towards a deterministic and adaptive Particle Swarm Optimization. In *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99*. Washington, DC, USA: IEEE.
- Clerc, M. (2006). *Stagnation analysis in Particle Swarm Optimisation or what happens when nothing happens*. HAL archives. <https://hal.archives-ouvertes.fr/hal-00122031/document>.
- Clerc, M., & Kennedy, J. (2002). The particle swarm-explosion, stability, and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation*, 6(1), 58–73.
- Daneshyari, M., & Yen, G. (2008). Cultural MOPSO: A cultural framework to adapt parameters of multi objective Particle Swarm Optimization. In *Proceedings of IEEE Congress on Evolutionary Computation*. Hong Kong: IEEE.
- Daneshyari, M., & Yen, G. G. (2011). Cultural-based multiobjective Particle Swarm Optimization. *IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics*, 41(2), 553–567.
- Das, S., Abraham, A., & Konar, A. (2008). *Particle Swarm Optimization and differential evolution algorithms: Technical analysis, applications and hybridization perspectives*. Berlin, Heidelberg: Springer-Verlag. Retrieved from <http://www.springerlink.com/>.
- Dorigo, M. (1992). *Optimization, learning and natural algorithms*. Milano, Italy: Politecnico di Milano.
- Eberhart, R., & Shi, Y. (2001). Tracking and optimizing dynamic systems with particle swarms. In *Proceedings of the 2001 congress on evolutionary computation*. Nagoya, Japan: IEEE.
- Eberhart, R., Shi, Y., & Kennedy, J. (2001). *Swarm Intelligence*. Morgan: Kaufmann.
- Eberhart, R. C., & Kennedy, J. (1995). A new optimizer using particle swarm theory. In *Proceedings of the sixth international symposium on micromachine and human science* (pp. 39–43). Nagoya, Japan: IEEE.
- Eiben, A., & Smith, J. (2015). In T. B. G. Rozenberg (Ed.), *Introduction to evolutionary computing* (Natural Computing Series) (2nd ed.). London: Springer-Verlag GmbH.
- Engelbrecht, A. P. (2007). *Computational Intelligence: An introduction*. New York: Wiley Publishing.
- Farina, M., Deb, K., & Amato, P. (2004). Dynamic multiobjective optimization problems: Test cases, approximations, and applications. *IEEE Transactions on Evolutionary Computation*, 8 (5), 425–442. <https://doi.org/10.1109/TEVC.2004.831456>.
- Heppner, F., & Grenander, U. (1990). In S. Krasner (Ed.), *A stochastic nonlinear model for coordinated bird flocks*. Washington, DC: AAAS Publications.
- Hu, X., & Eberhart, R. (2002). Adaptive Particle Swarm Optimization: Detection and response to dynamic systems. In *Proceedings of the 2002 congress on evolutionary computation, Vol 2*. New York: IEEE.

- Hu, X., Eberhart, R. C., & Shi, Y. (2003). Particle swarm with extended memory for multiobjective optimization. In *Proceedings of the 2003 IEEE Swarm Intelligence Symposium* (pp. 193–197). Indianapolis: IEEE.
- IEEE CIS. (2019). *IEEE Computational Intelligence Society*. Retrieved from <https://cis.ieee.org/about/what-is-ci>.
- Jong, K. A. (1975). *Analysis of the behavior of a class of Genetic Adaptive Systems*. Michigan, USA: PhD thesis submitted to the University of Michigan, College of Literature, Science and the Arts.
- Kadkol, A., & Yen, G. G. (2012). A culture-based Particle Swarm Optimization framework for dynamic, constrained multi-objective optimization. *International Journal of Swarm Intelligence Research*, 3(1), 1–29. <https://doi.org/10.4018/jcir.2012010101>.
- Kadkol, A. A. (2010, July). *Culture based Particle Swarm Optimization framework for constrained dynamic multiple objective optimization*. Master of Science Thesis. Stillwater, OK: Oklahoma State University. <https://shareok.org/>
- Kennedy, J. (1997). The particle swarm: Social adaptation of knowledge. In *Proceedings of 1997 IEEE International Conference on Evolutionary Computation (ICEC)* (pp. 303–308). Indianapolis, IN, USA: IEEE.
- Kennedy, J. (1999). Small worlds and mega-minds: Effects of neighborhood topology on particle swarm performance. In *Proceedings of the 1999 Congress on Evolutionary Computation, Vol. 2*. New York: IEEE.
- Kennedy, J., & Eberhart, R. C. (1995). Particle Swarm Optimization. In *Proceedings of IEEE International Conference on Neural Networks* (pp. 1942–1948). Piscataway, NJ: IEEE.
- Kennedy, J., & Mendes, R. (2002). Population structure and particle swarm performance. In *Proceedings of the 2002 congress on evolutionary computation—Vol. 2*. New York: IEEE.
- Krishnanand, K., & Ghose, D. (2005, June). Detection of multiple source locations using a glowworm metaphor with applications to collective robotics. In *Proceedings of the IEEE Swarm Intelligence Symposium (SIS '05)* (pp. 84–94). New York: IEEE.
- Kuo, R., & Hong, C. (2013). Integration of genetic algorithm and Particle Swarm Optimization for investment Portfolio Optimization. *Applied Mathematics & Information Sciences*, 2013, 2397–2408.
- Martí, R., Pardalos, P. M., & Resende, M. G. (2018). *Handbook of heuristics*. New York: Springer International Publishing.
- Mendes, R., Kennedy, J., & Neves, J. (2004). The fully informed particle swarm: Simpler, maybe better. *IEEE Transactions on Evolutionary Computation*, 8(3), 204–210.
- Millonas, M. (1994). In C. Langton (Ed.), *Swarms, phase transitions, and collective intelligence (Vol. III)*. Reading, MA, USA: Addison-Wesley.
- Miranda, L. J. (2018). PySwarms: A research toolkit for Particle Swarm Optimization in python. *Journal of Open Source Software*, 3(21), 433. <https://doi.org/10.21105/joss.00433>.
- Miranda, L. J., Moser, A., Cronin, S. K., Carl-K, A. J., Papadimitriou, C., Mamady Nabé, E. J., & Bradahoward, T. (2017). <https://pyswarms.readthedocs.io/en/latest/examples/tutorials/visualization.html>. L. J. Miranda (ed.) Retrieved from pyswarms.readthedocs.io: <https://pyswarms.readthedocs.io/en/latest/intro.html>.
- Muller, S., Airaghi, S., Marchetto, J., & Koumoutsakos, P. (2000). Optimization algorithms based on a model of bacterial chemotaxis. *Proceedings of the 6th International Conference on Simulation of Adaptive Behavior: From Animals to Animals*, 375–384.
- Passino, K. (2002). Biomimicry of bacterial foraging for distributed optimization and control. *IEEE Control Systems Magazine*, 22(3), 52–67.
- Reynolds, C. (1987). Flocks, herds and schools: A distributed behavioral model. *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, (pp. 25–34).
- Russel, S., & Norvig, P. (2018). *Artificial Intelligence, a modern approach*. London: Pearson.
- Shi, Y. (2006). *Swarm Intelligence*. Retrieved from <http://www.swarmintelligence.org/tutorials.php>.

- Shi, Y., & Eberhart, R. (1999). Empirical study of Particle Swarm Optimization. In *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99* (pp. 1945–1950). Washington, DC, USA: IEEE.
- Shi, Y., & Eberhart, R. C. (2001). Fuzzy adaptive Particle Swarm Optimization. In *Proceedings of the congress on evolutionary computation*. Seoul, South Korea: IEEE.
- Sörensen, K., Sevaux, M., & Glover, F. (2018). A history of metaheuristics. In P. P. Martí (Ed.), *Handbook of heuristics* (pp. 791–808). Cham: Springer. https://doi.org/10.1007/978-3-319-07124-4_4.
- Sörensen, K., Sevaux, M., & Glover, F. (2019). A history of metaheuristics. In P. P. Martí (Ed.), *Handbook of heuristics*. New York: Springer.
- Suganthan, P. (1999). Particle swarm optimizer with neighborhood operator. In *Proceedings of the IEEE congress on evolutionary computation* (pp. 1958–1961). New York: IEEE.
- Talukder, S. (2011). Mathematical modelling and applications of Particle Swarm Optimization. In *Mathematical modelling and applications of Particle Swarm Optimization*. Karlskrona, Sweden: Thesis submitted to Department of Mathematics and Science, BTH.
- Zhang, Y., Wang, S., & Ji, G. (2015). A comprehensive survey on Particle Swarm Optimization algorithm and its applications. *Mathematical Problems in Engineering*.
- Zitzler, E. (1999). *Evolutionary algorithms for multiobjective optimization: Methods and applications*. PhD Thesis. Zurich: Swiss Federal Institute of Technology Zurich.

Chapter 6

Particle Swarm Optimization: The Foundation



Dadabada Pradeep Kumar

Abstract Particle swarm optimization (PSO) is a very much popular swarm intelligence algorithm. Since its inception in the year 1995, it is being applied to solve optimization problems in many domains, including portfolio optimization. This chapter lays the basic PSO foundation and introduces existing PSO variants for researchers who want to solve the portfolio optimization problem. It starts with the introduction of PSO, describing the advantages, disadvantages, and applied areas of PSO. Later, the basic PSO procedure and its parameter selection mechanisms are presented. The chapter also presents three popular applications of PSO in finance, including portfolio optimization. Finally, the chapter ends by introducing the existing PSO variants to solve the portfolio optimization problem.

Keywords Portfolio optimization · PSO algorithm · Applications · Fitness · Position update · Velocity update · Swarm intelligence

6.1 Introduction

Optimization aims at obtaining optimal solutions to a problem from a set of feasible solutions based on one or several criteria. Optimization techniques cover large application areas in business, finance, service, industry, engineering, and computer science. For example, *portfolio optimization* is an optimization problem to select the best portfolio (asset distribution) with the objectives of maximizing factors such as expected return and minimizing costs like financial risk. Constraints, if any, can help in reducing the search space of feasible solutions. The *global optimal solution*, if possibly found, can be the best solution to the problem. However, sometimes, suboptimal solutions can also be considered the possible optimal solutions to the problem (Parsopoulos & Vrahatis, 2010).

D. P. Kumar (✉)

Area of Information Systems and Analytics, Indian Institute of Management Shillong, Shillong, Meghalaya, India

Swarm intelligence (SI) is a distributed, intelligent computing mechanism for solving optimization problems. SI took its inspiration from the flocking of birds, swarming, and herding phenomenon invertebrates. Sometimes SI is considered a part of evolutionary computing, as it shares many similarities with it. SI starts working with individuals, where each individual tries to find out the optimal solution. The solution is shared among individuals, and then each individual improves themselves based on the information gathered from others. The most crucial SI property is that all the individuals work in a coordinated way without a coordinator's presence.

PSO is a population-based SI algorithm developed by Eberhart and Kennedy in 1995, inspired by the social behavior of bird flocking or fish schooling (Eberhart & Kennedy, 1995). From its inception, it is attracting a lot of researchers to solve optimization problems in different domains. In the beginning, PSO can only handle real-valued problems. Later, it has been extended to cover both binary and discrete problems (Eberhart & Shi, 2004).

PSO is a meta-heuristic algorithm that deals with a population of random solutions (particles). Each particle in PSO flies through the search space with a dynamically adjusted velocity and positions according to its own and its companion's historical behaviors. The particles move to optimal positions based on objective functions.

PSO is the most popular algorithm in comparison with other evolutionary algorithms (AlRashidi & El-Hawary, 2008; Eberhart & Shi, 2004; Pradeepkumar & Ravi, 2014, 2017; Ravi, Pradeepkumar, & Deb, 2017) as it is:

1. Very intuitive and flexible.
2. Less sensitive to the nature of the objective function.
3. Able to handle objective functions with stochastic nature.
4. Derivative-free.
5. Easy to comprehend and implement.
6. With the requirement of fewer user-defined parameters to tweak.
7. Without the requirement of a good initial solution to start its iteration process.

However, PSO also has disadvantages. These include:

- It does not always guarantee the optimal solution to the problem than the dynamic programming approach; instead, it results in a near-optimal solution.
- It is slow to convergence in the refined search stage (weak local searchability).

As it is advantageous to apply, PSO is used in various domains involving optimization problems such as antennas, biomedicine, communication networks, clustering and classification, combinatorial optimization, control, design, distribution networks, electronics and electromagnetics, engines and motors, entertainment, fault diagnosis and recovery, finance, fuzzy and neuro-fuzzy systems, graphics and visualization, image and video analysis, metallurgy, modelling, neural networks, prediction and forecasting, power systems and plants, robotics, scheduling, security and military, sensor networks, and signal processing (AlRashidi & El-Hawary, 2008; Poli, 2008; Poli, Kennedy, & Blackwell, 2007; Pradeepkumar & Ravi, 2018).

6.2 Background

The particle swarm concept originated with the effort of Reeves (1983), who came up with the idea of particles. These particles are considered as independent entities that work in harmony to achieve the objective. Reynolds (1987) then added a concept of communication between the particles' social behavior, with the help of a flocking algorithm, whereby each particle adheres to the flocking rules. Later, Nowak, Szamrej, and Latané (1990) also helped us understand the principles underlying how particles are affected by the social environment. In addition to this, Heppner and Grenander (1990) related a roost concept, i.e., the flock aims for some roosting area. In these systems, the particles are autonomous, but a class of rules regulates their movements. These observations on collective behaviors in these social animals led to implementing this model to solve different optimization problems.

6.3 The Basic PSO Algorithm

The PSO technique encompasses the following features. PSO is a metaheuristic because it makes almost nil or very few inferences about the optimization problem. It can search for vast space with distributed candidate solutions. PSO exhibits SI in its optimization process. It mainly follows five fundamental principles observed in SI-based algorithms. Mark Millonas (1993) has stated these principles are followed by the particles while communicating with other fellow particles in the swarm.

In the procedure of basic PSO and its variants, a population of particles in the n -dimensional search space gets initialized randomly. Each particle represents a possible solution. Let $X_i = X_{i,1}, X_{i,2}, \dots, X_{i,d}, \dots, X_{N,p}$ be a vector denoting the position and $V_i = V_{i,1}, V_{i,2}, \dots, V_{i,d}, \dots, V_{N,p}$ be a vector denoting velocity of particle i . A particle's position and velocity can be updated dynamically until optimal values are obtained. The basic PSO procedure is depicted by a flowchart (see Fig. 6.1) and described in Algorithm 6.1, and the notations used in the algorithm are presented in Table 6.1.

It is worth noting that the updated equations, Eqs (1) and (2), are stochastic. As the velocities are getting updated dynamically, they may become too high, leading particles to become uncontrolled. Therefore, the V_{max} (Eberhart, Shi, & Kennedy, 2001), as in Eq. (6.3), helps in restricting the uncontrolled movement of particles in search space.

A parameter, namely, inertia weight (G) as in Eq. (6.4) (Shi & Eberhart, 1998a, 1999), helps in adjusting the trade-off between explorative and exploitative capabilities of PSO. The lesser the inertia weight is, the more the PSO's exploration capability will be and vice versa. And also, Clerc and Kennedy (2002) introduced constriction factor γ , as in Eq. (6.5), which ensured convergence and improved the convergence rate of PSO.

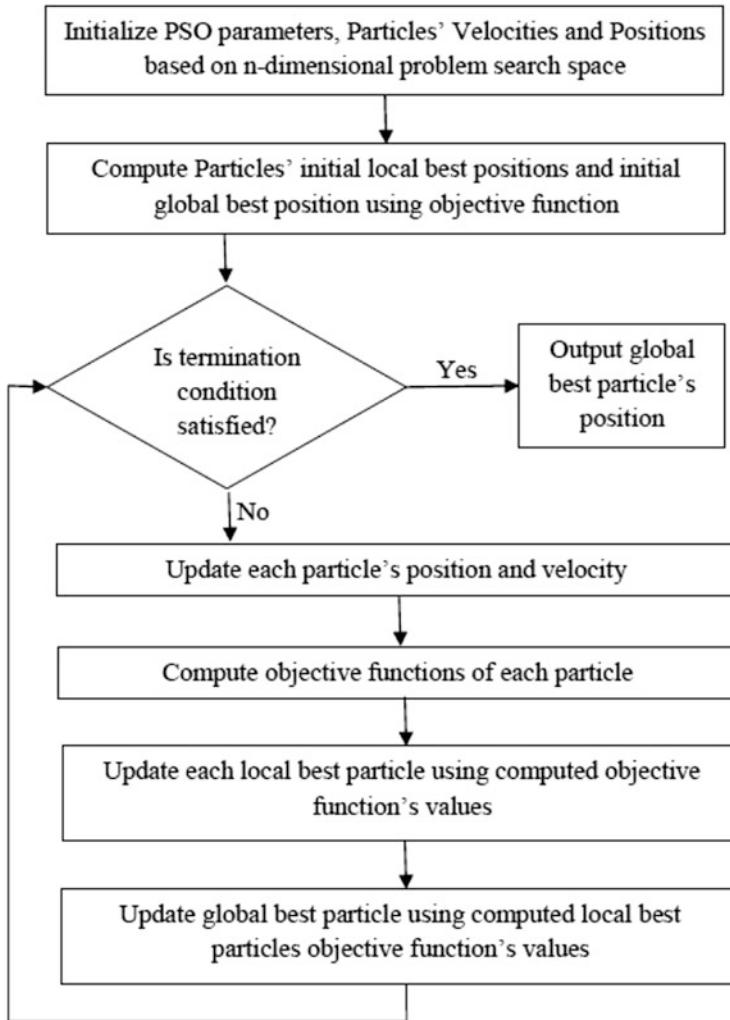


Fig. 6.1 Flowchart of particle swarm optimization's procedure

6.4 Parameter Selection

Shi and Eberhart (1998b), Rezaee Jordehi and Jasni (2013), and Wang, Tan, and Liu (2018) surveyed and presented various parameter selection methods found in the literature. Table 6.2 shows PSO parameters, the purpose of each parameter, and possible values or selection methods for each of these parameters. These parameter selections can help in achieving the best output from PSO. Furthermore, sensitivity analysis (Bartz-Beielstein, Parsopoulos, & Vrahatis, 2002), regression trees (Bartz-Beielstein, Parsopoulos, Velt, & Vrahatis, 2004), and statistics (Bartz-Beielstein,

Parsopoulos, & Vrahatis, 2004) can help in selecting the optimal parameters of the PSO algorithm so that PSO algorithm can solve practical problems better.

Algorithm 6.1: Particle Swarm Optimization

Input: $X[][], V[], f()$; Position Matrix, Velocity Matrix, Objective function
Output: P_g ; Global best particle

```

1 for each i in {1, 2, ..., Np} do
2   for each d in {1, 2, ..., n} do
3     //UB= Upper Boundary, LB=Lower Boundary of search space
4     Initialize  $X_{i,d}$  with a uniformly distributed random vector with  $(LB, UB)$ 
5     Initialize  $V_{i,d}$  with a uniformly distributed random vector with
6        $(-|UB - LB|, |UB - LB|)$ 
7    $P_i = X_i$ 
8    $P_g = P_1$ 
9   for each i in {2, 3, ..., Np} do
10    if  $f(P_i)$  is better than  $f(P_g)$  then
11       $P_g = P_i$ 
12       $gbest = f(P_g)$ 
13 for each t in {1, 2, ..., MaxIterations} do
14   for each i in {1, 2, ..., Np} do
15     for each d in {1, 2, ..., n} do
16       /* $C_1$  and  $C_2$  are two positive numbers, and  $r_{1d}$  and  $r_{2d}$  are two random
17       numbers with uniform distribution in the interval [0,1].*/
18        $V_{i,d}(t+1) = V_{i,d}(t) + C_1 r_{1d}(P_{i,d} - X_{i,d}) + C_2 r_{2d}(P_{gd} - X_{i,d})$  (6.1)
19        $X_{i,d}(t+1) = X_{i,d}(t) + V_{i,d}(t+1)$  (6.2)
20   //Update the particle's best known position
21   if  $f(X_i)$  is better than  $f(P_i)$  then
22      $P_i = X_i$ 
23      $lbest = f(P_i)$ 
24   //Update the swarm's best known position
25   if  $f(P_i)$  is better than  $f(P_g)$  then
26      $P_g = P_i$ 
27      $gbest = f(P_g)$ 
```

$$\text{If } |V_{i,d}| > V_{\max}, \text{ then } V_{i,d} = \text{sign}(V_{i,d})V_{\max} \quad (6.3)$$

$$V_{i,d}(t+1) = \omega V_{i,d}(t) + C_1 r_{1d}(P_{i,d} - X_{i,d}) + C_2 r_{2d}(P_{gd} - X_{i,d}) \quad (6.4)$$

$$V_{i,d}(t+1) = \chi(V_{i,d}(t) + C_1 r_{1d}(P_{i,d} - X_{i,d}) + C_2 r_{2d}(P_{gd} - X_{i,d})) \quad (6.5)$$

Table 6.1 Notations and their interpretation

Notation	Interpretation
X_i	Position vector of particle i
$X_{i,d}$	d th dimension of X_i
V_i	Velocity vector of particle i
V	d th dimension of V
$f(X_i)$	Objective function value of X_i
n	Dimension of problem in hand
t	Iteration number or time step
C_1	Cognitive acceleration coefficient
C_2	Social acceleration coefficient
P_i	The best position vector of particle i so far (local best position)
P_{best}	The best objective of particle i so far (local best fitness)
P_g	The best position vector of swarm particles so far (global best position)
g_{best}	The best objective of swarm particles (global best fitness)
V_{\max}	Maximum allowable velocity for particles
ω	Inertia weight
χ	Construction factor
N_p	Number of particles in swarm (swarm size)

6.5 PSO in Finance

PSO is applied to solve various optimization problems in finance. This section presents three such popular applications in finance:

6.5.1 Financial Market Prediction

The goal of financial market prediction problems such as FOREX rate prediction, stock market prediction, and commodity price prediction is to obtain accurate predictions to make the right decisions. One of the hybrid approaches using PSO is proposed by Pradeepkumar and Ravi (2014). In this approach, the artificial neural network (ANN) is used to obtain predictions. Later, the PSO-based regression model of errors is used to fine-tune the predictions obtained by ANN. The PSO minimizing mean squared error (MSE) is used to obtain optimal coefficients of the regression function of errors. The authors concluded that the proposed hybrid outperformed the standalone approaches.

Ravi et al. (2017) extended the approach aforementioned using multi-objective PSO (MOPSO) in place of PSO. The two objectives of MOPSO are the minimization of MSE and maximization of Dstat (directional change statistic). The authors concluded that MOPSO could yield optimal coefficients of regression in comparison with PSO.

Table 6.2 Parameter selection for PSO

Parameter	Purpose	Possible values/Selection methods
Swarm size (N_p)	Affects performance of PSO	20–50 (Sørensen & Glover, 2013; Wang et al., 2018)
Acceleration coefficients (C_1 & C_2)	Pull particles towards P_{best} and g_{best}	(1) $C_1 = C_2 = 2$ (Ozcan & Mohan, 1999)
		(2) Time-varying acceleration coefficients (Achayuthakan & Ongsakul, 2009; Bao & Mao, 2009)
		(3) Adaptive acceleration coefficients (Guo & Chen, 2009; Yun & Xue, 2009; Zhan, Xiao, Zhang, & Chen, 2007; Zhengjia & Jianzhong, 2009; Ziyu & Dingxue, 2009)
		(4) $C_1 = C_2 = 1.49445$ (Clerc & Kennedy, 2002)
		(5) $C_1 = 2.8$, $C_2 = 1.3$ (Carlisle & Dozier, 2001; Schutte & Groenwold, 2005)
		(6) Genetic algorithm (Yu, Zhang, Chen, Song, & Hu, 2005)
		(7) Adaptive fuzzy algorithm (Juang, Tung, & Chiu, 2011)
		(8) Differential evolutionary algorithm (Parsopoulos & Vrahatis, 2002)
Inertia weight (ω)	Adjusts the trade-off between exploration and exploitation of capabilities of PSO	(1) Fixed inertia weight (Shi & Eberhart, 1999)
		(2) Fuzzy adaptive (Bajpai & Singh, 2007; Liu, Ouyang, Zhu, & Tang, 2010)
		(3) Linearly decreasing (Shi & Eberhart, 1998b, 1999)
		(4) Multi-stage linearly decreasing (Xin, Chen, & Hai, 2009)
		(5) Linearly increasing (Zheng, Ma, Jhang, & Qian, 2003)
		(6) Non-linear (Li, Xue, Niu, Chai, & Wu, 2009)
		(7) Random (Lin & Hong, 2007; Zhang, Tang, Hua, & Guan, 2015)
		(8) Chaotic (Feng, Teng, Wang, & Yao, 2007)
		(9) Exponential (Jianxin, Xin, Weiguo, & Rui, 2009)
		(10) Gaussian (Pant, Radha, & Singh, 2007)
		(11) Parallel (Liu, Su, Gao, & Xu, 2009)

(continued)

Table 6.2 (continued)

Parameter	Purpose	Possible values/Selection methods
		(12) Simulated annealing inertia weight (Hassan, Fayeck, & Shaheen, 2006)
		(13) $\omega_{\max} = 0.9$ and $\omega_{\min} = 0.4$ (Han, Yang, Ren, & Sun, 2010)
		(14) $\omega = [0.9, 1.2]$ (Shi & Eberhart, 1999)
		(15) $\omega = [0.5 + (\text{rnd}/2.0)]$ (Eberhart et al., 2001)
Maximum velocity (V_{\max})	Constrains the speed of the particles	<p>(1) Set to a fixed value (Wang et al., 2018)</p> <p>(2) Linearly decreased value with time (Fan, 2002)</p> <p>(3) Dynamically reduced based on success of search history (Fourie & Groenwold, 2002)</p> <p>(4) $V_{\max} = \frac{X_{\max} - X_{\min}}{N_I}$</p>
		Where N_I is the number of intervals in the d th dimension selected by user. X_{\max} and X_{\min} are the maximum and minimum values that particles have achieved so far, respectively (Abido, 2001, 2002)
Maximum position (X_{\max})	Constrains positions of the particles	<p>(1) Absorbing wall, reflecting wall and invisible wall (Robinson & Rahmat-Samii, 2004)</p> <p>(2) Absorbing wall + reflecting wall (Huang & Mohan, 2005)</p> <p>(3) Hard position limit+absorbing wall +reflecting wall (Mikki & Kishk, 2005)</p>
Stopping Criteria	Terminates the particles convergence process	<p>(1) Prespecified number of iterations.</p> <p>(2) Achievement of a specified quality in solution.</p> <p>(3) Lapsing a specified time.</p> <p>(4) Lack of change in a certain successive iteration</p> <p>(5) a combination of above (Rezaee Jordehi & Jasni, 2013)</p>

6.5.2 Volatility Forecasting

The volatility forecasting problem's goal is to obtain accurate predictions to assist various financial stakeholders. Pradeepkumar and Ravi (2017) presented a PSO-trained quantile regression neural network (QRNN), namely, PSOQRNN, to forecast volatility of financial markets. In this approach, the weights of QRNN are obtained using PSO so that the PSOQRNN could yield accurate forecasts. The

authors concluded that PSO helped QRNN obtain accurate volatility forecasts compared to standalone QRNN and other similar volatility forecasting approaches.

6.5.3 Portfolio Optimization

The goal of portfolio optimization is to build the best investment portfolio according to a defined set of assets. Let us assume that we have selected N financial assets we want to invest in. They can be (daily, monthly, etc.) stocks, funds, bonds, ETF, etc. Each of these has many historical returns that are the relative price difference from one period to another.

Kunwar Madan (<https://github.com/KunwarMadan/Optimal-Financial-Portfolio-Selection>), in this context, presented an example of portfolio selection using PSO and genetic algorithm (GA) in Python. The author solved a 470-dimensional problem in which 470 stocks were considered in the portfolio. In 470-dimensional search space, PSO and GA are applied in finding the optimal combination of weights representing all stocks' capital using the Sharpe ratio. The author concluded that GA results after 2000 iterations were not even close to PSO results after 250 iterations. Hence, the author proved that PSO is better than GA to solve the portfolio optimization problem. The same fact is also proved by Chen and Zhu (2010). And also, the PSO is applied in constructing optimal risky portfolio (Cura, 2009; Dashti, Farjami, Vedadi, & Anissee, 2007; Kendall & Su, 2005; Mercangoz, 2019) and in solving constrained portfolio selection problem (Chen, Zhang, Cai, & Xu, 2006; Cui, Cheng, & Bai, 2014; Zhu, Wang, Wang, & Chen, 2011).

6.6 Variants of PSO for Portfolio Optimization

Table 6.3 presents various variants of PSO proposed for solving portfolio optimization problem. The authors concluded that the proposed PSO variants outperformed basic PSO aforementioned and other PSO variants.

Table 6.3 Variants of PSO for portfolio optimization

Year	Author(s)	PSO variant
2009	Niu, Xue, Li, and Chai (2009)	Symbiotic Multi-swarm PSO (SMPSO)
2009	Mario Villalobos-Arias (2009)	PSO with stripes (MOPSO-ST)
2012	Sharma, Thulasiram, and Thulasiraman (2012)	Normalized PSO (NPSO)
2014	Soleimanivareki, Fakharzadeh, and Poormoradi (2014)	Fuzzy Adaptive PSO
2015	Yin, Ni, and Zhai (2015)	Heterogeneous Multiple Population PSO (HMIPPSO)

6.7 Conclusion

Portfolio optimization aims at building the best investment portfolio according to a defined set of assets and constraints. PSO is good at obtaining the near-best global optimal solution from the search space of feasible solutions. Literature provided a base for the readers that PSO and its variants are the best fit for achieving the portfolio optimization problem's objective. This chapter provided descriptions of basic PSO, its parameter selection methods, and its variants. The readers can also be further directed to refer to Yarpiz (2020) for solving portfolio optimization problem using various classic and other SI algorithms such as imperialist competitive algorithm (ICA), non-dominated sorting genetic algorithm II (NSGA-II), and strength Pareto evolutionary algorithm 2 (SPEA2).

References

- Abido, A. (2001). Particle swarm optimisation for multi-machine power system stabilizer design. In *Proceedings of power engineering society summer meeting*. Washington, DC: IEEE Computer Society.
- Abido, A. (2002). Optimal power flow using particle swarm optimisation. *International Journal of Electrical Power & Energy Systems*, 24, 563–571.
- Achayuthakan, C., & Ongsakul, W. (2009). TVAC-PSO based optimal reactive power dispatch for reactive power cost allocation under deregulated environment. In *Proceedings of the IEEE international meeting of power and energy society* (pp. 1–9). Washington, DC: IEEE Computer Society.
- AlRashidi, M. R., & El-Hawary, M. E. (2008). A survey of particle swarm optimization applications in electric power systems. *IEEE Transactions on Evolutionary Computation*, 13(4), 913–918.
- Bajpai, P., & Singh, S. N. (2007). Fuzzy adaptive particle swarm optimisation for bidding strategy in uniform price spot market. *IEEE Transactions on Power Systems*, 22, 2152–2160.
- Bao, G. Q., & Mao, K. F. (2009). Particle swarm optimisation algorithm with asymmetric time-varying acceleration coefficients. In *Proceedings of the IEEE international conference on robotics and biomimetics* (pp. 2134–2139). Washington DC: IEEE Computer Society.
- Bartz-Beielstein, T., Parsopoulos, K. E., Vegt, M. D., & Vrahatis, M. N. (2004). Designing particle swarm optimization with regression trees. In *Technical Report CI 173/04, SFB 531. University of Dortmund*. Dortmund, Germany: Department of Computer Science.
- Bartz-Beielstein, T., Parsopoulos, K. E., & Vrahatis, M. N. (2002). Tuning PSO parameters through sensitivity analysis. In *Technical Report CI 124/02, SFB 531. University of Dortmund*. Dortmund, Germany: Department of Computer Science.
- Bartz-Beielstein, T., Parsopoulos, K. E., & Vrahatis, M. N. (2004). Analysis of particle swarm optimization using computational statistics. In *Proceedings of the international conference of numerical analysis and applied mathematics (ICNAAM 2004)* (pp. 34–37). Chalkis, Greece: ICNAAM.
- Carlisle, A., & Dozier, G. (2001). An off-the-shelf PSO. In *Proceedings of the workshop on particle swarm optimization*. Indiana, USA: Indianapolis.
- Chen, W., Zhang, R., Cai, Y., & Xu, F. (2006). Particle swarm optimization for constrained portfolio selection problems. In *2006 International Conference on Machine Learning and Cybernetics* (pp. 2425–2429). China: Dalian. <https://doi.org/10.1109/ICMLC.2006.258773>.

- Chen, Y., & Zhu, H. (2010). PSO heuristics algorithm for portfolio optimization. In Y. Tan, Y. Shi, & K. C. Tan (Eds.), *Advances in Swarm Intelligence. ICSI 2010* (Lecture Notes in Computer Science) (Vol. 6145). Berlin, Heidelberg: Springer. https://doi.org/10.1007/978-3-642-13495-1_23.
- Clerc, M., & Kennedy, J. (2002). The particle swarm-explosion, stability and convergence in a multi dimensional complex space. *IEEE Transactions on Evolutionary Computation*, 6(2), 58–73.
- Cui, T., Cheng, S., & Bai, R. (2014, July). A combinatorial algorithm for the cardinality constrained portfolio optimization problem. In *2014 IEEE Congress on Evolutionary Computation (CEC)* (pp. 491–498). New York: IEEE.
- Cura, T. (2009). Particle swarm optimization approach to portfolio optimization. *Nonlinear Analysis: Real World Applications*, 10(4), 2396–2406.
- Dashti, M. A., Farjami, Y., Vedadi, A., & Anissee, M. (2007). Implementation of particle swarm optimization in construction of optimal risky portfolios. In *2007 IEEE international conference on industrial engineering and engineering management* (pp. 812–816). Singapore: IEEE. <https://doi.org/10.1109/IEEM.2007.4419303>.
- Eberhart, R., & Kennedy, J. (1995, November). Particle swarm optimization. In *Proceedings of the IEEE international conference on neural networks* (Vol. 4, pp. 1942–1948). New York: IEEE.
- Eberhart, R., Shi, Y., & Kennedy, J. (2001). *Swarm intelligence*. San Mateo, CA: Morgan Kaufmann.
- Eberhart, R. C., & Shi, Y. (2004). Guest editorial special issue on particle swarm optimization. *IEEE Transactions on Evolutionary Computation*, 8(3), 201–203.
- Fan, H. (2002). A modification to particle swarm optimization algorithm. *Engineering Computations*, 19(8), 970–989.
- Feng, Y., Teng, G. F., Wang, A. X., & Yao, Y. M. (2007). Chaotic inertia weight in particle swarm optimisation. In *Proceedings of the IEEE international conference on innovative computing, information and control* (pp. 475–479). Washington, DC: IEEE Computer Society.
- Fourie, P. C., & Groenwold, A. A. (2002). The particle swarm optimization algorithm in size and shape optimization. *Structural and Multidisciplinary Optimization*, 23(4), 259–267.
- Guo, L., & Chen, X. (2009). A novel particle swarm optimisation based on the self-adaptation strategy of acceleration coefficients. In *Proceedings of the IEEE international conference on computational intelligence and security* (pp. 277–281). Washington, DC: IEEE Computer Society.
- Han, W., Yang, P., Ren, H., & Sun, J. (2010). Comparison study of several kinds of inertia weight for PSO. In *Proceedings of the IEEE international conference on progress in informatics and computing* (pp. 280–284). Washington, DC: IEEE Computer Society.
- Hassan, W. A., Fayek, M. B., & Shaheen, S. I. (2006). PSOSA: An optimised particle swarm technique for solving the urban planning problem. In *Proceedings of the IEEE international conference on computer engineering and systems* (pp. 401–405). Washington, DC: IEEE Computer Society.
- Heppner, F., & Grenander, U. (1990). A stochastic nonlinear model for coordinated bird flocks. In S. Krasner (Ed.), *The ubiquity of chaos*. Washington, DC: AAAS Publications.
- Huang, T., & Mohan, A. S. (2005). A hybrid boundary condition for robust particle swarm optimization. *Antennas Wirel Propag Lett*, 4, 112–117.
- Jianxin, W., Xin, H. X., Weiguo, Z., & Rui, W. (2009). Exponential inertia weight particle swarm algorithm for dynamic optimisation of electromechanical coupling system. In *Proceedings of the IEEE international conference on intelligent computing and intelligent systems* (pp. 479–483). Washington DC: IEEE Computer Society.
- Juang, Y. T., Tung, S. L., & Chiu, H. C. (2011). Adaptive fuzzy particle swarm optimization for global optimization of multimodal functions. *Information Sciences*, 181, 4539–4549.
- Kendall, G., & Su, Y. (2005, January). Particle swarm optimisation approach in the construction of optimal risky portfolios. In *Proceedings of the 23rd IASTED International Multi-Conference Artificial Intelligence and Applications*. Innsbruck: IASTED.

- Li, L., Xue, B., Niu, B., Chai, Y., & Wu, J. (2009). The novel nonlinear strategy of inertia weight in particle swarm optimisation. In *Proceedings of the IEEE international conference on bio-inspired computation* (pp. 1–5). Washington, DC: IEEE Computer Society.
- Lin, G. Y., & Hong, D. Y. (2007). A new particle swarm optimisation algorithm with random inertia weight and evolution strategy. In *Proceedings of the IEEE international conference on computational intelligence and security* (pp. 199–203). Washington, DC: IEEE Computer Society.
- Liu, C., Ouyang, C., Zhu, P., & Tang, W. (2010). An adaptive fuzzy weight PSO algorithm. In *Proceedings of genetic and evolutionary computing* (pp. 8–10). Washington, DC: IEEE Computer Society.
- Liu, H., Su, R., Gao, Y., & Xu, R. (2009). Improved particle swarm optimisation using two novel parallel inertia weights. In *Proceedings of the IEEE international conference on intelligent computation technology and automation engineering and systems* (pp. 185–188). Washington, DC: IEEE Computer Society.
- Mario Villalobos-Arias, M. (2009). Portfolio optimization using particle swarms with stripes. *Revista de Matematica: Teoria y Aplicaciones*, 16(2), 205–220, cimpaucr issn: 1409- 2433.
- Mercangoz, B. A. (2019). Particle swarm algorithm: An application on portfolio optimization. In J. Ray, A. Mukherjee, S. K. Dey, & G. Klepac (Eds.), *Metaheuristic Approaches to Portfolio Optimization* (pp. 27–59). Hershey: IGI Global. <https://doi.org/10.4018/978-1-5225-8103-1.ch002>.
- Mikki, S., & Kishk, A. (2005). Improved particle swarm optimization technique using hard boundary conditions. *Microwave and Optical Technology Letters*, 46(5), 422–426.
- Millonas, M. M. (1993). Swarms, phase transitions, and collective intelligence. In C. G. Langton (Ed.), *Proceedings of ALIFE III*. Santa Fe Institute, USA: Addison-Wesley.
- Niu, B., Xue, B., Li, L., & Chai, Y. (2009, September). Symbiotic multi-swarm PSO for portfolio optimization. In *International Conference on Intelligent Computing* (pp. 776–784). Berlin, Heidelberg: Springer.
- Nowak, A., Szamrej, J., & Latané, B. (1990). From private attitude to public opinion: A dynamic theory of social impact. *Psychological Review*, 97, 362. <https://doi.org/10.1037/0033-295X.97.3.362>.
- Ozcan, E., & Mohan, C. (1999). Particle swarm optimisation: Surfing the waves. In *Proceedings of the IEEE international congress on evolutionary computation* (pp. 1939–1944). Washington, DC: IEEE Computer Society.
- Pant, M., Radha, T., & Singh, V. P. (2007). Particle swarm optimisation using Gaussian inertia weight. In *Proceedings of the IEEE international conference on computational intelligence and multimedia* (pp. 97–102). Washington, DC: IEEE Computer Society.
- Parsopoulos, K. E., & Vrahatis, M. N. (2002). Recent approaches to global optimization problems through particle swarm optimization. *Natural Computing*, 1, 235–306.
- Parsopoulos, K. E., & Vrahatis, M. N. (2010). Particle swarm optimization and intelligence: Advances and applications. In *Information Science Reference (an imprint of IGI Global)*. Hershey: IGI Global.
- Poli, R. (2008). Analysis of the publications on the applications of particle swarm optimisation. *Journal of Artificial Evolution and Applications*, 2008, 685175.
- Poli, R., Kennedy, J., & Blackwell, T. (2007). Particle swarm optimization—an overview. *Swarm Intelligence*, 1(1), 33–57.
- Pradeepkumar, D., & Ravi, V. (2014, October). FOREX rate prediction using chaos, neural network and particle swarm optimization. In *International Conference in Swarm Intelligence* (pp. 363–375). Cham: Springer.
- Pradeepkumar, D., & Ravi, V. (2017). Forecasting financial time series volatility using particle swarm optimization trained quantile regression neural network. *Applied Soft Computing*, 58, 35–52.
- Pradeepkumar, D., & Ravi, V. (2018). Soft computing hybrids for FOREX rate prediction: A comprehensive review. *Computers & Operations Research*, 99, 262–284.

- Ravi, V., Pradeepkumar, D., & Deb, K. (2017). Financial time series prediction using hybrids of chaos theory, multi-layer perceptron and multi-objective evolutionary algorithms. *Swarm and Evolutionary Computation*, 36, 136–149.
- Reeves, W. T. (1983). Particle systems—A technique for modeling a class of fuzzy objects. *ACM Transactions on Graphics*, 2(2), 91–108.
- Reynolds, C. W. (1987). Flocks, herds, and schools: A distributed behavioral model. *Computer Graphics and Interactive Techniques*, 21(4), 25–34.
- Rezaee Jordehi, A., & Jasni, J. (2013). Parameter selection in particle swarm optimisation: A survey. *Journal of Experimental & Theoretical Artificial Intelligence*, 25(4), 527–542.
- Robinson, J., & Rahmat-Samii, Y. (2004). Particle swarm optimization in electromagnetics. *IEEE Transactions on Antennas and Propagation*, 52(2), 397–407.
- Schutte, J. F., & Groenwold, A. A. (2005). A study of global optimization using particle swarms. *Journal of Global Optimization*, 31, 93–108.
- Sharma, B., Thulasiram, R., & Thulasiraman, P. (2012). Portfolio Management Using Particle Swarm Optimization on GPU. In *2012 IEEE 10th International Symposium on Parallel and Distributed Processing with Applications (ISPA)* (pp. 103–110). Leganes: IEEE. <https://doi.org/10.1109/ISPA.2012.22>.
- Shi, Y., & Eberhart, R. (1998a). A modified Particle swarm optimiser. In *Proceedings of the IEEE international conference on computational intelligence* (pp. 69–73). Washington, DC: IEEE Computer Society.
- Shi, Y., & Eberhart, R. (1998b). Parameter selection in particle swarm optimisation. In *Proceedings of the IEEE international conference on evolutionary programming* (pp. 591–600). Washington, DC: IEEE Computer Society.
- Shi, Y., & Eberhart, R. (1999). Empirical study of particle swarm optimisation. In *Proceedings of the IEEE international conference on computational intelligence* (pp. 1945–1950). Washington, DC: IEEE Computer Society.
- Soleimani-vareki, M. A., Fakharzadeh, J., & Poormoradi, M. (2014). Fuzzy adaptive Pso approach for portfolio optimization problem. *The Journal of Mathematics and Computer Science*, 12(3), 235–242.
- Sörensen, K., & Glover, F. W. (2013). Metaheuristics. In *Encyclopedia of operations research and management science* (pp. 960–970). New York: Springer US.
- Wang, D., Tan, D., & Liu, L. (2018). Particle swarm optimization algorithm: an overview. *Soft Computing*, 22(2), 387–408.
- Xin, J., Chen, G., & Hai, Y. (2009). A particle swarm optimiser with multi-stage linearly decreasing inertia weight. In *Proceedings of the IEEE international conference on computational sciences and optimisation* (pp. 505–508). Washington, DC: IEEE Computer Society.
- Yarpiz. (2020). Portfolio optimization using classic and intelligent algorithms. MATLAB Central File Exchange. Retrieved Nov 17, 2020, <https://www.mathworks.com/matlabcentral/fileexchange/53143-portfolio-optimization-using-classic-and-intelligent-algorithms>.
- Yin, X., Ni, Q., & Zhai, Y. (2015). A novel PSO for portfolio optimization based on heterogeneous multiple population strategy. In *2015 IEEE Congress on Evolutionary Computation (CEC)* (pp. 1196–1203). Sendai: CEC. <https://doi.org/10.1109/CEC.2015.7257025>.
- Yu, H., Zhang, L., Chen, D., Song, X., & Hu, S. (2005). Estimation of model parameters using composite particle swarm optimization. *Journal of Chemical Engineering of Chinese Universities*, 19(5), 675–680.
- Yun, W. G., & Xue, H. D. (2009). Particle swarm optimisation based on self-adaptive acceleration factors. In *Proceedings of the IEEE international conference on genetic and evolutionary computing* (pp. 637–640). Washington, DC: IEEE Computer Society.
- Zhan, Z. H., Xiao, J., Zhang, J., & Chen, W. N. (2007). Adaptive control of acceleration coefficients for particle swarm optimisation based on clustering analysis. In *Proceedings of the IEEE international congress on evolutionary computation* (pp. 3276–3282). Washington, DC: IEEE Computer Society.

- Zhang, L., Tang, Y., Hua, C., & Guan, X. (2015). A new particle swarm optimization algorithm with adaptive inertia weight based on Bayesian techniques. *Applied Soft Computing*, 28, 138–149.
- Zheng, Y. L., Ma, L. H., Jhang, L. Y., & Qian, J. X. (2003). Empirical study of particle swarm optimiser with an increasing inertia weight. In *Proceedings of the IEEE international congress on evolutionary computation* (pp. 221–226). Washington, DC: IEEE Computer Society.
- Zhengjia, W., & Jianzhong, Z. (2009). A self-adaptive particle swarm optimisation algorithm with individual coefficients adjustment. In *Proceedings of the IEEE international symposium on intelligent information technology application* (pp. 396–399). Washington, DC: IEEE Computer Society.
- Zhu, H., Wang, Y., Wang, K., & Chen, Y. (2011). Particle swarm optimization (PSO) for the constrained portfolio optimization problem. *Expert Systems with Applications*, 38(8), 10161–10169.
- Ziyu, T., & Dingxue, Z. (2009). A modified particle swarm optimisation with adaptive acceleration coefficients. In *Proceedings of the IEEE international conference on information processing* (pp. 330–332). Washington, DC: IEEE Computer Society.

Chapter 7

The PSO Family: Application to the Portfolio Optimization Problem



Lucas Fernández-Brillet, Oscar Álvarez, and
Juan Luis Fernández-Martínez

Abstract Nonlinear high-dimensional optimization problems are generally ill-posed and ill-conditioned, with different sets of models located in one or different disconnected valleys of the cost function landscape with similar values. This situation generates uncertainty in the identification of the optimum model parameters that should be translated to the decision that has to be made. Therefore, the analysis of the uncertainty space of this type of problems is required in order to adopt robust decisions. This is done by sampling the cost function topography within the intricate solution set valleys that belong to the nonlinear equivalence region and validating the existence of different scenarios. This is also the case of the portfolio optimization problem, which admits multiple solutions depending on the expected return-risk ratio. As the popular wisdom says, you cannot have the butter and the money of having sold it. This mathematical situation is known as a Pareto front, which aims to show the boundary of the nonlinear equivalence region of the corresponding decision problem. In this chapter, we introduce the concept of uncertainty in high-dimensional problems, proposing the particle swarm optimization family as a parameter free-tuning global algorithm, capable of sampling the nonlinear equivalent region in parallel with the optimization. For that purpose, these algorithms should be exploratory. This feature is related to the automatic tuning of the PSO parameters in the neighborhood of the stochastic second-order stability region of the particle trajectories. These algorithms are faster than Monte Carlo methods. The chapter concludes with the application of PSO to the portfolio optimization problem of the IBEX-35, which is the main stock market index of the *Bolsa de Madrid*. In this case, the cost function is constructed in a way that the investors seek to maximize the

L. Fernández-Brillet · O. Álvarez
Miscellany Creative-StockFink, Oviedo, Spain
<https://www.stockfink.com>

J. L. Fernández-Martínez (✉)
Group of Inverse Problems, Optimization and Machine Learning, Department of Mathematics,
University of Oviedo, Oviedo, Spain
e-mail: jlfm@uniovi.es

portfolio's expected return subjected to a given risk. The optimization of the portfolio composition follows a previous selection of the stocks.

Keywords PSO family · Uncertainty · Portfolio optimization

7.1 Introduction

The problem of portfolio optimization is an important discipline of risk management in finance that consists in finding the optimum allocation among several assets. Nevertheless, this problem can be generalized in its abstract form to different fields. In general, an investor has a series of possibilities for the composition of his portfolio, and the question is to choose that composition that maximizes the return of investment, minimizing the risk. For instance, a similar problem occurs in the field of betting, in which a player has a series of options (bets) with different probabilities of success and different quotas (odds), and the problem consists in optimizing the stake to maximize the gain, minimizing the risk of loss. Generally speaking, we can talk about different options that provide different pairs of expected return and risk.

As any optimization problem, its result depends on how the cost function is built and the space of restrictions that is provided to the problem. The first works were made by Markowitz in 1952, who proposed the mean-variance model and used the technique of quadratic programming to solve it. It is out of scope of this paper to analyze all the different approaches that have been proposed to measure risk instead of the standard deviation (Sortino & Price, 1994; Tyrrell Rockafellar & Uryasev, 2000).

Also, the numerical algorithms used are very diverse and include the use of the simplex algorithm (Wolfe, 1959) and also the use of heuristic methods such as genetic algorithms (see, e.g., Chang, Yang, & Chang, 2009) or particle swarm optimization (Cura, 2009; Xu, Chen, & Yang, 2007). The main difference between local and global optimization methods is that, while the first try to converge to the optimum solution, the seconds are able to perform sampling while optimizing along the low misfit regions if they are tuned to be exploratory enough. This is one of the main advantages of the PSO family that is presented in this paper, the free parameter tuning, since the good PSO parameters that ensure a good balance between exploration and exploitation are located in the neighborhood of the upper limit of the second-order stability region of these algorithms (Fernández-Martínez & García-Gonzalo, 2009, 2011, 2012; García-Gonzalo & Fernández-Martínez, 2014). Our particular experience in many inverse problems shows that PSO can achieve this task much faster than random sampling methods and better than other global optimization algorithms.

Particularly, Markowitz (1956) introduced the critical line algorithm (CLA) to simplify the solution of the minimization of the portfolio variance with equality and inequality constraints. This method, which is used in practice, can be considered as a local optimization method designed to track the efficient frontier. Compared to the

methodology exhibited in this paper, PSO samples all the possible combinations of the weights of the preselected assets within some bounds, and the efficient frontier arises as the result of this sampling approach. Besides, PSO can be viewed as an intelligent search method (doubled stochastic gradient in the portfolio space) that does not need any optimization procedure, only the solution of the forward problem for each particle of the swarm, that is, given the weights of the portfolio, computing the value of the optimization function. The trick is that with iterations, the swarm will explore different interesting basins of the cost function landscape where the efficient portfolios are located. These portfolios belong to the nonlinear equivalence region of the portfolio optimization problem, and based on that, it is possible to calculate the posterior distribution of the portfolio composition. Besides, the implementation of PSO is much simpler and remains also very fast.

The structure of this chapter is as follows: in Sect. 7.2, we explain the topography of the cost function in nonlinear high-dimensional problems introducing the nonlinear region of uncertainty, where the efficient portfolios will be located in the case of the portfolio optimization problem. The existence of this region is somehow independent on how the cost function is constructed. In Sect. 7.3, we present PSO, the PSO continuous model, and some interesting members of the PSO family introduced by ourselves in the recent past. In Sect. 7.4, we present the application of PSO to the portfolio optimization problem. The goal is not to compare the different PSO family members, since all of them behave equally well, but to show how combining these methods with a preselection of the stocks, we can get very fast and interesting results. In this case, PSO can optimize the weights of the selected assets or can also perform the optimization of the selection in the preselected set. In both cases, the portfolio return-risk regions exhibit different shapes. Anyone should bear in mind the free lunch theorem for search and optimization (Wolpert & Macready, 1997) which proves no superiority of any particular algorithm among the rest when they are used in many different optimization problems. Besides and finally, models are to be used, not to be believed, that is, mathematical models are abstractions that serve to analyze and take decisions. The last section is devoted to the conclusions.

7.2 Nonlinear High-Dimensional Problems and Their Cost Function Topography

Optimization problems are generally ill-posed, i.e., multiple different sets of model parameters exist that satisfy a given cost value tolerance. In this sense, let us briefly analyze the nature of high-dimensional optimization problems in relation to their cost function topography (or landscape).

Let us consider a model $\mathbf{m} \in \mathbf{M} \subset \mathbb{R}^n$, where \mathbf{M} is the set of admissible models. The optimization problem consists in finding the model $\mathbf{m}_p = \min_{m \in M} C(\mathbf{m})$ having the minimum value of the scalar cost function $C(\mathbf{m}) : \mathbb{R}^n \rightarrow \mathbb{R}$. This is a challenging

problem in higher dimensions due to the intricate landscape of the cost function, which does not have a convex character, with different models located on flat curvilinear valleys of low values of the cost function topography.

To show this fact, let us suppose that \mathbf{m}_p is the global optimum of the cost function; thus, $\nabla C(\mathbf{m}_p) = \mathbf{0}$. Considering a Taylor expansion of the energy function E around the model \mathbf{m}_p , we have:

$$\begin{aligned} C(\mathbf{m}) &\approx C(\mathbf{m}_p) + \nabla C(\mathbf{m}_p) \cdot (\mathbf{m} - \mathbf{m}_p) + \frac{1}{2} (\mathbf{m} - \mathbf{m}_p)^T HC(\mathbf{m}_p) \\ &\quad + o\left(\|\mathbf{m} - \mathbf{m}_p\|_2^2\right), \end{aligned} \quad (7.1)$$

where $HC(\mathbf{m}_p)$ is the Hessian or curvature matrix of the cost function C calculated in model \mathbf{m}_p , and $o\left(\|\mathbf{m} - \mathbf{m}_p\|_2^2\right)$ is a scalar function that vanishes faster than the squared distance between the models \mathbf{m} and \mathbf{m}_p .

Neglecting the $o\left(\|\mathbf{m} - \mathbf{m}_p\|_2^2\right)$ term, that is, approaching $C(\mathbf{m})$ by its second-order Taylor expansion:

$$\begin{aligned} C(\mathbf{m}) &\approx C_S(\mathbf{m}) \\ &= C(\mathbf{m}_p) + \nabla C(\mathbf{m}_p) \cdot (\mathbf{m} - \mathbf{m}_p) + \frac{1}{2} (\mathbf{m} - \mathbf{m}_p)^T HC(\mathbf{m}_p), \end{aligned} \quad (7.2)$$

and taking into account that $\nabla C(\mathbf{m}_p) = \mathbf{0}$ and the positive definite character of $HC(\mathbf{m}_p)$ if \mathbf{m}_p is the global minimum, then, we have:

$$C_S(\mathbf{m}) \leq \text{tol} \rightarrow \frac{1}{2} (\mathbf{m} - \mathbf{m}_p)^T HC(\mathbf{m}_p) (\mathbf{m} - \mathbf{m}_p) \leq \text{tol} - C(\mathbf{m}_p). \quad (7.3)$$

Expression (7.3) implies that the model parameters with a cost function less than tol belong locally to a hyperquadric centered in \mathbf{m}_p that have the Hessian, $HC(\mathbf{m}_p)$, as matrix. This is obviously a local approximation.

The Hessian is a symmetric matrix and allows orthogonal factorization $HC(\mathbf{m}_p) = \mathbf{V}\mathbf{D}\mathbf{V}^T$, where \mathbf{V} is an orthogonal matrix and the eigenvalues of \mathbf{D} (diagonal) are positive real numbers due to the definite positive character of $HC(\mathbf{m}_p)$.

By calling $\Delta\mathbf{m} = \mathbf{m} - \mathbf{m}_p$, the hyperquadric that approximates locally the nonlinear equivalent region of value tol writes:

$$\Delta\mathbf{m}^T HC(\mathbf{m}_p) \Delta\mathbf{m} \leq 2(\text{tol} - C(\mathbf{m}_p)), \quad (7.4)$$

$$\Delta\mathbf{m}_V^T HC(\mathbf{m}_p) \Delta\mathbf{m}_V \leq 2(\text{tol} - C(\mathbf{m}_p)), \quad (7.5)$$

when the model increments are referred to the \mathbf{V} base, $\Delta\mathbf{m}_V = \mathbf{V}^T \Delta\mathbf{m}$.

In the case where $HC(\mathbf{m}_p)$ is full rank, the bounding hyperquadric in (Eq. 7.5) can be written as:

$$\sum_{k=1}^n \left(\frac{\Delta m_{vk}}{\sqrt{\lambda_k}} \right)^2 = 2(tol - C(\mathbf{m}_p)), \quad (7.6)$$

which is an hyper-ellipsoid centered in \mathbf{m}_p whose principal directions coincide with the eigenvectors \mathbf{v}_k (columns of \mathbf{V}), and the length axes are $1/\sqrt{\lambda_k}$, with λ_k being the eigenvalues of \mathbf{D} . In the case where the Hessian $HC(\mathbf{m}_p)$ is semi-definite positive, that is, it has a nontrivial null space associated to the null eigenvalues, then, the hyperquadric (Eq. 7.6) becomes an elliptical cylinder. As it has been explained above, the bounding hyperquadric (Eq. 7.6) only delimitates locally in the neighborhood of \mathbf{m}_p —the nonlinear equivalent region—that is, the models \mathbf{m} which fulfill the condition (Eq. 7.3).

Now, considering the same type of analysis for a different model \mathbf{m}_n which is located in the neighborhood of the model \mathbf{m}_p and belongs to the nonlinear stability region, we have:

$$\nabla C(\mathbf{m}_n) \cdot (\mathbf{m} - \mathbf{m}_n) + \frac{1}{2}(\mathbf{m} - \mathbf{m}_n)^T HC(\mathbf{m}_n)(\mathbf{m} - \mathbf{m}_n) \leq tol - C(\mathbf{m}_p). \quad (7.7)$$

Several remarks are important (Fernández-Martínez, Fernández-Muñiz, & Tompkins, 2012) to understand the complexity of the uncertainty analysis in non-convex optimization problems:

1. It is possible to prove that the center of the hyperquadric (Eq. 7.7) coincides with one iteration of the Gauss-Newton algorithm applied to $\min_{\mathbf{m} \in M} C(\mathbf{m})$ taking \mathbf{m}_n as initial guess. Therefore, local optimization methods in non-convex optimization problems wander around different models of the nonlinear equivalent region searching for the global optimum. Unfortunately, these methods might not converge, and they do not keep track of the good models that have been visited during the optimization process.
2. The Hessian matrix $HC(\mathbf{m}_n)$ might lose its semi-definite positive character. In this case, the hyperquadric becomes a hyperboloid, and the function landscape shows saddle points that indicate the presence of different basins of low-cost equivalent models. Therefore, the cost function topography in the nonlinear case becomes very intricate.
3. The main orientations of the local hyperquadric change with the model that is considered, since $HC(\mathbf{m}_n)$ is the hyperquadric matrix. Thus, the nonlinear region of the equivalent models exhibits a banana-shape structure. Besides, the cost function landscape could be multimodal, composed by different low-cost value elongated basins with almost null gradients. This fact also applies to the portfolio optimization problem and means that the efficient frontier might be composed by different disconnected regions.

Fernández-Martínez, Pallero, Fernández-Muñiz, and Pedruelo-González (2014a, 2014b) have studied the effect of noise and that of regularization in linear and nonlinear problems proving that noise perturbs the location of the global optimum

that is found. Besides, the regularization techniques do not impede the existence of other equivalent models. This fact is also important. The noise always enters through the observed data in the cost function construction. In the case of portfolio optimization, we can conclude that its nonlinear equivalent region has to be related to the region delimited by the [efficient frontier](#) curve, as described by Markowitz ([1952](#)). To the best of our knowledge, this analysis has not been performed before in the case of the portfolio optimization problem. Nevertheless, it has some similarities with predicting the protein tertiary structure ([Álvarez, Fernández-Martínez, Corbeanu, et al., 2019](#)), where we have used a similar way of reasoning.

7.3 PSO: The Continuous PSO and the PSO Family

Particle swarm optimization (PSO) is an evolutionary computation technique for optimization, which is inspired by the social behavior of individuals in groups in nature ([Kennedy & Eberhart, 1995](#)). In essence, any PSO algorithm consists of the following:

1. Particles are vectors whose length is equal to the number of degrees of freedom of the optimization problem.
2. A population of particles is generated as initial conditions with random positions (\mathbf{x}_i^0) and velocities (\mathbf{v}_i^0).
3. At each iteration, a misfit or cost function is evaluated for each particle. The velocities and positions for each particle are updated as per each particle's own history of misfit and the misfit of its neighboring particles. At each time step, k , positions (\mathbf{x}_i^k) and velocities (\mathbf{v}_i^k) are updated according to the mechanical equations:

$$\begin{aligned}\mathbf{v}_i^k &= w\mathbf{v}_i^{k-1} + \phi_1 \cdot 1 \cdot (\mathbf{g}^{k-1} - \mathbf{x}_i^{k-1}) + \phi_2 \cdot 1 \cdot (\mathbf{l}_i^{k-1} - \mathbf{x}_i^{k-1}), \\ \mathbf{x}_i^k &= \mathbf{x}_i^{k-1} + \mathbf{v}_i^k \cdot 1,\end{aligned}\tag{7.8}$$

where $\phi_1 = r_1 a_g$, $\phi_2 = r_2 a_l$, $r_1, r_2 \rightarrow U(0, 1)$, and w , a_g , $a_l \in \mathbb{R}$. \mathbf{l}_i^{k-1} is the i -th particle's best position, \mathbf{g}^{k-1} is the global best position found until that iteration, ϕ_1 , ϕ_2 are the global and local accelerations, and w is a constant, called inertia. The scalar 1 in Eq. (7.8) stands for the time step necessary to make this algorithm dimensionally correct, as it corresponds to the relationship between space, velocity, and acceleration.

The flowchart for the PSO algorithm is very simple:

1. A prismatic search space for the model parameters is given, and an initial swarm of N_s particles is uniformly distributed in the search space, and their initial velocities are (usually) set to zero. This is called the initial swarm, and it is obviously very far from being the solution to the optimization problem. The

search space is the only prior information that is used during the optimization. It is important to remark that no inversion of any matrix is performed; therefore, no regularization term is needed. Besides, if the search space does not intersect the low misfit (or cost function value) region, then the algorithm will not be able to locate low misfit models.

2. The misfit (cost function values) of the initial population is evaluated solving N_s forward problems (or cost function evaluations), and the global best and the previous best of each particle are determined. Therefore, to use this kind of algorithms, the cost function evaluation has to be fast to solve.
3. Choosing of the PSO parameters w , a_g , a_l for each particle of the swarm, drawing of the random numbers r_1 , r_2 , and updating of the velocities and positions of each particle of the swarm using formula (7.8). This formula only accounts for PSO. Other PSO family members have different updating expressions for the velocity and the positions.
4. Iterate to point 2 until the maximum number iterations is reached, or some criteria are fulfilled. Typically, in this kind of sampling procedures, the algorithm finishes by iterations, when a correct sampling is performed, and/or the swarm has collapsed.

There are many papers and implementations devoted to the study of PSO. A search in Google of the acronym PSO gives more than 28.4 million results. Although PSO has been originally heuristically proposed based on a bioinspiration and also many PSO heuristic variants have been published in the literature, the stochastic convergence of this algorithm is related to the stability of the swarm; particularly, the first-order stability (stability of the mean trajectories) depends on the total mean acceleration and on the inertia constant. All these fancy modifications of the original algorithm are in fact not needed and are due to a poor understanding of the PSO dynamics (Fernández-Martínez & García-Gonzalo, 2013; García-Gonzalo & Fernández-Martínez, 2012).

In fact, the random numbers, r_1 and r_2 , affect the local and global acceleration terms, a_l and a_g , causing the particle trajectory to oscillate at each iteration around its center, namely:

$$\mathbf{o}_i^k = \frac{\phi_1 \mathbf{g}^{k-1} + \phi_2 \mathbf{l}_i^{k-1}}{\phi_1 + \phi_2}. \quad (7.9)$$

Due to the random effect introduced by the random variables r_1 and r_2 , the particle trajectories have to be considered as stochastic processes whose first-(mean) and second-order moments (variance and temporal covariance) are important to understand the algorithm convergence.

Therefore, the relative values of ϕ_1 and ϕ_2 will affect the global search and the local optimum found by the algorithm during iterations. The first-order stochastic stability region depends on $(w, \bar{\phi})$ (see, e.g., Fernández-Martínez, García-Gonzalo, & Fernández-Alvarez, 2008):

$$S_1 = \left\{ (w, \bar{\phi}) : |w| < 1, \bar{\phi} < \frac{1 - 2|\omega| + \omega^2}{1 + \omega} \right\}. \quad (7.10)$$

This region has a very important role, because if the PSO values $(w, \bar{\phi})$ of the different particles are chosen outside this region, then the particles are not attracted toward their oscillation center and the PSO algorithm might not converge. Conversely, if the PSO parameters (w, a_g, a_l) fall inside the first-order stability region, the trajectories of the particles will oscillate around their own center of attraction (Eq. 7.9), and the algorithm will converge if the global best, \mathbf{g}^{k-1} , and the local best, \mathbf{l}_i^{k-1} , approach the optimum of the cost function, supposing that it exists and it is unique. As a consequence, it is possible to infer that the algorithm converges to the global minimum if \mathbf{o}_i^k is attracting the particle i toward the cost function's global minimum. Nevertheless, as will be shown later on in this chapter, this condition is not sufficient, because a given amount of exploration is needed to avoid entrapment in the valley of equivalent solutions, or in local optima. The second-order stochastic stability analysis serves to clarify this condition (Fernández-Martínez & García-Gonzalo, 2011). A good balance between exploitation and exploration is needed in order to avoid being trapped in local minima valleys (such as in the case of multimodal functions). In that sense, it is very important to achieve a good exploration of the search space, and for that purpose, the best PSO parameters (w, a_g, a_l) are located close to the second-order stability region, where the attraction of the particle toward their oscillation center is weak and the exploratory behavior of the swarm occurs (Fernández-Martínez et al., 2008).

The following difference equation is defined for each particle within the swarm:

$$\begin{cases} \mathbf{x}_i^k + (\phi - w - 1)\mathbf{x}_i^{k-1} + w\mathbf{x}_i^{k-1} = \phi_1 \mathbf{g}^{k-1} + \phi_2 \mathbf{l}_i^{k-1}, \\ \mathbf{x}_i^0 = \mathbf{x}_{i0}, \\ \mathbf{x}_i^1 = \mathbf{x}_i^0 + w\mathbf{v}_{io} + \phi(\mathbf{o}_i^o - \mathbf{x}_i^0), \end{cases} \quad (7.11)$$

where $\phi = \phi_1 + \phi_2$ is a random variable with a mean $\bar{\phi} = \frac{a_g + a_l}{2}$, whose statistical distribution is triangular or trapezoidal. \mathbf{v}_{io} and \mathbf{x}_{i0} are the initial velocity and position of particle i , and \mathbf{x}_i^1 is the i th particle position in the first iteration.

The PSO algorithm can be physically interpreted as a stochastic damped mass-spring system (Fernández-Martínez et al., 2008). The PSO continuous model describes the continuous movement of each coordinate of any particle in the swarm and physically corresponds to a damped mass-spring system with two attractors, \mathbf{l}_i^{k-1} and \mathbf{g}^{k-1} , and whose rigidity constants are ϕ_2 and ϕ_1 :

$$\left\{ \begin{array}{l} \mathbf{x}_i''(t) + (1-w)\mathbf{x}_i'(t) + \phi \mathbf{x}_i(t) = \phi_1 \mathbf{g}(t) + \phi_2 \mathbf{l}_i(t), \\ \mathbf{x}_i(0) = \mathbf{x}_{i0}, \\ \mathbf{x}_i'(0) = \mathbf{v}_{i0}. \end{array} \right. \quad (7.12)$$

This model has scalar character since it holds for any of the coordinates of any particle in the swarm. PSO corresponds to a particular finite difference discretization of the differential equation. Indeed, considering a discretization of the PSO continuous model, centered in acceleration and backward in velocity:

$$x_i''(t) \simeq \frac{\mathbf{x}_i(\mathbf{t} + \Delta\mathbf{t}) - 2\mathbf{x}_i(\mathbf{t}) + \mathbf{x}_i(\mathbf{t} - \Delta\mathbf{t})}{\Delta\mathbf{t}^2}, \quad (7.13)$$

$$x_i'(t) \simeq \frac{\mathbf{x}_i(\mathbf{t}) - \mathbf{x}_i(\mathbf{t} - \Delta\mathbf{t})}{\Delta\mathbf{t}}, \quad (7.14)$$

it is possible to deduce the generalized PSO (GPSO) algorithm for any discretization time step, Δt :

$$\begin{aligned} \mathbf{v}(t + \Delta t) &= (1 - (1 - w)\Delta t)\mathbf{v}(t) + \phi_1 \Delta t (\mathbf{g}(t) - \mathbf{x}(t)) + \phi_2 \Delta t (\mathbf{l}(t) - \mathbf{x}(t)), \\ \mathbf{x}(t + \Delta t) &= \mathbf{x}(t) + \mathbf{v}(t + \Delta t)\Delta t. \end{aligned} \quad (7.15)$$

PSO equations are obtained from Eq. (7.15) by adopting a time unit discretization ($\Delta t = 1$). A systematic study of this discrete PSO algorithm was performed by (Fernández-Martínez & García-Gonzalo, 2008); Fernández-Martínez et al. (2008). In this case, the first-order stability region depends also on Δt . If $\Delta t > 1$, the stability region decreases in size, and the algorithm becomes more exploratory. Conversely if $\Delta t < 1$, this region increases in size, and the algorithm becomes more exploitative.

Fernández-Martínez and García-Gonzalo (2011) proved that GPSO performs fairly well in a very broad area in the parameter space (inertia weight and mean acceleration), and these regions are fairly similar for different types of benchmark functions. The best parameter performance is found when the median line of the first-order stability triangle (where the covariance between trajectories goes fast to zero implying iteration non-correlation) intersects with the hyperbola that represents the upper border of the second-order stability region where the variance of particle trajectories becomes unbounded. In this case, the trajectories have a high degree of exploration, that is, high variance and temporal uncorrelation. This result also holds for other PSO family members (Fernández-Martínez & García-Gonzalo, 2009, 2012) and was also generalized for any statistical distribution of the PSO parameters (García-Gonzalo & Fernández-Martínez, 2014).

Just till this point, we have presented PSO and its physical analogy, the PSO continuous model. We have also shown that the generalized PSO (or GPSO) can be deduced from the PSO continuous using a centered discretization in acceleration and

Table 7.1 Some interesting PSO family members

Algorithm	Expression
GPSO	$v(t + \Delta t) = (1 - (1 - w)\Delta t)v(t) + \phi_1\Delta t(g(t) - x(t)) + \phi_2\Delta t(l(t) - x(t)),$ $x(t + \Delta t) = x(t) + v(t + \Delta t)\Delta t$
PP-GPSO	$v(t + \Delta t) = (1 - (1 - w)\Delta t)v(t) + \phi_1\Delta t(g(t) - x(t)) + \phi_2\Delta t(l(t) - x(t)),$ $x(t + \Delta t) = x(t) + v(t)\Delta t$
CP-GPSO	$v(t + \Delta t) = \frac{1 - \bar{\phi}\Delta t^2}{1 + (1 - w)\Delta t}v(t) + \frac{\phi_1\Delta t}{1 + (1 - w)\Delta t}(g(t) - x(t)) + \frac{\phi_2\Delta t}{1 + (1 - w)\Delta t}(l(t) - x(t)),$ $x(t + \Delta t) = x(t) + v(t)\Delta t$
RR-GPSO	$v(t + \Delta t) = \frac{v(t) + \phi_1\Delta t(g(t) - x(t)) + \phi_2\Delta t(l(t) - x(t))}{1 + (1 - w)\Delta t + \bar{\phi}\Delta t^2},$ $x(t + \Delta t) = x(t) + v(t + \Delta t)\Delta t$

regressive discretization in velocity and exhibits a good balance between exploration and exploitation (convergence) if the PSO parameters (w, a_g, a_i) are correctly tuned in the neighborhood of the upper limit of the second-order stability region. In this section, we present other interesting PSO family members having medium to high exploratory character. These PSO variants are obtained by adopting different discretization methodologies of the continuous PSO model:

- PP-PSO corresponds to a progressive finite difference discretization of both acceleration and velocity. PP-PSO has medium exploratory character.
- CP-PSO uses a centered discretization in acceleration and progressive in velocity. PP-PSO has a very high exploratory character.
- RR-PSO uses regressive discretization in both velocity and acceleration. RR-PSO has a very high exploratory character.

The cloud versions of these algorithms are based on the stochastic stability analysis of the particle trajectories. The advantage of the cloud versions is that no parameter tuning (inertia, local and global accelerations) is needed, since each particle in the swarm has its own PSO parameters that are randomly selected from a set of PSO parameters that are located in the neighborhood of the upper limit of their second-order stability regions. Additionally, in the cloud design, each particle might have a different time step Δt . As it has been already outlined, Δt is a numerical parameter that serves to achieve the exploration of the search space when this parameter increases, and it is greater than 1.0. Conversely, the algorithm freezes the solution found when Δt is decreased to values lower than 1.0.

Table 7.1 summarizes the analytical expressions of all the PSO family members that have been cited above.

Particularly, in the case of RR-PSO, the optimum parameter sets are located along the line mean $\bar{\phi} = 3(w - \frac{3}{2})$, mainly for inertia values $w > 2$ (Fernández-Martínez & García-Gonzalo, 2012). Therefore, the tuning of the RR-PSO algorithm to achieve a good exploratory behavior is very simple. This line is located in a region of medium attenuation and very high frequency for the particle trajectories. This feature confers to RR-PSO a good equilibrium between exploration and exploitation, allowing for an efficient and exploratory search. Besides, it remains invariant when the number of

parameters increases for different kinds of cost functions (multimodal or valley shape). RR-PSO was the best-performing algorithm of the PSO family when using benchmark functions in different dimensions (Fernández-Martínez & García-Gonzalo, 2012). Among the rest of family members that have been introduced, CP-PSO is the most exploratory.

Therefore, both RR-PSO and CP-PSO constitute an interesting choice for performing nonlinear uncertainty analysis and exploring the cost function topography efficiently. PP-PSO has the same velocity update as GPSO, but the positions of the particles are in time t instead of $t + 1$. PP-PSO has a more exploratory character than GPSO but a lower convergence rate. Finally, CC-PSO has showed in the numerical analysis the fastest convergence rate.

All these algorithms can perform on the portfolio optimization equally well, since they have very interesting exploratory and convergence capabilities. In the present case, the particles of the swarm are the percentages of the stocks in the portfolio. These algorithms are used as samplers, by increasing the exploration of the search space. The sampling is performed in a search space, which is a hyperprism of the dimension of the number of preselected assets. The percentage of each stock varies between 0 and a maximum percentage that can be fixed by the user to enforce diversity in the portfolio. Finally, we introduce a variant where a preselected asset can be taken or not into consideration for the selection. In this case, the assets that are not considered to have lower and upper bounds in the search space set to zero. We show these results in the next section for IBEX35.

Besides, it has been shown that the PSO family members are much more efficient and simpler than other global algorithms in a wide range of inverse problems (Álvarez et al., 2019; Fernández-Martínez, García-Gonzalo, Fernández-Álvarez, Kuzma, & Menéndez-Pérez, 2010; Fernández-Martínez, García-Gonzalo, & Naudet, 2010; Fernández-Martínez, Mukerji, García-Gonzalo, & Suman, 2012; Pallero, Fernández-Martínez, Bonvalot, & Fudym, 2015, 2017). These papers are also a very good reference to better understand some interesting details about the PSO family implementation. It is important to note that most of these problems are also high-dimensional.

7.4 Application to the Portfolio Optimization Problem

Modern portfolio theory was first introduced by Markowitz (1952). This model assumes that any investor wants to maximize a portfolio's expected return contingent on any given amount of risk. The portfolios that meet this criterion are defined as efficient portfolios. This constrained optimization problem does not have a unique solution, since there is a trade-off between the risk and the expected return. The representation of the expected return-risk relationship of the different portfolios can be considered as the uncertainty region of the portfolio optimization problem. Every point in the expected return-risk region corresponds to a set of selected stocks and their corresponding weights. Therefore, the portfolio optimization problem can be

efficiently solved as a sampling problem via particle swarm optimization. This problem can be divided into several steps:

1. The selection of the stocks composing the portfolio. The selection is important due to the curse of dimensionality in inverse and optimization problems (Fernández-Martínez & Fernández-Muñiz, 2020) that hampers the sampling procedure in spaces whose dimension is greater than 5 to 10.
2. The cost function definition.
3. The optimization of the weights of each stock composing the portfolio.

7.4.1 The Portfolio Stock Selection

The selection of stocks requires several working hypotheses:

1. The choice of the stock temporality; in this case, we work with closing prices and a daily temporality.
2. Only long entries are selected. The approach could be also adapted to short entries (against the market).
3. Some restriction parameters, concerning the risk profile of the investor.

The stock selection can be written as follows, given a set of stocks for potential investment:

$$M = \{s_k(t), t \in [t_0, t_n]\}_{k=1,N}$$

finding a subset composed of a maximum number of q shares exhibiting a good equilibrium between return and risk. In this case, we propose to perform the analysis adopting a memory of 100 days. This parameter can be changed and influences the final result that has been obtained.

7.4.2 The Cost Function Definition

Given a set of q potential shares that have been selected in the previous step, the following concepts are important:

7.4.2.1 Expected Return

The expected return is the profit or loss that an investor anticipates on an investment that has known historical rates of return. This concept integrates the probability of gaining and losing:

$$r = pG + (1 - p)L, \quad (7.16)$$

where p is the probability of gaining, G is the gain, and L is the loss. The rate of return is the net gain or loss of an investment over a specified time period, expressed as a percentage of the investment initial cost.

The expected return of a portfolio is the linear combination of the expected return of the shares with the weights of the composition of the assets in the portfolio, calculated by multiplying the weight of each asset by its return and summing the values of all the assets together:

$$E_p(\mathbf{w}) = \sum_{k=1}^q w_k r_k, \quad (7.17)$$

where r_k is the return of each asset in the portfolio.

7.4.2.2 Portfolio Variance

Portfolio variance is a measure of risk of the portfolio: a higher variance indicates a higher risk for the portfolio. The variance of the portfolio is calculated as the variance of a combination of random variables, which are the returns of the shares composing the portfolio:

$$\begin{aligned} \sigma_p^2(\mathbf{w}) &= \sum_{k=1}^q w_k^2 \sigma_{k+}^2 \sum_{i=1}^q \sum_{j \neq i} w_i w_j C_{ij} = \sum_{k=1}^q w_k^2 \sigma_{k+}^2 \sum_{i=1}^q \sum_{j \neq i} w_i w_j \rho_{ij} \sigma_i \sigma_j = \\ &= \sum_{i=1}^q \sum_{j=1}^q w_i w_j C_{ij}, \end{aligned} \quad (7.18)$$

where σ_i is the variance of the stock i , ρ_{ij} is the correlation coefficient between the returns of the assets i and j , and C_{ij} is the sample covariance of these returns.

The cost function is the ratio between the expected return and the portfolio standard deviation which is a measure of its risk:

$$C(\mathbf{w}) = \frac{E_p(\mathbf{w})}{\sqrt{\sigma_p^2(\mathbf{w})}}. \quad (7.19)$$

The problem consists in a given portfolio composed of q assets, finding the set of weights $\mathbf{w} \in \mathbb{R}^q$, such that $C(\mathbf{w})$ is maximum, that is, we achieve a maximum return with a minimum risk.

The idea is that investors can reduce their exposure to individual asset risk by holding a [diversified](#) portfolio of assets. Diversification allows obtaining an expected return through a reduced risk.

7.4.2.3 Selection of the Stocks and Optimization of the Weights

Let us consider a portfolio composed of q possible investments.

The algorithm workflow is as follows:

1. Defining a swarm of N particles of q weights such that $\sum_{k=1}^q w_k = 1$.
2. For each particle of the swarm $w \in \mathbb{R}^q$, computing $E_p(w)$ and $\sigma_p^2(w)$ through the computation of the covariance matrix of the returns among the selected stocks.
3. Applying PSO and finding the set of weights that provide $C(w) > C_{\min}$, where C_{\min} is the minimum cost function value that we want to maximize. In this case, the swarm is composed by the weights of the q portfolios. The search space design is simple because the weights are real numbers between 0 and 1. In the search, there is always a compulsory normalization, such as $\sum_{k=1}^q w_k = 1$.
4. Once this sampling procedure is finished, the different portfolios are analyzed to find the [efficient frontier](#) curve in the expected return-risk sampled region. This sampling procedure in a reduced dimensional space could also be performed via other sampling and global optimization methods.

Compared to classical methods used to solve the portfolio optimization problem, such as, the critical line algorithm (CLA) (Markowitz, 1956), we have bear in mind that PSO is a search method composed by a double stochastic gradient in the model space (portfolio composition), that is, PSO does not need to perform any minimization, to calculate any gradient of the cost function, linearization, and/or inversion of the Jacobian matrix as local optimization methods need to do. The stochastic gradient is different for any particle of the swarm, and they are the differences between the particle and the global best and the local best of each particle in each iteration.

The only requirement for the PSO family is to provide the search space for the portfolio compositions, which is a hyperprism of the dimension of the number of assets that have been preselected. Besides, the solution of the forward problem, that is, the cost function calculation for any particle of the swarm (portfolio composition), should be fast, as it is the case. All additional constraints that are part of the optimization problem are introduced as penalties in the objective function. Besides, any a priori (or initial) solution of the portfolio composition can be easily introduced as an additional particle of the swarm to check if this solution might be improved. Nevertheless, the PSO sampling is not aimed at finding the global optimum but at sampling the posterior distribution. Therefore, it can be concluded that PSO is well suited to address efficiently the portfolio optimization problem.

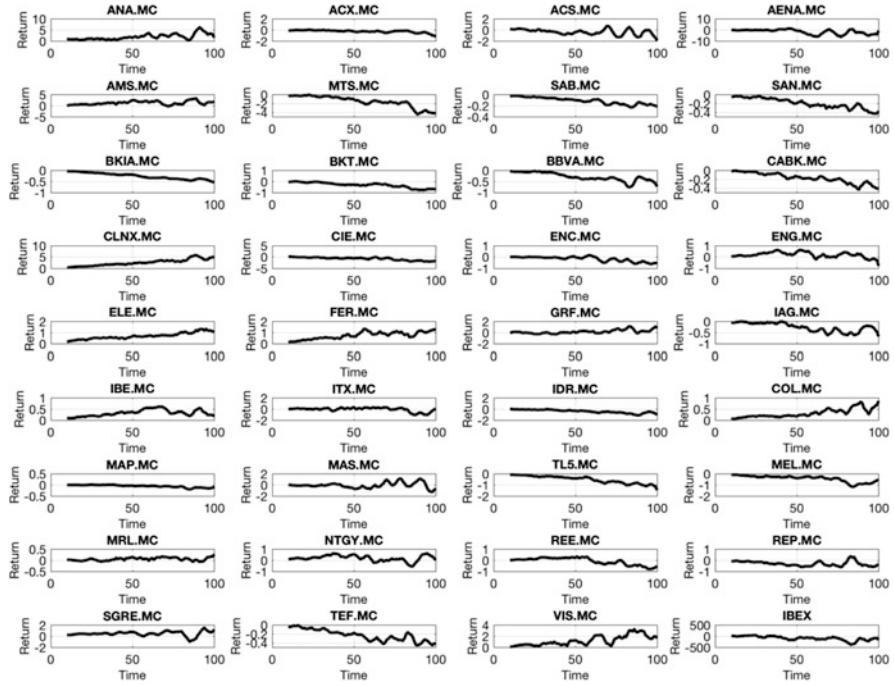


Fig. 7.1 Expected return moving window analysis for IBEX35 shares. Source: Author's own creation

This algorithm can be used for different approaches of the portfolio optimization problem that use different definitions of the cost function to measure the risk and the expected return, as it has been pointed out in the introduction.

Figure 7.1 shows the expected return for the different assets of IBEX35 (including the average index in different windows for time lengths between 1 week (5 days) and 100 days). Then, the median return is calculated, and a proxy of the risk is established by calculating the distance between the maximum and minimum returns obtained in this analysis. In this case, the risk is defined as a maximum derivative of the return. This is a more robust method for selecting the potential assets than performing the analysis for a given time frame, for instance, 1 year. Figure 7.2 shows the Markowitz's ratio of the stocks of IBEX35. To produce this plot, we have performed a historical analysis of the returns of the different assets (see Fig. 7.1). Note that in this case, the ratios are given with their sign and can be negative if the median expected return is.

Figure 7.3 shows the medium expected returns and risks that serve to calculate this plot. These are the two views of this plot. It can be observed that, although ELE.MC has no highest return, its lower-medium risk makes this asset a very interesting investment. This analysis served to select the assets with the highest Markowitz's ratio. Table 7.2 shows the list of nine companies obtained from this analysis. We also

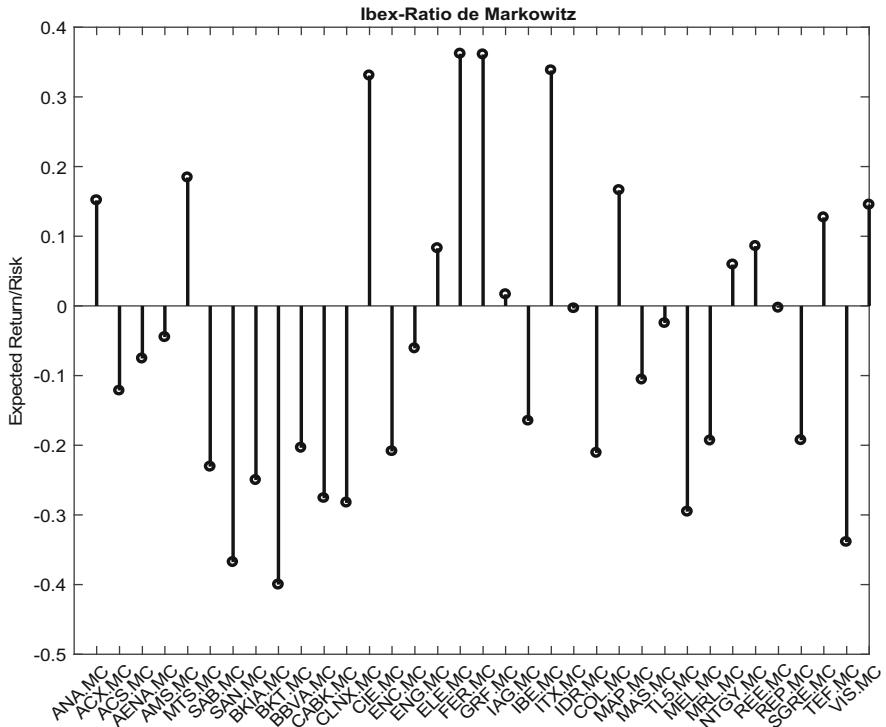


Fig. 7.2 Markowitz's ratio for the stocks of IBEX35. Source: Author's own creation

provide the proxy of the Markowitz ratio and the best composition obtained after optimization by using all the stocks.

Concerning the portfolio optimization within this list via PSO, the optimization tries to maximize expression (7.19). The result provides the weights of the portfolio. Figure 7.4 shows the Markowitz's ratio of portfolio risk region. It can be observed the Pareto front shape. What it is interesting in this approach is that the maximum Markowitz's ratio, calculated following Eqs. (7.18) and (7.19), coincides with the minimum portfolio risk.

This feature is due to the preselection of the stocks before performing the weight optimization. Figures 7.5 and 7.6 show the other two views of the portfolio optimization: portfolio return-portfolio risk and Markowitz's ratio. Figure 7.5 shows the portfolio return-risk region, which is almost symmetric around the axis of constant portfolio return equal to 0.9, highlighting the importance of the portfolio weight optimization. Therefore, risk increase does not compulsorily provide better outcomes, that is, higher portfolio returns.

Figure 7.6 shows that the maximum Markowitz's ratio does not correspond to the maximum portfolio return. The region has almost a triangular shape, symmetrical around the constant portfolio return equal to 0.9. Therefore, risk tuning is crucial. Finally, Fig. 7.7 shows the histograms of the best compositions of the portfolios.

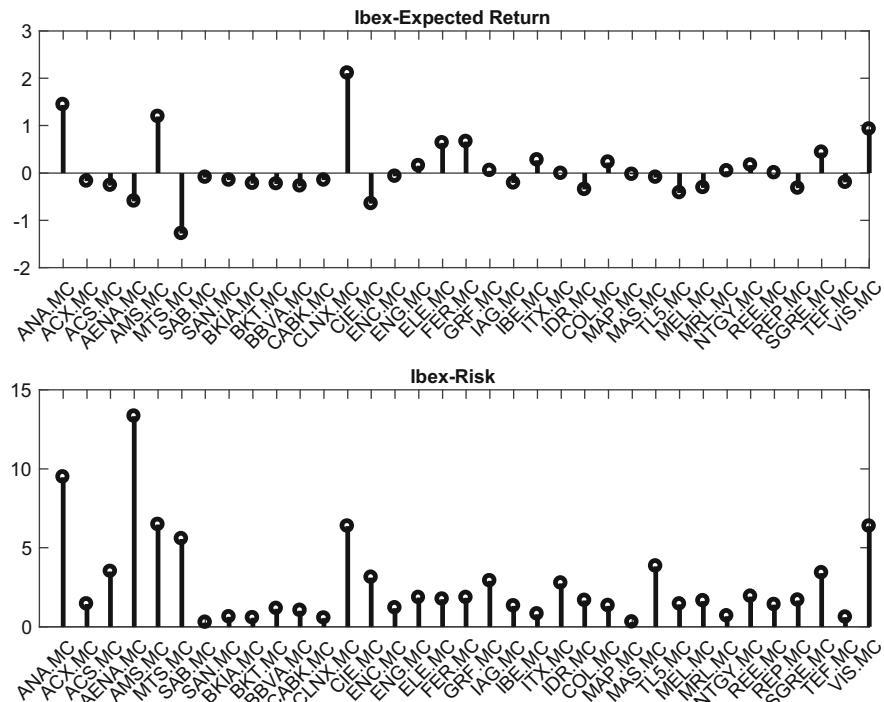


Fig. 7.3 Medium returns and risk of the different assets in IBEX35. Source: Author's own creation

Table 7.2 Selected assets in IBEX35 with highest Markowitz's ratio

Asset	Markowitz ratio (proxy)	Best composition (%)
"ELE.MC"	[0.3617]	10.23
"FER.MC"	[0.3608]	13.43
"IBE.MC"	[0.3382]	12.96
"CLNX.MC"	[0.3307]	12.47
"AMS.MC"	[0.1842]	11.90
"COL.MC"	[0.1660]	8.26
"ANA.MC"	[0.1516]	11.17
"VIS.MC"	[0.1450]	13.59
"SGRE.MC"	[0.1268]	6.00

These histograms contain the best portfolio composition shown in Table 7.1. We believe that giving this kind of information provides a better idea about the limits of the portfolio composition. It can also be observed that all the histograms show similar extents and modes. This analysis shows that none of the assets should have in any case a weight higher than 20%.

A different possibility that also offers this global optimization approach is to optimize the selection of the stocks and at the same time the optimization of the

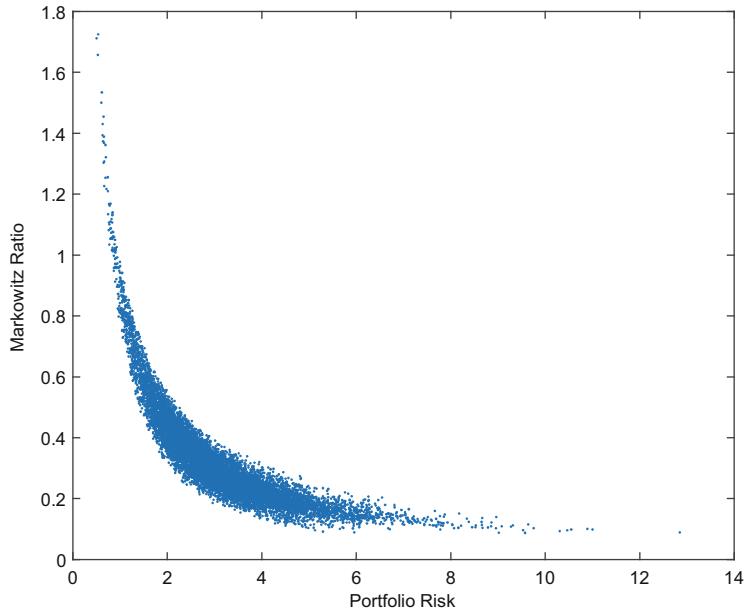


Fig. 7.4 Markowitz ratio-Portfolio risk region obtained by PSO sampling. Source: Author's own creation

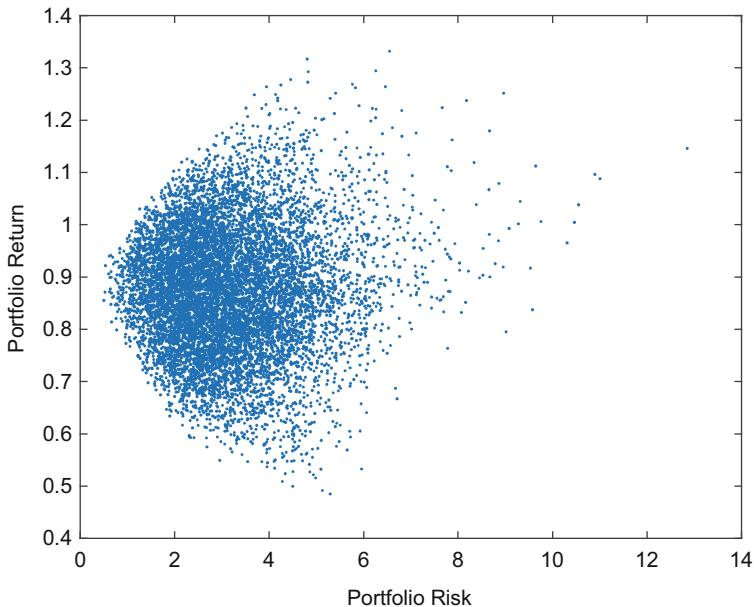


Fig. 7.5 The portfolio of return-risk region for the number of preselected portfolios (constrained optimization). Source: Author's own creation

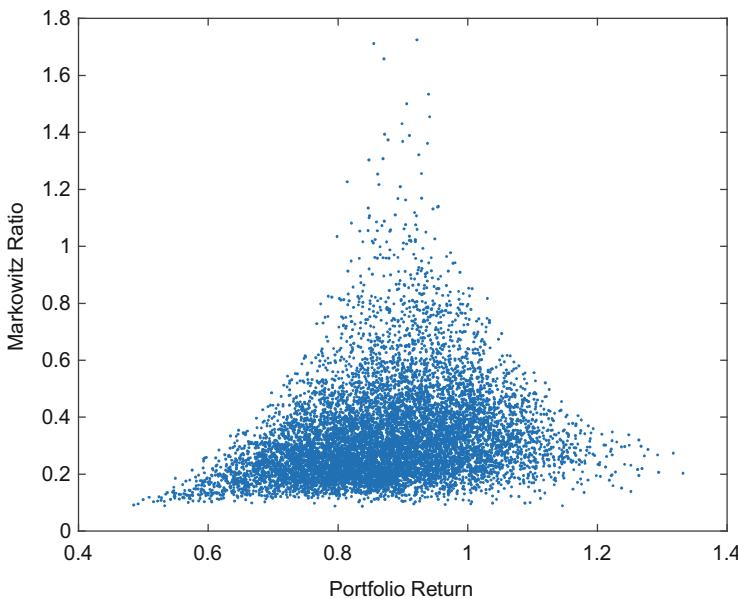


Fig. 7.6 Markowitz ratio and portfolio risk region. Source: Author's own creation

portfolios. In this approach, not all the preselected stocks will take part in the portfolio optimization. The procedure allows to consider a minimum number of assets to take part in the portfolio, for instance, one. This might be interesting in very unstable situations of the stock market. Figure 7.8 shows the views of the new regions commented previously, which have a more complex shape showing different basins of the cost function landscape. In this particular case, the optimum portfolio was just composed by one asset: CNLX.

7.5 Conclusions

This paper shows the application of the PSO family to the portfolio optimization problem. First, we have introduced the topography of the cost function in high-dimensional optimization problems, concluding the existence of different equivalent models with a similar cost function value. All these models belong to the nonlinear region of equivalence of the optimization problem that has a curvilinear shape and might be composed of different isolated valleys. Besides, we show that the linearized hyperquadric, which is oriented by the Hessian matrix, only spans the nonlinear equivalence region locally, that is, in the neighborhood of the current portfolio weights. This theoretical development highlights the importance of sampling the nonlinear region of equivalence, which in the case of the portfolio optimization is directly related to the efficient portfolio region, first described by Markowitz.

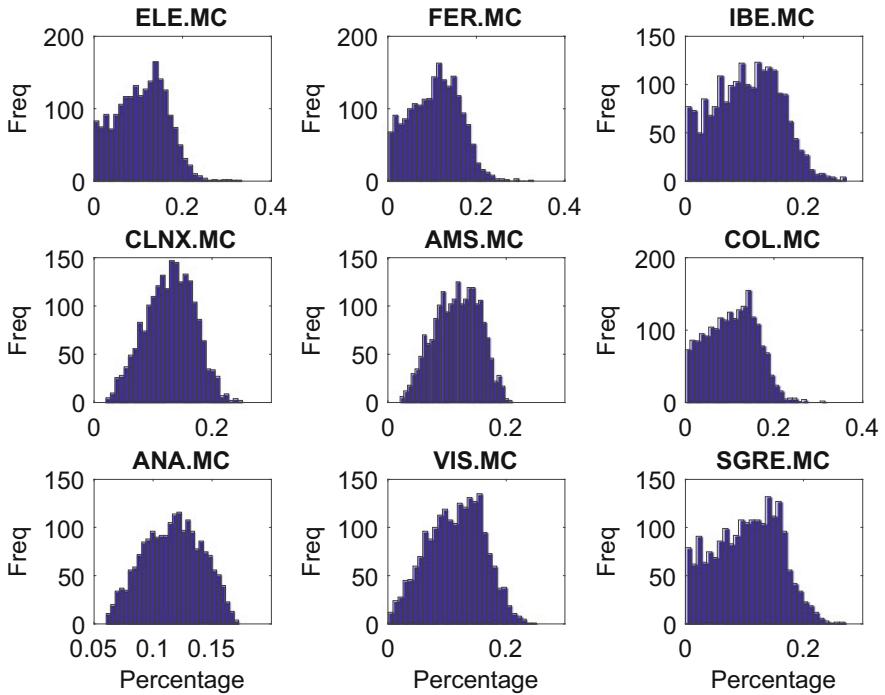


Fig. 7.7 Histograms of the percentages (compositions) of the best portfolios. These histograms are obtained using the particles of the swarm (portfolios compositions) with the lower misfits. Source: Author's own creation

Although the development is given for minimization problems, it can be also generalized for maximization as well by switching the sign in the definition of the cost function. In a second well-differentiated part, we introduced particle swarm optimization and the family of PSO optimizers that were analyzed in previous research, explaining how the tuning of the PSO parameters in the neighborhood of their second-order stability region is a compulsory condition to perform a correct sampling of the non-equivalence region of the portfolio optimization problem. Finally, we have presented the application of the methodology to the assets of IBEX35 via PSO. Although PSO can easily handle 35 dimensions and even more, to reduce the dimension and allow for a fast optimization, first, we have preselected the assets on which the optimization is performed. The aim is to find the assets with the highest potential, being the rest of the assets not optimum for selection and optimization. We have presented two different approaches on which the weights and the assets are optimized.

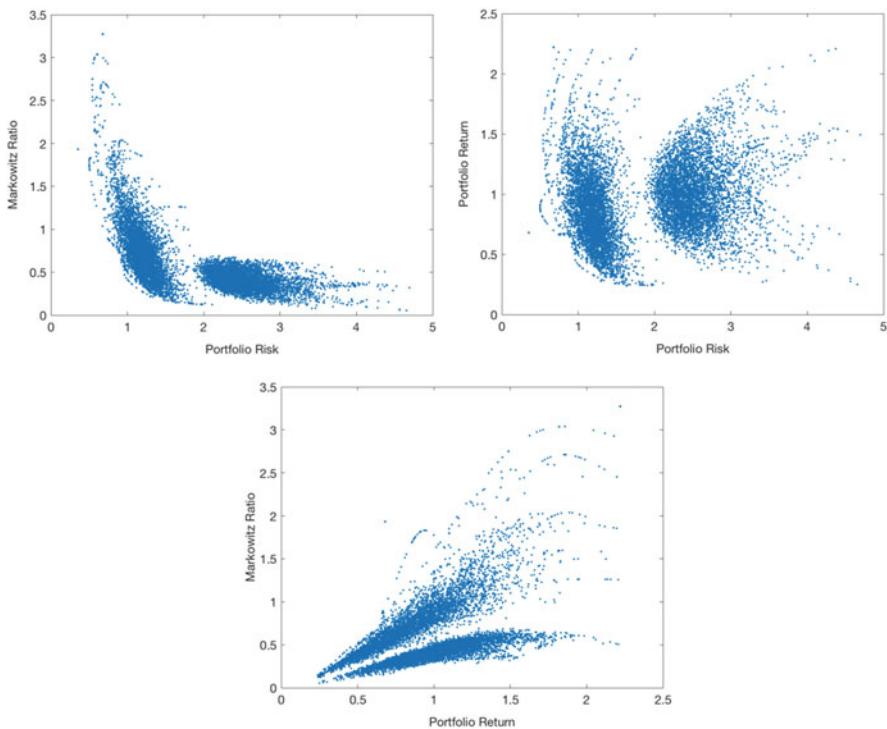


Fig. 7.8 New regions for the variable number of the selected assets of the portfolio optimization problem. In this case, PSO performs at the same time the selection of the stocks and the optimization of their weights. Source: Author's own creation

References

- Álvarez, Ó., Fernández-Martínez, J. L., Corbeanu, A. C., et al. (2019). Predicting protein tertiary structure and its uncertainty analysis via particle swarm sampling. *Journal of Molecular Modeling*, 25, 79.
- Chang, T. J., Yang, S., & Chang, K. (2009). Portfolio optimization problems in different risk measures using genetic algorithm. *Expert Systems with Applications*, 36, 10529–10537.
- Cura, T. (2009). Particle swarm optimization approach to portfolio optimization. *Nonlinear Analysis: Real World Applications*, 10, 2396–2406.
- Fernández-Martínez, J. L., & Fernández-Muñiz, Z. (2020). The curse of dimensionality in inverse problems. *Journal of Computational and Applied Mathematics*, 369, 112571–112584.
- Fernández-Martínez, J. L., Fernández-Muñiz, Z., & Tompkins, M. J. (2012). On the topography of the cost functional in linear and nonlinear inverse problems. *Geophysics*, 77(1), W1–W15.
- Fernández-Martínez, J. L., & García-Gonzalo, E. (2008). The generalized PSO: A new door to PSO evolution. *Journal of Artificial Evolution and Applications*, 2008, 1–15.
- Fernández-Martínez, J. L., & García-Gonzalo, E. (2009). The PSO family: Deduction, stochastic analysis and comparison. *Swarm Intelligence*, 3, 245–273.
- Fernández-Martínez, J. L., & García-Gonzalo, E. (2011). Stochastic stability analysis of the linear continuous and discrete PSO models. *IEEE Transactions on Evolutionary Computation*, 15, 405–423.

- Fernández-Martínez, J. L., & García-Gonzalo, E. (2012). Stochastic stability and numerical analysis of two novel algorithms of PSO family: PP-PSO and RR-PSO. *International Journal on Artificial Intelligence Tools*, 21, 1240011–1240029.
- Fernández-Martínez, J. L., & García-Gonzalo, E. (2013). Particle swarm optimisation: Time for uniformisation. *International Journal of Computing Science and Mathematics*, 4(1), 16–33.
- Fernández-Martínez, J. L., García-Gonzalo, E., & Fernández-Alvarez, J. P. (2008). Theoretical analysis of Particle Swarm trajectories through a mechanical analogy. *International Journal of Computational Intelligence Research*, 4, 93–104.
- Fernández-Martínez, J. L., García-Gonzalo, E., Fernández-Álvarez, J. P., Kuzma, H. A., & Menéndez-Pérez, C. O. (2010). PSO: A powerful algorithm to solve geophysical inverse problems. Application to a 1D-DC resistivity case. *Journal of Applied Geophysics*, 71, 13–25.
- Fernández-Martínez, J. L., García-Gonzalo, E., & Naudet, V. (2010). Particle swarm optimization applied to solving and appraising the streaming-potential inverse problem. *Geophysics*, 75(4), WA3–WA15.
- Fernández-Martínez, J. L., Mukerji, T., García-Gonzalo, E., & Suman, A. (2012). Reservoir characterization and inversion uncertainty via a family of particle swarm optimizers. *Geophysics*, 77, M1–M16.
- Fernández-Martínez, J. L., Pallero, J. L. G., Fernández-Muñiz, Z., & Pedruelo-González, L. M. (2014a). The effect of noise and Tikhonov's regularization in inverse problems. Part I: The linear case. *Journal of Applied Geophysics*, 108, 176–185.
- Fernández-Martínez, J. L., Pallero, J. L. G., Fernández-Muñiz, Z., & Pedruelo-González, L. M. (2014b). The effect of noise and Tikhonov's regularization in inverse problems. Part II: The non-linear case. *Journal of Applied Geophysics*, 108, 186–193.
- García-Gonzalo, E., & Fernández-Martínez, J. L. (2012). A brief historical review of particle swarm optimization (PSO). *Journal of Bioinformatics and Intelligent Control*, 1(1), 3–16.
- García-Gonzalo, E., & Fernández-Martínez, J. L. (2014). Convergence and stochastic stability analysis of particle swarm optimization variants with generic parameter distributions. *Applied Mathematics and Computation*, 249, 286–302.
- Kennedy, J., & Eberhart, R. C. (1995). Particle swarm optimization. *Proceedings IEEE International Conference on Neural Networks*, 4, 1942–1948.
- Markowitz, H. M. (1952). Portfolio Selection. *The Journal of Finance*, 7(1), 77–91.
- Markowitz, H. M. (1956). The optimization of a quadratic function subject to linear constraints. *Naval Research Logistics Quarterly*, III, 111–133.
- Pallero, J. L. G., Fernández-Martínez, J. L., Bonvalot, S., & Fudym, O. (2015). Gravity inversion and uncertainty assessment of basement relief via particle swarm optimization. *Journal of Applied Geophysics*, 116, 180–191.
- Pallero, J. L. G., Fernández-Martínez, J. L., Bonvalot, S., & Fudym, O. (2017). 3D gravity inversion and uncertainty assessment of basement relief via particle swarm optimization. *Journal of Applied Geophysics*, 139, 338–350.
- Sortino, F. A., & Price, L. N. (1994). Performance measurement in a downside risk framework. *Journal of Investing*, 3, 50–58.
- Tyrrell Rockafellar, R., & Uryasev, S. (2000). Optimization of conditional value-at-risk. *Journal of Risk*, 2(3), 21–42.
- Wolfe, P. (1959, July). The simplex method for quadratic programming. *Econometrica*, 27(3), 382–398.
- Wolpert, D. H., & Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1, 67.
- Xu, F., Chen, W., & Yang, L. (2007). Improved Particle Swarm Optimization for realistic portfolio selection. In *Eighth ACIS international conference on software engineering, artificial intelligence, networking, and parallel/distributed computing* (pp. 185–190). New York: IEEE Computer Society.

Chapter 8

A Constrained Portfolio Selection Model Solved by Particle Swarm Optimization Under Different Risk Measures



Akbar Esfahanipour and Pouya Khodaee

Abstract Portfolio selection has been one of the crucial problems in financial engineering. Investors' interest is to construct a portfolio having a balance between the investor's risk-taking and his/her expectations about the portfolio returns. The Markowitz model is a nonlinear constrained multi-objective optimization model that is usually impossible to solve at a good time. In this chapter, the purpose is to examine portfolio optimization models and applications of the particle swarm optimization (PSO) technique in solving these models. A constrained portfolio selection model has been developed, which is solved by the PSO technique as a metaheuristic approach using data from the Tehran Stock Exchange (TSE) to assess the developed model. In this case, the effects of three different risk measures have been analyzed on the constructed portfolios. The numerical results show that conditional value at risk (CVaR) performs better than the other two risk measures, including semivariance and variance. However, from the diversification perspective, the model with the variance risk measure produces a more diversified portfolio compared to the other two risk measures, although the differences are trivial.

Keywords Constrained portfolio optimization · Particle swarm optimization (PSO) · Risk measure · Tehran Stock Exchange (TSE)

8.1 Introduction

The construction of investment portfolios has been one of the crucial problems in financial engineering. Making a balance between risk level and expected return introduces the concept of investment portfolio optimization (PO), in which Markowitz pioneers the construction of its primary mean-variance model (Markowitz, 1959). Based on this model, researchers have done extensive work on this model,

A. Esfahanipour (✉) · P. Khodaee
Amirkabir University of Technology, Tehran, Iran
e-mail: esfahaa@aut.ac.ir; pouyakhodaee@aut.ac.ir

mainly due to considering real situations. Extensions are more about different risk measures such as semivariance, MAD,¹ VaR,² CVaR, and spectral risk measures, which are described in Mao (1970); Chang, Meade, Beasley, and Sharaiha (2000); and Celikyurt and Özekici (2007). In other developments, different constraints have been considered to deal with more realistic investment opportunities. Typical constraints are about the number of assets in a portfolio, transaction costs, and short sales, which have been studied by Markowitz and Chang, among others (Markowitz, 1959) (Chang et al., 2000). Markowitz's basic model and its extensions have a nonlinear structure and consist of multi-objective functions that make it very difficult to find optimal solutions with traditional methods and techniques in a good time, especially when there are a lot of financial assets to be considered. That is why researchers have resorted to metaheuristic approaches to solve this problem to achieve optimal/near-optimal solutions in a short time.

Particle swarm optimization (PSO) has been applied as a metaheuristic approach to solve these optimization problems. Researchers have proposed different types of PSOs from different perspectives. Eberhart suggests the PSO technique as a population-based metaheuristic algorithm to solve multi-objective optimization problems. These algorithms retain and enhance multiple possible solutions, often using population features to help the search process (Kennedy & Eberhart, 1995). In PSO algorithms, the topology of the population directly affects the mechanisms of information sharing between particles; thus, it affects the solution's performance. Clerc proposes a method to produce population topologies in a random manner (Clerc, 2007). The research mentioned above is on population topology to improve performance so that the PSO technique can be utilized in more areas of the search space. In optimizing particle swarm, a particle population in a multidimensional search space seeks the best values. Each particle's position is a possible solution that can be evaluated by its performance, and the best position of the particle and the best position of the population are updated based on the current position. During the moving process, the particles converge at the optimal point of a fitting function. The PSO algorithm is very convenient to use and experimentally has been shown that performs better in a lot of optimization problems. However, when solving complex multi-objective problems, the algorithm may get trapped into a local optimal. Due to these drawbacks, different PSOs have been proposed to improve the performance of the basic PSO algorithm. Extended models of the PSO technique have enhanced its ability to moderate a wide range of intricate engineering and science of optimization issues, improving its final performance.

Another aspect influencing the portfolio optimization problem is to opt for a suitable risk measure. Risk measures are statistical measures used for risk evaluation of a set of assets based on changes in those assets' historical prices. Investors usually use these measures to be able to control the risk limits of their investments. Many risk measures have been studied in Momen, Esfahanipour, and Seifi (2019) and

¹Mean absolute deviation.

²Value at risk.

Esfahanipour and Mousavi (2011) in detail. The authors in this chapter examine some of these measures in the literature section. Regarding a diverse set of risk measures, investors decide to allot a different portion of assets to obtain maximum return while minimizing the risk. This chapter reports a case study in which the performance of varying risk measures has been evaluated to select optimal stock portfolios in the Tehran Stock Exchange.

This chapter aims to investigate the investment portfolio optimization problem with constraints and apply the PSO technique to solve this problem. To that end, a base portfolio optimization model is selected to develop and solving utilizing the PSO technique. This chapter's developed model is implemented using data of 20 stocks listed on the Tehran Stock Exchange as a case study. In this implementation, the effects of different risk measures are examined with the suggested portfolios from risk-adjusted return and diversification point of views.

The rest of this chapter has been organized as described next. First of all, the authors review the literature of (1) portfolio optimization problem, (2) particle swarm optimization technique, and (3) risk measures; besides, they report different studies that apply the PSO technique to solve constrained portfolio optimization problems as well as some portfolio optimization models. Then, the authors present details of a base portfolio optimization model to develop and implement in this study. A constrained portfolio optimization model has been developed based on the selected base model from the literature, which is solved by the PSO technique using data of the Tehran Stock Exchange (TSE) as an emerging market. To sum up, this chapter ends with some concluding remarks and possible future research directions in this regard.

8.2 Literature Review

In this section, the portfolio optimization models and the PSO technique, especially its applications in optimizing portfolio models, are discussed.

8.2.1 *Portfolio Optimization (PO) Problem*

Markowitz divides the steps of opting for investment portfolios into two phases (Markowitz, 1959):

- (a) Firstly, the selection is started with observation and experience and is terminated with some confidence about existing assets' future.
- (b) Secondly, the selection is continued with the related confidence about the future and is terminated with opting for a portfolio.

Markowitz contends that an investor should consider his/her expected return of a portfolio as a desirable goal and the portfolio return's variance return as an

unfavorable goal. He shows the geometric relationship between this behavior and the selection of investment portfolios due to the “expected return-variance of returns”³ rule. An investor wants to maximize the discounted value of future returns. Since the future is uncertain, it must be “expected” or “anticipated” returns that need to be discounted. In other words, investors should consider expected returns as an allowance level of risk, which is defined as a variation type of the Markowitz model. The hypothesis of efficient market is regarded as a simplification assumption for modeling and solving the model. This model does not necessarily suggest that a diverse portfolio is preferable to all non-diversified portfolios. The basic model fails to indicate diversity because it does not consider how the expected return is formed and whether the same or various discount rates are applied for multiple assets. It is not essential how these discount rates are decided upon or how they change during time. This hypothesis suggests that an investor invests all of his/her capital in securities having the most significant, discounted value. It also states that if two or more assets have equal value, each of them or any composition of them is the same.

Considering the assumptions mentioned above, the model which is applied by Markowitz for presenting his proposed model is as follows:

$$\text{Min} \sum_{i=1}^N \sum_{j=1}^N w_i w_j \sigma_{ij} \quad (8.1)$$

$$\text{s.to} \sum_{i=1}^N w_i r_i = R^* \quad (8.2)$$

$$\sum_{i=1}^N w_i = 1 \quad (8.3)$$

$$0 \leq w_i \leq 1 \quad i = 1, 2, \dots, N \quad (8.4)$$

where N is the number of assets, σ_{ij} is the covariance between assets i and j , w_i is the weight of each asset in the portfolio, r_i is the expected return of asset i , and R^* is the desired expected return of the portfolio.

Based on the mean-variance model, Salehpoor and Molla-Alizadeh-Zavardehi (2019) apply three risk measures of MAD,⁴ SV⁵ and VWS⁶ to develop their algorithms for portfolio selection. The developed algorithms include EM,⁷ PSO, GA,⁸ GNP,⁹ and SA.¹⁰ Moreover, a diversification strategy has been utilized and combined with the developed algorithms to enhance diversity and overcome local optima. The authors validate their model using data of 20 companies in the

³Mean-variance.

⁴Mean absolute deviation.

⁵Semivariance.

⁶Variance with skewness.

⁷Electromagnetism-like algorithm.

⁸Genetic algorithm.

⁹Genetic network programming.

¹⁰Simulated annealing.

Tehran Stock Exchange. Their results indicate that the SA acts much better than the other techniques, which were evaluated regarding mean-variance, MAD, SV, and VWS as risk measures.

Babazadeh and Esfahanipour (2019) develop a model for portfolio optimization in which the VaR is used as a risk measure, which is estimated by using extreme value theory (EVT). A set of trading constraints have been considered, such as cardinality, budget, floor, and ceiling constraints, making the proposed model more realistic. Since these constraints have given rise to non-convex solution space, this problem is classified as NP-hard problems. A new design of the GA, the NSGA-II,¹¹ has been proposed to solve their proposed model. Using data from the S&P 100 indices, their results show that their proposed algorithm has an extraordinary ability to solve the PSO technique in the mean value at risk (VaR) problem. The results also show that their model in the low-risk area performs better than the Pareto border. Moreover, the NSGA-II algorithm has a significant improvement in solving time.

8.2.2 *Particle Swarm Optimization (PSO)*

This section describes the PSO technique's applications in the portfolio optimization problem and the main steps of its implementation. Chen, Zhang, Cai, and Xu (2006) propose a development for the Markowitz model in which transaction costs and floor and ceiling restrictions are considered. Due to additional constraints, Chen's model is more sophisticated than the Markowitz model, so traditional optimization algorithms cannot find the optimal solution efficiently. Thus, they use the PSO algorithm to solve the problem. They also provide a numerical implementation of the problem for selecting investment portfolios to indicate the effectiveness of the proposed model (Chen et al., 2006).

Cura (2009) uses the mean-variance model with the cardinality constraint as the investigated model. She applies the PSO technique to solve this investment portfolio optimization model as an integer quadratic programming problem. Her research's numerical results show that the PSO technique is succeeded in optimizing investment portfolios in comparison with the application of the metaheuristic approaches of GA, SA, and TS.¹²

Zhu, Wang, Wang, and Chen (2011) present a metaheuristic approach for investment portfolio optimization using the PSO technique. Their model was tested on two types of portfolios, which have considered restricted and unrestricted risk. Additionally, a comparative study was done with the genetic algorithm. Since the portfolio optimization model has been selected as a base model to develop and

¹¹Non-dominated sorting genetic algorithm.

¹²Tabu search.

implement in the present study's case study, the authors illustrate more details about this model in the next section entitled *The Base Portfolio Optimization Model*.

To solve an investment portfolio optimization model with the generalized mean-variance model, Yin, Ni, and Zhai (2015) present four algorithms of (RT-AD),¹³ (RT-D),¹⁴ (DRT-AD),¹⁵ and (DRT-D)¹⁶ to improve PSO method based on random topology strategies. They have abstracted the PSO method's topology into an undirected connected graph, which can be generated randomly with a predetermined degree. They contend that when a dynamic population topology strategy is adopted, the topology changes. By determining this degree, the communication mechanisms in the evolutionary period can be controlled, and the performance of the solving PSO algorithms can be improved. Their numerical results show that the PSO population topology directly affects data sharing between particles, so the PSO method's performance will improve. In particular, their proposed algorithm (DRT-D) demonstrates outstanding performance and provides an effective solution to investment portfolios optimizing.

The model that they chose to optimize the investment portfolio is the CCMV¹⁷ model, which is a generalized version of the Markowitz model and is generally defined as follows:

$$\text{Min } \lambda \left[\sum_{i=1}^N \sum_{j=1}^N z_i w_i z_j w_j \sigma_{ij} \right] - (1 - \lambda) \left[\sum_{i=1}^N z_i w_i r_i \right] \quad (8.5)$$

$$\text{s.to } \sum_{i=1}^N w_i = 1 \quad (8.6)$$

$$\sum_{i=1}^N z_i = K \quad (8.7)$$

$$z_i \leq w_i \leq \delta_i z_i \quad i = 1, \dots, N \quad (8.8)$$

$$z_i \in \{0, 1\} \quad i = 1, \dots, N \quad (8.9)$$

The above model is different from the basic Markowitz model in two (Eq. 8.7) constraints and (Eq. 8.8). Constraint (Eq. 8.7) shows the number of assets in a portfolio that cannot be greater than a known number of K . Constraint (Eq. 8.8) shows the upper and the lower limits for the number of assets in the portfolio so that z_i as a zero-one variable $z_i \in \{0, 1\}$, it will become one if the asset i is in the portfolio, and it will become zero if the asset i is not in the portfolio (Yin et al., 2015).

Zaheer, Abd Aziz, Kashif, and Raza (2018) present simple models for selecting and optimizing investment portfolios. They also develop a hybrid algorithm based on the PSO technique to obtain better solutions. They consider the mean-variance

¹³Random population topology based on the average degree.

¹⁴Random population topology based on the degree.

¹⁵Dynamic random population topology based on the average degree.

¹⁶Dynamic random population topology based on the degree.

¹⁷Cardinality constrained mean-variance.

concept, which presents mean return as a benefit and variance of return as the risk measure. Besides, the Sharpe ratio (SHR) was considered as the objective function of the optimization problem. They use the Shanghai Stock Exchange data to evaluate their models. The algorithm of HPSO¹⁸ is a metaheuristic algorithm that is applied in their study. Their study considers the budget a constraint and examines the two versions of the model with and without short sales. The results indicate that the techniques, as mentioned above, perform well in obtaining the optimal solution and in solution time. They develop a two-stage model of selecting and optimizing investment portfolios that can be used by investors. The developed two-stage model is described next, which is based on the following model.

$$\max R_i \quad i = 1, \dots, N \quad (8.10)$$

$$\min V_R \quad i = 1, \dots, N \quad (8.11)$$

$$\bar{R}_i = \frac{\sum_{i=1}^D R_{di}}{D} \quad d = 1, \dots, D \quad (8.12)$$

$$R_{di} = \frac{CP_d - CP_{d-1}}{CP_{d-1}} \quad (8.13)$$

$$V_R = \frac{\sum_{i=1}^D (R_{di} - \bar{R}_i)^2}{D} \quad d = 1, \dots, D \quad (8.14)$$

R_i is the mean return of asset i , V_i is the variance of the return of asset i , R_{di} is the daily return for asset i , and CP_d is the close price of the day d .

The Eqs. (8.10)–(8.14) models can be rewritten in two ways of (a) average conditions with the Sharpe ratio and (b) variance conditions with the Sharpe ratio as follows.

(a) Average conditions with the Sharpe ratio

$$\max R_i \quad i = 1, \dots, N \quad (8.15)$$

$$\max \text{SHR} \quad (8.16)$$

$$\text{S.to} \sum_{i=1}^N w_i = 1 \quad (8.17)$$

$$0 \leq w_i \leq 1 \quad i = 1, 2, \dots, N \quad (8.18)$$

(b) Variance conditions with the Sharpe ratio

$$\min V_R \quad i = 1, \dots, N \quad (8.19)$$

¹⁸Hybrid particle swarm optimization.

$$\max \text{SHR} \quad (8.20)$$

$$\text{S.to} \sum_{i=1}^N w_i = 1 \quad (8.21)$$

$$0 \leq w_i \leq 1 \quad i = 1, 2, \dots, N \quad (8.22)$$

The above models are part of the two-stage model in which, at the first step, the mean of return is maximized regarding the SHR, which is used as a fitness function. In the second step, the variance is minimized regarding the SHR as well.

8.2.3 Different Risk-Adjusted Measures

Esfahanipour and Mousavi (2011) categorize risk-adjusted measures by analyzing different studies, which are described as follows:

- (a) **Classic risk measures:** This category consists of Jensen's α , Treynor's ratio, the Sharpe ratio, and the information ratio based on CAPM¹⁹ and portfolio theory.
- (b) **Drawdown risk measures:** Sterling ratio, Calmar ratio, and Burke ratio create this group based on drawdowns. A drawdown is a loss occurring during a particular investment period. A drawdown measures the surplus return in comparison with the drawdown period.
- (c) **Lower partial moment risk measures:** The Sortino ratio, omega, kappa 3, and upside-potential ratio have been categorized in this group. The n th order of the kappa index calculates the extra return as the difference between the return on securities and the minimum acceptable return. It calculates the risk as the n th lower partial moment.
- (d) **VaR risk measures:** This category includes conditional Sharpe ratio, R-ratio, modified Sharpe ratio, excess return on VaR, MVaR,²⁰ and CVaR. These measures are based on the ratio of the expected excess return of securities over the risk-free rate and the risk.

Esfahanipour and Mousavi (2011) use the conditional Sharpe ratio for their study because it uses the CVaR as a coherent risk measure. For the same reason, in this chapter, CVaR has been used as one of the risk measures.

In Table 8.3, which is shown in Appendix, the reviewed studies have been summarized on applications of the PSO technique in the portfolio optimization problem.

¹⁹Capital asset pricing model.

²⁰Modified VaR.

8.3 The Base Portfolio Optimization Model

This section presents a base portfolio optimization model selected for developing and implementing in the Tehran Stock Exchange as a case study reported in this chapter. This base model has been chosen by Zhu et al. (2011) as a mean-variance portfolio optimization model, which is solved by the PSO technique. Based on the mean-variance model, there are two generalizations for modeling a single-objective function, as described below.

(a) Efficient Frontier (EF)

Due to having different objective function values for modifying (R^*) values, empirical studies introduce a risk aversion parameter of $\lambda \in [0, 1]$. Due to this new parameter, the model can be rewritten as follows:

$$\text{Min } \lambda \left[\sum_{i=1}^N \sum_{j=1}^N w_i w_j \sigma_{ij} \right] - (1 - \lambda) \left[\sum_{i=1}^N w_i r_i \right] \quad (8.23)$$

$$\text{s.to} \sum_{i=1}^N w_i = 1 \quad (8.24)$$

$$0 \leq w_i \leq 1 \quad i = 1, \dots, N \quad (8.25)$$

When the value of λ is zero, the model maximizes the portfolio return regardless of the risk. When the value of λ is one, the model minimizes the portfolio risk irrespective of the return. By changing the value of λ , a curve can be drawn showing the relationship between the portfolio return and the portfolio risk, called the efficient frontier (EF).

(b) Sharpe Ratio (SHR)

The Sharpe ratio is obtained by combining information of the mean-variance in an asset. The Sharpe ratio (SHR) is a risk adjustment measurement commonly used to evaluate portfolio performance as follows:

$$\text{SR} = \frac{R_p - R_f}{\text{StdDev}(p)} \quad (8.26)$$

where P is a portfolio, R_P is the mean return of the portfolio P , R_f is the risk-free rate of return, and $\text{StdDev}(P)$ is the standard deviation of the R_P , which is a portfolio risk measure. By modifying w_i , we can diminish the amount of risk while maximizing the expected return. Zhu uses the SHR as a risk measure for portfolio selection, which is solved by the PSO. According to Zhu et al. (2011), PSO implementation includes the following four steps.

(i) Initiation Step

Initially, some particles are known to be the best particles among neighboring particles due to their fitness function. Then all the particles accelerate in the direction of these particles and also in the order of their best solutions they have found before. The PSO technique's central concept

is to accelerate the movement of each particle toward the best place (bestpos) that has been obtained by this particle till that time and the best place that has ever been procured by neighboring particles (N_bestpos). With random weighting, it is fast to move toward one of these directions at any time. Particles update their positions due to the following steps:

- **Step 1:** The instantaneous position $\vec{X}(t)$
- **Step 2:** The instantaneous speeds $\vec{V}(t)$
- **Step 3:** The distance between the bestpos and the current position

$$\overrightarrow{(P_i - \vec{X}(t))}$$
- **Step 4:** The distance between the N_bestpos and the current position

$$\overrightarrow{(P_G - \vec{X}(t))}$$

(ii) Calculate Fitness Function

Each particle in the population of PSO has a unique fitness value. According to its previous position, a particle moves inside the solution space where it has seen the best fitness value (bestpos) and its last neighbor position, which has seen the best fitness value (N_bestpos). In Zhu et al. (2011), the Sharpe ratio is used as a single-objective function, which is defined as follows:

$$f_p = \text{SR} = \frac{\sum_{i=1}^N w_i r_i - R_f}{\sum_{i=1}^N \sum_{j=1}^N w_i w_j \sigma_{ij}} \quad (8.27)$$

where f_p is the fitness value for each particle. Bestpos and N_bestpos are changed if an enhancement is seen in any of the best fitness values.

(iii) Particle Position Changing

Every particle is known for its position, speed, and value of its fitness. In each iteration, each particle moves toward its best own position and the best-condensed particle ever. The motion of the particles depends on its current speed, and the pace is changed as follows:

$$\vec{v}_{ij}(t+1) = w \vec{v}_{ij}(t) + c_1 r_1 |\vec{p}_{ij}(t) - \vec{x}_{ij}(t)| + c_2 r_2 |\vec{p}_{gj}(t) - \vec{x}_{ij}(t)| \quad (8.28)$$

The index j is the number of dimensions of the particles, t is the sequence of repetitions, and c_1 and c_2 are the positive constant parameters called the accelerator coefficients. They are responsible for controlling the maximum number of steps, r_1 and r_2 , which are random numbers in the range of (0, 1). w is a constant, and $\vec{v}_{ij}(t+1)$ is the velocity of the particle i in the dimension of j in a repetition of $t+1$. \vec{x}_{ij} is the location of particle i in the dimension of j in a repetition of t ; $\vec{p}_{ij}(t)$ is the historical individual

best position of the swarm. With these interpretations, the new situation is obtained as follows:

$$\vec{x}_{ij}(t+1) = \vec{v}_{ij}(t+1) + \vec{x}_{ij}(t) \quad (8.29)$$

The parameter can be adjusted as follows to improve the performance of the PSO:

$$w = w_{\max} - \frac{w_{\max} - w_{\min}}{\text{iter}_{\max}} \times \text{iter} \quad (8.30)$$

$$c_1 = c_{1\max} - \frac{c_{1\max} - c_{1\min}}{\text{iter}_{\max}} \times \text{iter} \quad (8.31)$$

$$c_2 = c_{2\max} - \frac{c_{2\max} - c_{2\min}}{\text{iter}_{\max}} \times \text{iter} \quad (8.32)$$

Iter is the current iteration number, iter_{max} is the predefined maximum iteration number, w and c are constant accelerator coefficients, and their values and maximum and minimum values are usually defined with experiments.

(iv) Apply Portfolio Constraints

Generally speaking, there are two types of portfolio problems: with and without constraints (Benninga & Mayshar, 2000). In the constrained portfolio problems, short selling is allowed; it means that the stocks' weights in the portfolio can be negative. In the unconstrained one, short selling is not allowed, in which the portfolio weights can only be positive or zero. In both cases, the sum of the portfolio weights must be equal to one. The goal of creating a risky portfolio is to find the optimal composition of assets to maximize the Sharpe ratio.

- A Constrained Portfolio Optimization Model

$$\text{MaxSR} = \text{Max} \frac{\sum_{i=1}^N w_i r_i - R_f}{\sum_{i=1}^N \sum_{j=1}^N w_i w_j \sigma_{ij}} \quad (8.33)$$

$$\text{s.to } \sum_{i=1}^N w_i = 1 \quad (8.34)$$

$$0 \leq w_i \leq 1 \quad i = 1, \dots, N \quad (8.35)$$

- An Unconstrained Portfolio Optimization Model

$$\text{MaxSR} = \text{Max} \frac{\sum_{i=1}^N w_i r_i - R_f}{\sum_{i=1}^N \sum_{j=1}^N w_i w_j \sigma_{ij}} \quad (8.36)$$

$$\text{s.to} \quad \sum_{i=1}^N w_i = 1 \quad (8.37)$$

$$i = 1, \dots, N \quad (8.38)$$

If the risky portfolio is restricted, then the bestpos and N_bestpos are assessed by Eqs. (8.33)–(8.35). On the contrary, the bestpos and N_bestpos are assessed using Eqs. (8.36)–(8.38). While a particle comes to a new position in the search space, all the portfolio's constraints must be satisfied to ensure a valid movement within the search space.

8.4 Main Focus of the Chapter

This chapter examines the literature on constraint portfolio optimization models, as described in the previous section. A constraint portfolio selection model has been developed, which is described in the next section. Then the performance of the developed model has been evaluated using different risk measures. The proposed portfolios from the various risk measures have also been examined from the diversification viewpoint. The numerical results have been achieved using data of 20 stocks listed on the Tehran Stock Exchange.

8.4.1 The Proposed Model

In this chapter, Sarykalin, Serraino, and Uryasev's (2008) model has been selected as a base model for implementation since it is an appropriate model for this chapter's aim, which is a comparison of different risk measures' performance. They used the CVaR as a risk measure for the portfolio selection model. Therefore, the proposed model is as follows (Sarykalin et al., 2008):

$$\min \text{Risk} \quad (8.39)$$

$$\text{S.to } \mu_P = w^T \mu \geq \mu_{P_0} \quad (8.40)$$

$$\sum_{i=1}^N w_i = 1 \quad (8.41)$$

$$w_i \geq 0 \quad i = 1, 2, \dots, N \quad (8.42)$$

where μ_P represents the mean return of the portfolio, μ_{P_0} is the minimum ideal return, and μ is the return vector whose values are related to the return of each available stock in the portfolio. In this chapter's proposed model, the authors use conditional value at risk (CVaR) as a coherent risk measure shown in Eq. (8.43).

$$\text{CVaR} = \frac{1}{1-c} \int_{-1}^{\text{VaR}} xp(x)dx \quad (8.43)$$

where $P(x)$ is the probability function to obtain the return of portfolio x , and C is the point of intersection obtained from the VaR analysis, and VaR is the agreed-upon VaR level.

In the models Eqs. (8.39)–(8.42), constraint (Eq. 8.40) may not be feasible; that means for a certain level of risk, investors cannot get the ideal return. To solve this problem, here, the authors add a penalty term to the objective function that by increasing the level of risk, investors can get the ideal return, which is modified as follows:

$$\min \text{ Risk} + \text{Penalty} \quad (8.44)$$

$$\text{s.to } \mu_P = w^T \mu \geq \mu_{P_0} \quad (8.45)$$

$$\sum_{i=1}^N w_i = 1 \quad (8.46)$$

$$w_i \geq 0 \quad i = 1, 2, \dots, N \quad (8.47)$$

The violation part of this penalty function and constraint (Eq. 8.45) are added to the objective function with the coefficient $V(w)$:

$$V(w) = \begin{cases} 0 & \mu_P \geq \mu_{P_0} \\ 1 - \frac{\mu_P}{\mu_{P_0}} & \mu_P < \mu_{P_0} \end{cases} \quad (8.48)$$

Hence, the proposed model is provided in Eqs. (8.49)–(8.51).

$$\min \text{ Risk}(w)[1 + \beta V(w)] \quad (8.49)$$

$$\text{s.to } \sum_{i=1}^N w_i = 1 \quad (8.50)$$

$$w_i = 0 \quad i = 1, 2, \dots, N \quad (8.51)$$

where β is a constant which can be selected based on investor preference. It shows the percentage of return that an investor passes up when the portfolio's mean return is less than the minimum ideal return. This coefficient has been set on 1000 in this study.

8.5 A Case Study in the Tehran Stock Exchange

To show the applicability of the developed model, Eqs. (8.49)–(8.51), which can be solved by the PSO technique, 20 stocks have been selected from the Tehran Stock Exchange (TSE). Fig. 8.1 shows the main steps of the PSO algorithm which is applied here along with its parameters as shown in Table 8.1. The selected stocks are among active stocks in the market from a historical trading point of view. They have been selected from different activity sectors to be a good representative of the TSE. The PSO algorithm parameters have been chosen from Ratnaweera, Halgamuge, and Watson (2004). The data range is from 2019/09/23 to 2020/04/05. We use MATLAB software for coding and solving the developed model.

Here, the CVaR risk measure with 0.95 confidence level, variance risk measure, and semivariance risk measure have been used to implement the proposed model. The portfolio weight of each stock and the return of them are calculated, as shown in Table 8.2. Table 8.2 indicates that, by utilizing the CVaR, the highest return is obtained due to a tolerable level of risk. Also, the calculation of the coefficient of variation (CV) emphasizes the superiority of the CVaR. Coefficient of variation (CV) is the ratio of portfolio risk by the total portfolio return. It is an excellent ratio to compare the suggested portfolios considering both the risk and the portfolios' return. Moreover, the diversification index (DI) is applied to evaluate the diversity of the proposed portfolios. The less value of this ratio shows a more diversified portfolio. Considering the Herfindahl index (HI), which is a measure of economic concentration, DI is calculated as follows (Yiğit & Tür, 2012):

$$DI = 1 - HI = \sum_{i=1}^N W_i^2 \quad (8.52)$$

where W_i is the asset weight and N is the number of total assets. The DI ratio of zero means ultimate diversification, and the DI ratio of one means no diversification. Although variance performs slightly better than CVaR and semivariance measures, this performance does not show its absolute privilege because the differences are trivial. To make more comparisons, Table 8.2 also indicates metrics of a simple strategy, “equal weights portfolio,” in which all stocks in the portfolio have the same weights. Regardless of the DI, which has shown this strategy’s privilege, the other index in Table 8.2 shows that using a portfolio optimization model helps investors act much better. Figure 8.2 shows a diagram of the penalty (cost) function used in the model for each iteration of the PSO algorithm regarding different risk measures. In this study, the second part of the developed model’s objective function in Eq. (8.49) is considered the penalty (cost) function. This function as $Risk(w)[\beta V(w)]$ shows the percentage of return that an investor passes up when the mean return of the portfolio is less than the minimum ideal return. Not only it shows a convergence, but also it shows a trivial amount of penalty. The amount of penalty (cost) in every three risk measures converges to 4.1%. It means that investors lose only a small portion of 0.041 out of their profit than their ideal return to obtain a better risk level.

As shown in Fig. 8.2, the model with variance risk measure converges faster than the other two models with CVaR and semivariance risk measures. It can also be seen

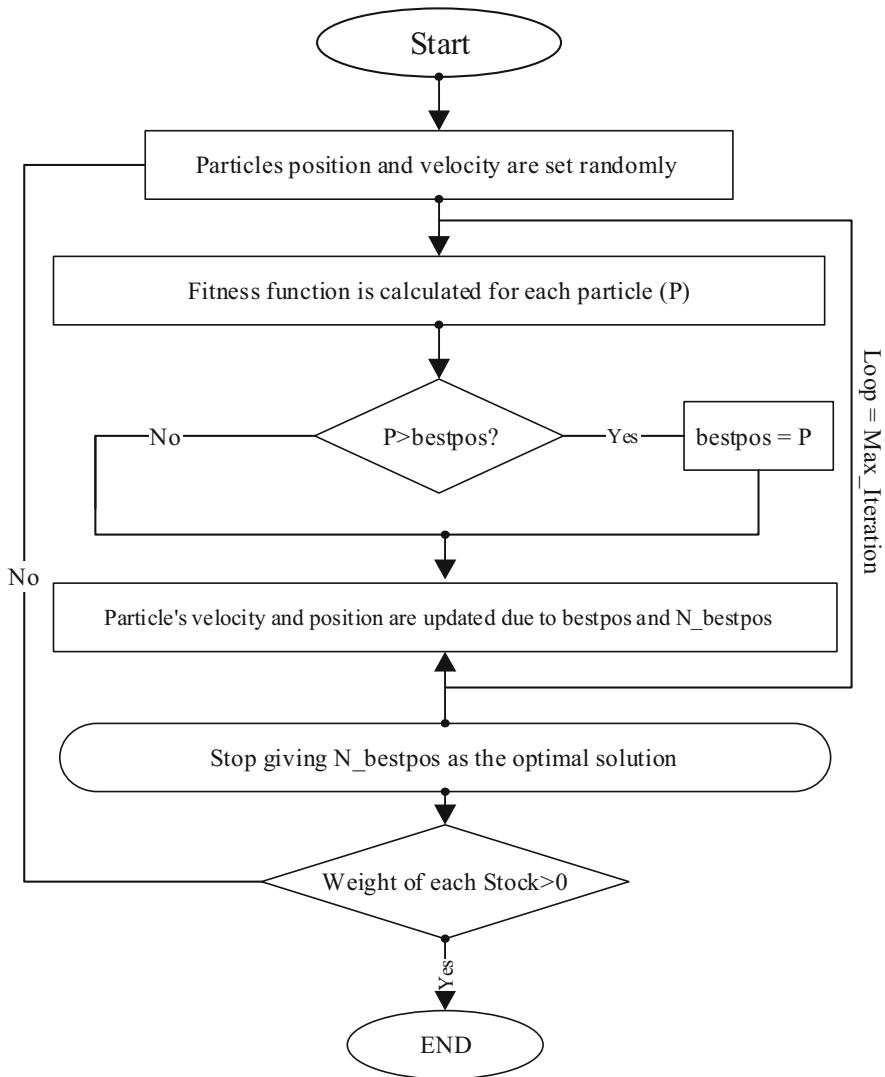


Fig. 8.1 The flowchart depicting our PSO algorithm's steps. Source: Author's own creation

that for the 3 risk measures, after 50 iterations, the value of the cost is almost the same; hence, 100 iterations are good enough to get the optimal solution. As shown in Table 8.2, solving time in variance risk measure is shorter. One reason is its simple calculations, and the other is its fast convergence. Figure 8.3 shows efficient frontiers for each risk measure. From Fig. 8.3, a conclusion can be drawn that the CVaR measure brings about a better return at a certain risk level. Semivariance and variance have approximately the same result, but the semivariance shows a little bit better results. The best value for return due to the level of risk is shown in Table 8.2.

Table 8.1 Parameters of the PSO algorithm

Parameters	Value
Maximum number of iterations	100
Population size	40
Inertia weight	0.7298
Inertia weight damping ratio	1
Personal acceleration coefficient (c_1)	1.4962
Global acceleration coefficient (c_2)	2
Velocity limits	Max = 1 Min = 0

8.6 Conclusion and Future Research Directions

In this chapter, the authors have reviewed the various models of stock portfolio optimization and the PSO technique applications for solving these models. The authors also proposed a constraint portfolio selection problem, which is solved by the PSO technique. The proposed model is implemented by using three risk measures of the CVaR, semivariance, and variance to see its performance on portfolio risk, return, and diversification. This study shows that the CVaR risk measure produces better performance on risk and return. The variance risk measure has better performance on portfolio diversification.

Constructing a profitable portfolio of assets is a financial expert's indisputable problem. As shown in the presented case study's numerical results in this chapter, the CVaR performs better than the other two risk measures of variance and semivariance from the risk-adjusted profit viewpoint. Based on that, one can conclude that if investors want to invest in the Tehran Stock Exchange, it seems that it will be preferable to utilize CVaR as a measure to construct their portfolio. This chapter's strength is to consider the effects of different risk measures in a constrained portfolio selection model, which is focused on the implementation of the PSO technique as a metaheuristic solution method.

For further studies, researchers can consider the following topics:

- Considering more constraints to the portfolio selection model to be more realistic.
- Using various risk measures to compare their performance.
- Using the multi-objective PSO method to solve this problem, which is a multi-objective problem in nature.
- Implementing the proposed model in other stock markets to compare the results.
- PSO algorithm can be used in reinforcement, machine, and deep learning methods, which are now very popular in portfolio management.

Key Terms and Definitions

CAPM: This is the abbreviation of the capital asset pricing model, which defines the relationship between the systematic risk and expected return of a financial asset. With this model, an asset's expected return can be estimated using its systematic risk and expected return of the market portfolio.

Cardinality Constraint: This constraint restricts the number of stocks in an investment portfolio.

Table 8.2 Portfolio weights which are obtained by the proposed portfolio optimization model using different risk measures

Stock symbol	Equal weights portfolio	PSO-semivariance	PSO-variance	PSO-CVaR
AKABER	0.05	0.043896	0.056136	0.107558
BTRANS	0.05	0.066087	0.093738	0.086625
PARS	0.05	0	0.000774	0.000384
TAPIKO	0.05	0.19022	0.175294	0.145712
JAM	0.05	0.039487	0.026256	0.001244
KHESAPA	0.05	0.000307	0	0
KHODRO	0.05	7.63E-06	0.00E+00	0.003276
SHABRIZ	0.05	0.008544	0.023186	0
SHETRAN	0.05	0.007827	0.005634	0.001494
FARS	0.05	0.037053	0.039749	0
FAMELI	0.05	0.110437	0.121528	0.076385
FOLAD	0.05	0.015957	0.017705	0.105798
KHECHAD	0.05	0	1.39E-05	0
KEGOL	0.05	0	8.31E-05	0.003061
MOBIN	0.05	0.002487	0.014051	0.015612
NOORI	0.05	0.013009	0.027995	0.060683
HIWEB	0.05	0.029024	0.037373	0
HAMRAH	0.05	0.09055	0.08538	0.102677
VABEMELLAT	0.05	0.150843	0.139772	0.15202
VATEJARAT	0.05	0.194265	0.135332	0.13747
Total portfolio return	0.639	0.716	0.697	1.16
Portfolio risk	0.00294	0.00073	0.00064	0.00021
Coefficient of variation	0.0046009	0.0010196	0.0009182	0.000181
Diversification index	0.05	0.127705	0.108106	0.113833
Solving time (sec)	–	134	119	186

Coefficient of Variation (CV): This is the portfolio risk ratio by the total portfolio return. This is a useful measure for comparing risky assets considering both the risk and the return of those assets.

Efficient Frontier (EF): Each point in this frontier shows a portfolio with the highest return due to a specific risk level. This is a good curve to show the relationship between efficient portfolios.

Particle Swarm Optimization (PSO): Particle swarm optimization is a population-based stochastic optimization technique used as an evolutionary computation technique to solve many types of multi-objective optimization problems such as constrained portfolio optimization models.

Risk Measures: These are statistical measures used to measure investment risk and volatility of portfolios using historical prices of assets of those portfolios.

Sharpe Ratio: As a risk measure, this ratio is the average return gained over the risk-free rate per unit of volatility or a risky asset's absolute risk.

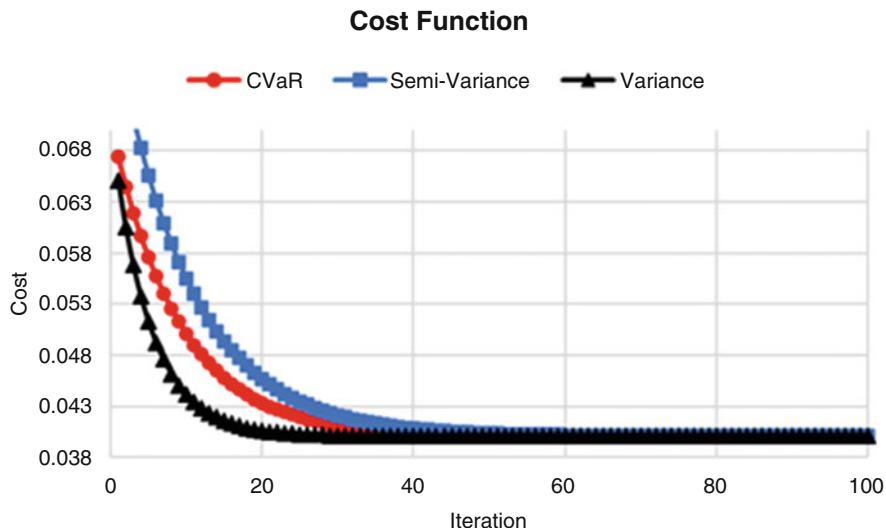


Fig. 8.2 Cost (penalty) function in each iteration of the PSO algorithm. Source: Author's own creation

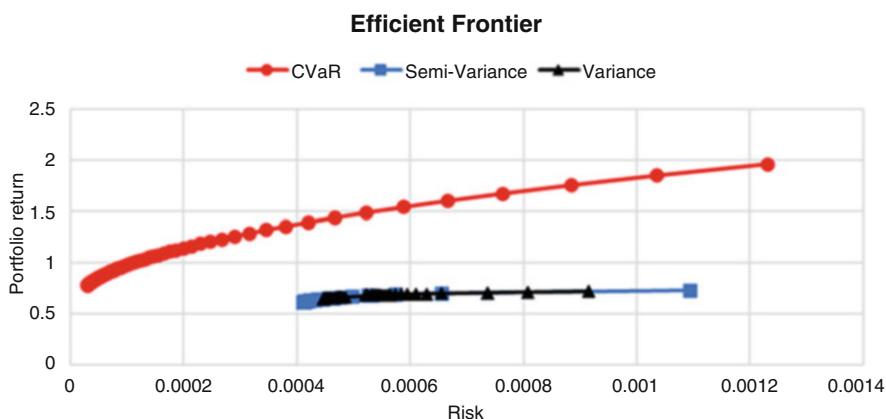


Fig. 8.3 The efficient frontier for the three risk measures. Source: Author's own creation

Short Sales: It is the sale of a stock that the seller does not own. In this kind of selling, the investor sells borrowed assets when she/he expects that the price of those assets will be decreased. The sellers ought to return the same number of assets at a specific time in the future.

Appendix

Table 8.3 gives a comprehensive view of relevant studies of portfolio optimization and the applied techniques as solution methods.

Table 8.3 Summary of some relevant studies in the literature

Research paper	Risk measure				Metaheuristic				Multi-period
	var	VaR	CVaR	Other	GA	SA	PSO	OTHER	
Markowitz (1959)	*								*
Mao (1970)	*								*
Konno and Yamazaki (1991)	*								*
Ouederni and Sullivan (1991)	*								*
Eberhart and Kennedy (1995)					*		*		
Kennedy (1999)							*		
Chang et al. (2000)	*				*	*			*
Mendes (2004)	*						*		
Yuan, Wang, Zhang, and Yuan, Wang, Zhang, and Yuan (2004)							*		
Zhao, Li, and Qian (2005)							*		
Chen et al. (2006)	*						*		
Celikyurt and Özekici (2007)	*							*	*
Clerc (2007)							*		
Sarykalin et al. (2008)		*	*						*
Cura (2009)	*				*	*	*	*	
Assareh, Behrang, Assari, and Ghanbarzadeh (2010)					*		*		
Esfahanipour and Mousavi (2011)			*		*				
Zhu et al. (2011)	*				*		*		
Kuo, Wang, and Huang (2011)							*		
Bank, Ghomi, Jolai, and Behnamian (2012)						*	*		
Reid and Malan (2015)	*						*	*	
Yin et al. (2015)	*						*		
Ni, Yin, Tian, and Zhai (2017)	*						*		*
Zaheer et al. (2018)	*						*	*	
Salehpoor and Molla-Alizadeh-Zavardehi (2019)	*				*	*	*	*	
Almahdi and Yang (2019)	*						*	*	
Babazadeh and Esfahanipour (2019)		*			*				

References

- Almahdi, S., & Yang, S. Y. (2019). A constrained portfolio trading system using particle swarm algorithm and recurrent reinforcement learning. *Expert Systems with Applications*, 130, 145–156.
- Assareh, E., Behrang, M., Assari, M., & Ghanbarzadeh, A. (2010). Application of PSO (particle swarm optimization) and GA (genetic algorithm) techniques on demand estimation of oil in Iran. *Energy*, 35(12), 5223–5229.
- Babazadeh, H., & Esfahanipour, A. (2019). A novel multi-period mean-VaR portfolio optimization model considering practical constraints and transaction costs. *Journal of Computational and Applied Mathematics*, 361, 313–342.
- Bank, M., Ghomi, S. F., Jolai, F., & Behnamian, J. (2012). Application of particle swarm optimization and simulated annealing algorithms in flow shop scheduling problem under linear deterioration. *Advances in Engineering Software*, 47(1), 1–6.
- Benninga, S., & Mayshar, J. (2000). Heterogeneity and option pricing. *Review of Derivatives Research*, 4(1), 7–27.
- Celikyurt, U., & Özekici, S. (2007). Multiperiod portfolio optimization models in stochastic markets using the mean-variance approach. *European Journal of Operational Research*, 179 (1), 186–202.
- Chang, T.-J., Meade, N., Beasley, J. E., & Sharaiha, Y. M. (2000). Heuristics for cardinality constrained portfolio optimization. *Computers and Operations Research*, 27(13), 1271–1302.
- Chen, W., Zhang, R.-T., Cai, Y.-M., & Xu, F.-S. (2006). Particle swarm optimization for constrained portfolio selection problems. *Paper presented at the 2006 International conference on machine learning and cybernetics*.
- Clerc, M. (2007). *Back to random topology*. Mar: Relatório Técnico.
- Cura, T. (2009). Particle swarm optimization approach to portfolio optimization. *Nonlinear Analysis: Real World Applications*, 10(4), 2396–2406.
- Eberhart, R., & Kennedy, J. (1995). Particle swarm optimization. *Paper presented at the Proceedings of the IEEE international conference on neural networks*.
- Esfahanipour, A., & Mousavi, S. (2011). A genetic programming model to generate risk-adjusted technical trading rules in stock markets. *Expert Systems with Applications*, 38(7), 8438–8445.
- Kennedy, J. (1999). Small worlds and mega-minds: Effects of neighborhood topology on particle swarm performance. *Paper presented at the proceedings of the 1999 congress on evolutionary computation-CEC99 (Cat. No. 99TH8406)*.
- Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. *Paper presented at the proceedings of ICNN'95-international conference on neural networks*.
- Konno, H., & Yamazaki, H. (1991). Mean-absolute deviation portfolio optimization model and its applications to the Tokyo stock market. *Management Science*, 37(5), 519–531.
- Kuo, R., Wang, M., & Huang, T. (2011). An application of particle swarm optimization algorithm to clustering analysis. *Soft Computing*, 15(3), 533–542.
- Mao, J. C. (1970). Models of capital budgeting, EV vs. ES. *Journal of Quantitative Financial Analysis*, 4, 657–675.
- Markowitz, H. (1959). Portfolio selection. *Investment under Uncertainty*.
- Mendes, R. (2004). Population topologies and their influence in particle swarm performance. PhD Final Dissertation, Departamento de Informática Escola de Engenharia Universidade do Minho.
- Momen, O., Esfahanipour, A., & Seifi, A. (2019). Portfolio selection with robust estimators considering behavioral biases in a causal network. *RAIRO-Operations Research*, 53(2), 577–591.
- Ni, Q., Yin, X., Tian, K., & Zhai, Y. (2017). Particle swarm optimization with dynamic random population topology strategies for a generalized portfolio selection problem. *Natural Computing*, 16(1), 31–44.
- Ouederni, B. N., & Sullivan, W. G. (1991). A semi-variance model for incorporating risk into capital investment analysis. *The Engineering Economist*, 36(2), 83–106.

- Ratnaweera, A., Halgamuge, S. K., & Watson, H. C. (2004). Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients. *IEEE Transactions on Evolutionary Computation*, 8(3), 240–255.
- Reid, S. G., & Malan, K. M. (2015). Constraint handling methods for portfolio optimization using particle swarm optimization. *Paper presented at the 2015 IEEE symposium series on computational intelligence*.
- Salehpour, I. B., & Molla-Alizadeh-Zavardehi, S. (2019). A constrained portfolio selection model at considering a risk-adjusted measure by using hybrid meta-heuristic algorithms. *Applied Soft Computing*, 75, 233–253.
- Sarykalin, S., Serraino, G., & Uryasev, S. (2008). Value-at-risk vs. conditional value-at-risk in risk management and optimization. In *State-of-the-art decision-making tools in the information-intensive age* (pp. 270–294). Catonsville, MD: Informs.
- Yiğit, İ., & Tür, Ş. (2012). Relationship between diversification strategy applications and organizational performance according to Herfindahl Index criteria. *Procedia - Social and Behavioral Sciences*, 58, 118–127.
- Yin, X., Ni, Q., & Zhai, Y. (2015). A novel particle swarm optimization for portfolio optimization based on random population topology strategies. *Paper presented at the International Conference in Swarm Intelligence*.
- Yuan, X.-H., Wang, C., Zhang, Y.-C., & Yuan, Y.-B. (2004). A survey on the application of particle swarm optimization to electric power systems. *Power System Technology*, 19, 003.
- Zaheer, K. B., Abd Aziz, M. I. B., Kashif, A. N., & Raza, S. M. M. (2018). Two-stage portfolio selection and optimization model with the hybrid particle swarm optimization. *MATEMATIKA: Malaysian Journal of Industrial Applied Mathematics*, 34(1), 125–141.
- Zhao, J., Li, T., & Qian, J. (2005). Application of particle swarm optimization algorithm on robust PID controller tuning. *Paper presented at the International Conference on Natural Computation*.
- Zhu, H., Wang, Y., Wang, K., & Chen, Y. (2011). Particle Swarm Optimization (PSO) for the constrained portfolio optimization problem. *Expert Systems with Applications*, 38(8), 10161–10169.

Chapter 9

Optimal Portfolio Selection with Particle Swarm Algorithm: An Application on BIST-30



Burcu Adıgüzel Mercangöz and Altaf Q. H. Badar

Abstract Optimization is to find the best-performing solution under the constraints given. It can be something better by optimization process. Heuristic algorithm is an optimization algorithm which depends on natural events. The algorithms are simple and easy to implement for the researcher. The portfolio optimization is a process to find a solution to select the most appropriate combination between all financial assets under certain expectations and constraints. While solving portfolio optimization problems, the aim is to create portfolios by selecting the assets that provide the highest return from huge numbers of financial assets at a certain risk level or provide the lowest risk at a certain level of return. This chapter aims to examine the optimum portfolio with minimum risk by using the particle swarm optimization (PSO) technique, for the stocks in the BIST-30 index. Logarithmic returns are calculated using the price data of the stocks. By using these returns, the optimum portfolio with minimum risk is created with PSO and nonlinear GRG (generalized reduced gradient) techniques. The empirical results obtained indicate that both methods give similar results.

Keywords Optimization · Particle swarm optimization (PSO) · Portfolio optimization · Markowitz portfolio theory · Heuristics · Swarm intelligence

B. A. Mercangöz
Istanbul University, Istanbul, Turkey
e-mail: burcua@istanbul.edu.tr

A. Q. H. Badar (✉)
National Institute of Technology, Warangal, India
e-mail: altafbadar@nitw.ac.in

9.1 Introduction

Particle swarm optimization (PSO) is one of the intuitive techniques. This technique was first introduced by researchers James Kennedy and Russell Eberhart in the 1990s to find optimum solutions to nonlinear numerical problems inspired by the collective movements of fish and bird flocks (Eberhart & Kennedy, 1995). The technique has evolved since then in several ways. The technique itself was improved through several publications by the duo of Shi and Eberhart (2008). As the robustness of the technique was proved, it was implemented and researched in many publications. PSO not only sees variations but also has a lot of hybrid versions introduced, along with different optimization techniques.

PSO works on a swarm of particles. A single particle represents a probable solution to the optimization problem. The particles move in the search space based on their respective velocities. The velocity of a particle is dependent on its own inertia, for instance, its own previous velocity, individual best, and global best. Individual best is the best position that a particular particle has achieved until the current iteration, while the global best is the best position occupied by any of the particles from within the swarm. Each particle in PSO can be mapped to a fitness function. As the particle moves in the search space, its fitness function also changes. PSO tries to obtain the optimal position in the search space through the movement of particles. Some of the factors that affect the movement of particles in the swarm are constriction factor, random factors, inertia constant, etc. These factors are responsible for the explorative and exploitative behavior of the swarm.

The portfolio optimization problem is related to how investors will allocate their wealth among various assets. Therefore, portfolio optimization problems have been an important research area in modern finance and risk management. In this study, the PSO technique issued for optimum solutions of the Markowitz mean-variance model in the portfolio selection problem. Optimal portfolios are created according to the PSO and nonlinear generalized reduced gradient (GRG) techniques by using the logarithmic return data between June 2016 and July 2018 for 25 stocks within the BIST-30 index. Then, the coefficients of variation are calculated, with the risks and returns that are obtained. The coefficients of variation of the two techniques are similar. It has similar results, and PSO can thus be used as an alternative for solving portfolio optimization problem. Since quite similar results are obtained from two different techniques, it also proves the reliability of the techniques.

9.2 Portfolio Optimization and Mathematical Model

Investors are willing to get the highest return at a given level of risk or are willing to take the lowest risk at a given level of return. This is the portfolio optimization problem, which arises from the desire to maximize return while minimizing the

investor's risk. In the stated balance, the best solution tried to be reached. The Markowitz mean-variance model is described below.

The expected return of the risky portfolio $E(R_p)$ is estimated as Eq. (9.1).

$$E(R_p) = W_A * R_A + W_B * R_B \quad (9.1)$$

R_A and R_B are the returns of two risky assets, R_p is the portfolio return, and W_A and W_B are the weights of A and B in the risky portfolio, respectively, with two risky assets.

The variance of the two assets' risky portfolio is calculated as shown in Eq. (9.2).

$$\sigma_p^2 = W_A^2 \sigma_A^2 + W_B^2 \sigma_B^2 + 2W_A W_B \text{Cov}(R_A, R_B) \quad (9.2)$$

The standard deviations of the two risky assets are σ_A and σ_B . The covariance between assets A and B is $\text{Cov}(R_A, R_B)$.

This is just an example for two-asset risky portfolio. This can be extended to risky portfolios with more than two assets. Eq. (9.3) shows the estimation of expected return $E(R_p)$ of a risky portfolio of multiple assets. The estimation of standard deviation (σ_p) of a multiple-asset risky portfolio uses covariance matrix of all assets in the portfolio. Portfolio returns of multiple assets depend on the risky assets' own returns and the weights that describe how the portfolio investment is split. Therefore, expected return for "n" assets is calculated as below:

$$E(R_p) = \sum_{i=1}^n E(R_i) \quad (9.3)$$

where n is the number of securities. The return is being finding by compute the weighted average returns of each security included in the portfolio. Portfolio risk also can be calculated using the weights and covariances of each asset in the risky portfolio as given in Eq. (9.4):

$$\text{Var}(R_p) = \sigma_p^2 = \sum_{i=1}^n \sum_{j=1}^n W_i W_j \text{Cov}(R_i R_j) \quad (9.4)$$

The mathematical indication of portfolio optimization problem by using the Markowitz mean-variance model is shown in the nonlinear programming model as below. Equation (9.5) describes the objective function, while Eq. (9.6) represents the constraint associated with the objective function.

9.2.1 Objective Function

$$\text{Min.Var}(R_p) = \sigma_p^2 = \sum_{i=1}^n \sum_{j=1}^n W_i W_j \text{Cov}(R_i R_j) \quad (9.5)$$

9.2.2 Constraints

$$\sum_i^n W_i = 1 \quad (9.6)$$

$$0 \leq W_i \leq 1 \quad i = 1, 2, \dots, n.$$

The constraints mean that the assets in the portfolio cannot be in short positions.

9.3 A Portfolio Optimization Application by Using PSO Algorithm

A basic application of the PSO technique is applied for the Markowitz mean-variance model with stocks of BIST-30 index for the period from June 2016 and July 2018. The data is obtained from www.investing.com. The daily data is used and analyzed for 25 common stocks in the index since the data availability and being able to equalize periods for all stocks.

Coding and running of PSO is done by MATLAB. The information of these 25 stocks in the BIST-30 index traded on the Istanbul Stock Exchange is as in Table 9.1.

Daily data between June 2016 and July 2018 is used for these 25 stocks in BIST-30. Logarithmic returns are calculated with 503 daily observations using Eq. (9.7).

$$R_t = \ln(P_t/P_{t-1}) \quad (9.7)$$

Table 9.2 shows the log returns, variance, and standard deviation of the stocks. The variance-covariance matrix is as shown below in Table 9.3.

Table 9.1 25 Stocks in BIST-30: Code and company names used in the application

	Code	Company name (*Original Turkish names from public disclosure platform)
1	AKBNK	AKBANK T.A.Ş. Unvanı
2	ARCLK	ARÇELİK A.Ş.
3	ASELS	ASELSAN ELEKTRONİK SANAYİ VE TİCARET A.Ş.
4	BIMAS	BİM BİRLEŞİK MAĞAZALAR A.Ş.
5	DOHOL	DOĞAN ŞİRKETLER GRUBU HOLDİNG A.Ş.
6	KOZAA	KOZA ANADOLU METAL MADENCİLİK İŞLETMELERİ A.Ş.
7	HALKB	TÜRKİYE HALK BANKASI A.Ş.
8	GARAN	TÜRKİYE GARANTİ BANKASI A.Ş.
9	ISCTR	TÜRKİYE İŞ BANKASI A.Ş.
10	SISE	TÜRKİYE ŞİŞE VE CAM FABRİKALARI A.Ş.
11	SAHOL	SABANCI HOLDİNG
12	KRDMD	KARDEMİR KARABÜK DEMİR ÇELİK SANAYİ VE TİCARET A.Ş.
13	TKFEN	TEKFEN HOLDİNG A.Ş.
14	TAVHL	TAV HAVALİMANLARI HOLDİNG A.Ş.
15	PETKM	PETKİM PETROKİMYA HOLDİNG A.Ş.
16	TOASO	TOFAŞ TÜRK OTOMOBİL FABRİKASI A.Ş.
17	SODA	SODA SANAYİİ A.Ş.
18	THYAO	TÜRK HAVA YOLLARI A.O.
19	TCELL	TURKCELL İLETİŞİM HİZMETLERİ A.Ş.
20	TUPRS	TÜPRAŞ-TÜRKİYE PETROL RAFİNERİLERİ A.Ş.
21	VAKBN	TÜRKİYE VAKIFLAR BANKASI T.A.O.
22	YKBNK	YAPI VE KREDİ BANKASI A.Ş.
23	EREGL	EREĞLİ DEMİR VE ÇELİK FABRİKALARI T.A.Ş.
24	TTKOM	TÜRK TELEKOMÜNİKASYON A.Ş.
25	KCHOL	KOÇ HOLDİNG A.Ş.

Source: KAP Public Disclosure Platform

9.4 Portfolio Optimization by Using PSO Algorithm

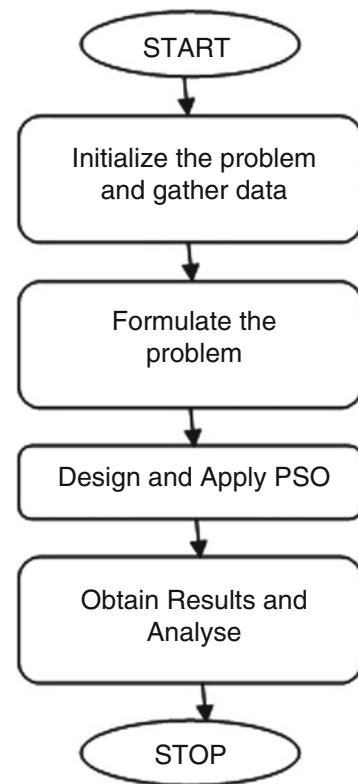
The problem of portfolio optimization is solved through the implementation of PSO. The implementation of PSO is realized as indicated in the flowchart of Fig. 9.1. Figure 9.1 shows how the problem is solved stepwise. In the first step, the problem is defined, and data required for solving the problem are gathered. Data related to various stocks is also to be collected in the initial step. Next the problem is formulated according to the Markowitz mean-variance model along with the data available. The data available initially is raw and requires to be processed to form different matrices as shown in the previous section. In the problem, the Markowitz mean-variance model is the fitness function that is to be optimized. After formulating the problem, the coding is done. The most common version of PSO is implemented (Clerc, 1999). The Markowitz mean-variance model and PSO are coded using MATLAB. The coding of PSO should also be accompanied by the debugging of

Table 9.2 Return, variance, and standard deviation of the 25 stocks

	AKBANK	ARCLK	ASELS	BIMAS	DOHOL	KOZAA	HALKB	GARAN	ISCTR
Average log return	-0.00013	-0.00053	0.002065	0.000349	0.001486	0.00373	-0.0004	0.000264	0.000314
Standard deviation	0.017606	0.015514	0.024001	0.014477	0.031727	0.043822	0.023784	0.018751	0.0179
Variance	0.00031	0.000241	0.000576	0.00021	0.001007	0.00192	0.000566	0.000352	0.00032
SISE	SAHOL	KRDMD	TKFEN	TAVHL	PETKM	TOASO	SODA	THYAO	
Average log return	0.001005	-8.8E-05	0.002446	0.002098	0.001284	0.000695	1.18E-05	0.001634	0.001751
Standard deviation	0.015689	0.014818	0.026747	0.260214	0.021248	0.018614	0.01592	0.015351	0.024391
Variance	0.000246	0.00022	0.000715	0.067712	0.000451	0.000346	0.000253	0.000236	0.000595
TCELL	TUPRS	VAKBN	YKBNK	EREGL	TTKOM	KCHOL			
Average log return	0.000315	0.001431	0.000184	-6.7E-05	0.001878	-0.00104	0.000118		
Standard deviation	0.015905	0.016677	0.019914	0.019309	0.020898	0.020677	0.016515		
Variance	0.000253	0.000278	0.000397	0.000373	0.000437	0.000428	0.000273		

Table 9.3 Variance-covariance matrix of the 25 stocks

Fig. 9.1 Implementation step flowchart of basic PSO.
Source: Authors' own creation



the program to get perfect results. PSO is then executed to obtain results for the defined problem.

9.4.1 Problem Definition

PSO is used to solve the optimization problem of portfolio optimization. The main component of the problem is the fitness function that is the Markowitz mean-variance model in this study. A separate function is defined for obtaining the fitness function of a random particle. MATLAB is used to implement PSO as well as the fitness function.

In the program, PSO uses the fitness function again and again to evaluate various particles. In the initial part of PSO, the particles are randomly generated within the search space. These particles represent a candidate solution, i.e., any particle can represent a full solution to the problem. The solution to the problem is a weight matrix of size 1×25 , as the number of stocks is 25. In case the number of stocks is more or less, the matrix size of the weights will also change accordingly. One row

Table 9.4 A particle in PSO

0.040936	0.056778	0.055031	0.060798	0.022407
0.053885	0.043314	0.030853	0.004792	0.060705
0.026267	0.047298	0.057676	0.008155	0.009951
0.042759	0.037755	0.069287	0.062166	0.057138
0.003994	0.005671	0.006888	0.062119	0.073375

matrix of size 1×25 represents one particle. The members of the matrix should be in the range of 0–1 and should follow the constraint mentioned in Eq. (9.6).

Table 9.4 gives an example of a particle generated randomly at the start of the program. Due to the space constraint, the values are arranged in five rows, but in the code, it should be a single row matrix.

If all the values are added of Table 9.4, it is observed that the constraint of the problem is satisfied and the total comes out to be unity.

In the program, 30 particles are taken which are generated at the beginning of the program. This collection of 30 particles is called the **population**. Next it is needed to evaluate the fitness function of each variable.

The fitness function is separately defined as mentioned earlier. The algorithm for the same function is given below:

Algorithm for Fitness Function Evaluation

```

Get Required Data
Get the Particle to be evaluated
SumFF = 0;
for 1 to No of Stocks do
  for 1 to No of Stocks do
    SumFF = SumFF + Wi * Wj * Cov (Ri, Rj)
  end for
end for
Return SumFF

```

In the above algorithm, W_i and W_j are the weights at positions “i” and “j,” respectively, in the particle. Also SumFF represents the **fitness function** value.

Once fitness function values are obtained for all the particles, it is proceeded to implement PSO. In PSO, all the particles move with a certain velocity to converge at the end to the global optimal solution. The movement of the particles is termed as its **velocity**. The velocity of a particle is dependent on inertia, the influence of its own best position, and the global best position.

In each iteration of PSO,

- (i) Velocity for all the particles is calculated.
- (ii) The individual positions of the particles are updated.
- (iii) Fitness function is found for all the particles.
- (iv) The individual and global best positions are updated.

The same is represented in the algorithm given below:

Algorithm for Particle Swarm Optimization

```

Input Required Data
Generate Random Particles
Evaluate the Fitness Function of these Particles
for 1 to Max Iterdo
    for 1 to Population Size do
        Calculate velocity
        Update Particle Position
        Obtain Fitness Function
        Update Individual Best
    endfor
    Update Global Best
endfor

```

9.4.2 Parameters of PSO

Some of the parameters that control PSO are described below:

Maximum Number of Iterations: PSO is an iteration-based optimization model. The PSO technique can be stopped through three different conditions, viz., (a) based on the number of iterations, (b) convergence, and (c) if the global best does not improve for a certain number of iterations. It is gone by the first condition and stops the PSO method at 200 iterations. A high number of iterations would lead to an extended solution time.

Population Size: Population size gives the number of particles used to search the optimal global best position. Usually 30–50 particles are utilized in a PSO method. In the case, it is created 30 particles to search the global best in the available search space.

Dimensions of the Search Space: The number of input variables defines the dimensions of these arch spaces. In portfolio optimization, the number of stocks shall define the number of dimensions of these arch spaces. Currently, as 25 stocks have been considered, the dimension of these arch spaces will be 25. In case the number of stocks changes, the dimensions of these arch spaces will also change.

Inertia Coefficient: Inertia coefficient is a very important parameter in these arch processes of PSO. A higher value of inertia coefficient represents more exploration and less exploitation, whereas a lesser value of inertia coefficient represents more exploitation and lesser exploration. Both exploration and exploitation should be properly matched for the optimal operation of PSO. Inertia coefficient is modified to constriction factor in Clerc (1999). The value of the constriction factor is fixed to 0.729 in the present case.

The other parameters of PSO are c_1 , individual acceleration coefficient, and c_2 , social or global acceleration coefficients which are taken as 2.05.

PSO is executed for 200 iterations with 30 particles for the case presented above. The variation in the global best position is presented in Fig. 9.2.

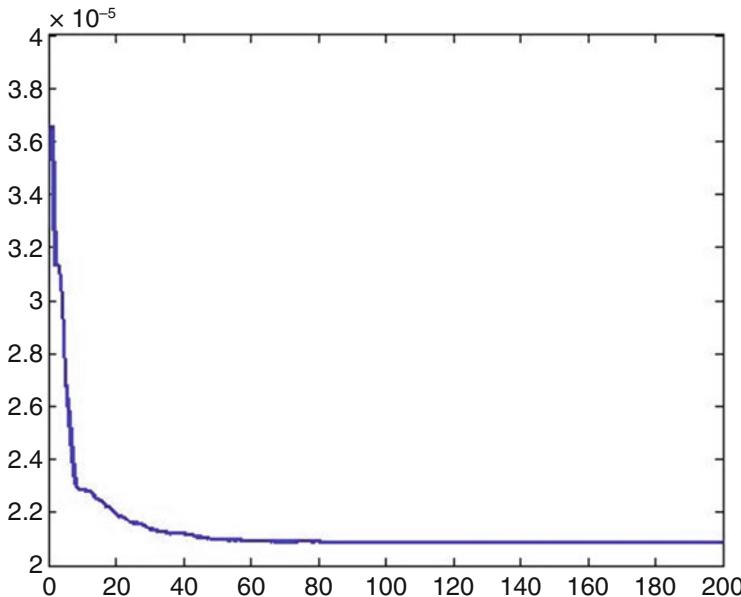


Fig. 9.2 Change in objective function value concerning the number of iterations. Source: Authors' own creation

A comparison of the optimal solution found by PSO and nonlinear GRG methods is presented in Table 9.5.

Portfolio return is estimated based on the weights which minimize the risk level. According to the PSO and nonlinear GRG techniques, the optimal portfolio return, variance, and standard deviation are calculated and shown in Table 9.6.

In Table 9.7, the results obtained by using PSO and nonlinear GRG techniques give close values. However, the coefficients of variation are calculated here to see which method gives better results.

It has similar results and PSO can thus be used as an alternative. Since quite similar results are obtained from two different techniques, it also proves the reliability of the techniques.

9.5 Conclusion

Optimization is the process of getting the best solution while conducting specific operations for a specific purpose. Portfolio selection problem depends on investors' expectations and model's constraints. According to the investors' certain expectations and model's certain constraints, the decision is made to create an optimum portfolio from a variety of assets. Regarding the correlation coefficients of the asset returns, while adding new assets to a risky portfolio, the total risk decreases.

Table 9.5 Weights of the 25 stocks in the optimal portfolio by using PSO and nonlinear GRG techniques

	Weights PSO	Weights Nonlinear GRG
AKBNK	0.00000000	0.00000000
ARCLK	0.07329542	0.07304953
ASELS	0.02131277	0.02123730
BIMAS	0.08607284	0.08620884
DOHOL	0.01410830	0.01435069
KOZAA	0.01253312	0.01272358
HALKB	0.04394518	0.04404135
GARAN	0.00000000	0.00000000
ISCTR	0.04603436	0.04640693
SISE	0.10035837	0.10070752
SAHOL	0.07655238	0.07521591
KRDMD	0.02236350	0.02222367
TKFEN	0.00052400	0.00052697
TAVHL	0.02135613	0.02159467
PETKM	0.04458688	0.04490066
TOASO	0.06552657	0.06545679
SODA	0.10943198	0.10966535
THYAO	0.02348696	0.02350349
TCELL	0.03149002	0.03098662
TUPRS	0.05588755	0.05625737
VAKBN	0.00000000	0.00000000
YKBNK	0.00000000	0.00001677
EREGL	0.03970906	0.03976780
TTKOM	0.05026441	0.05014628
KCHOL	0.06116019	0.06101188
Sum	1.00000000	1.00000000

Table 9.6 Optimal portfolio return, variance, and standard deviation

	Optimal portfolio PSO	Optimal portfolio Nonlinear GRG
Portfolio return	0.0006485847	0.0006514079
Portfolio var	0.0000208684	0.0000208690
Portfolio std dev	0.0045681989	0.0045682613

Obtained from PSO and nonlinear GRG techniques

Table 9.7 Coefficient of variation of optimum portfolios

	PSO	Nonlinear GRG
Coefficient of variation (standard dev./return)	7.043334356	7.01290436

According to the Markowitz portfolio theory, if the correlation coefficients of two asset returns are less than 1, the total risk of that portfolio constantly decreases (Markowitz, 1952). Indeed, if the correlation coefficient is negative, the total risk of

the portfolio can be decreased much more. However, it is an exceedingly difficult situation to be a negative correlation coefficient of two assets in real life. PSO is one of the techniques that can be used to determine the optimum portfolio. The technique depends on animals' environment. PSO used the ability of animals such as birds and fish to adapt to their environment by applying a "knowledge-sharing" approach, finding rich food sources, and avoiding predators. This chapter deals with portfolio selection problem and tries to indicate how to select financial assets to conduct optimal portfolio between BIST-30 index stocks by using the particle swarm optimization (PSO) technique, which is the heuristic algorithm. In addition, to compare the results, the optimization problem is solved by nonlinear GRG techniques. Then, the results of both techniques are compared.

The data set analyzed in the chapter is organized from simultaneous stocks of BIST-30 index for the period of June 2016–July 2018. The problem is coded with MATLAB to evaluate the optimal portfolio algorithm that requires a solution. By using these returns, the optimum portfolio with minimum risk is created with PSO and nonlinear GRG techniques.

The results obtained show that both methods give similar results. The results obtained by both using PSO and nonlinear GRG techniques give close values. The coefficients of variation are remarkably close. Since quite similar results are obtained from two different techniques, it also proves the reliability of the techniques. The application of PSO in solving optimization problems could be the very facilitator in real financial life.

References

- Clerc, M. (1999). The swarm and the queen: Towards a deterministic and adaptive particle swarm optimization. In *Proceedings of the 1999 congress on evolutionary computation, 1999. CEC 99* (Vol. 3, p. 1957). Washington, D.C.: IEEE.
- Eberhart, R., & Kennedy, J. (1995, November). Particle swarm optimization. In *Proceedings of the IEEE international conference on neural networks* (Vol. 4, pp. 1942–1948). Citeseer.
- Markowitz, H. (1952). Portfolio selection. *The Journal of Finance*, 7(1), 77–91. March. 1952. www.jstor.org.proxy.lib.chalmers.se/stable/10.2307/2975974?origin=api (2012-1030).
- Shi, Y., & Eberhart, R. (2008). Population diversity of particle swarms. In *IEEE congress on evolutionary computation. CEC 2008. IEEE World Congress on Computational Intelligence* (pp. 1063–1067). Washington, D.C.: IEEE.

Chapter 10

Cardinality-Constrained Higher-Order Moment Portfolios Using Particle Swarm Optimization



Mulazim-Ali Khokhar, Kris Boudt, and Chunlin Wan

Abstract Particle swarm optimization (PSO) is often used for solving cardinality-constrained portfolio optimization problems. The system invests in at most k out of N possible assets using a binary mapping that enforces compliance with the cardinality constraint. This may lead to sparse solution vectors driving the velocity in PSO algorithm. This sparse-velocity mapping leads to early stagnation in mean-variance-skewness-kurtosis expected utility optimization when k is small compared to N . A continuous-velocity driver addresses this issue. We propose to combine both the continuous- and the sparse-velocity transformation methods so that it updates local and global best positions based on both the drivers. We document the performance gains when k is small compared to N in the case of mean-variance-skewness-kurtosis expected utility optimization of the portfolio.

Keywords Particle swarm optimization · Cardinality mapping · Higher-order moment portfolio

M.-A. Khokhar

Solvay Business School, Vrije Universiteit Brussel, Ixelles, Belgium

Department of Business Administration, Sukkur IBA University, Sukkur, Pakistan
e-mail: mulazim.ali.khokhar@vub.be

K. Boudt

Solvay Business School, Vrije Universiteit Brussel, Ixelles, Belgium

School of Business and Economics, Vrije Universiteit Amsterdam, Amsterdam, Netherlands

Department of Economics, Ghent University, Ghent, Belgium
e-mail: kris.boudt@ugent.be

C. Wan (✉)

School of Economics, Sichuan University, Chengdu, Sichuan, China

10.1 Introduction

How to invest in risky assets when the portfolio formation is subject to a cardinality constraint? Boudt and Wan (2019) show that particle swarm optimization (PSO) is useful for cardinality-constrained mean-variance utility optimization. This objective function is suboptimal in case of risk-averse investors and non-normal distributions as Jondeau and Rockinger (2006) report a non-negligible opportunity cost of ignoring higher-order moments in such settings. The solution may lie in portfolio construction for utility preferences beyond mean-variance utility. The investors with constant absolute risk aversion and positive marginal utility would prefer odd moments and avoid even moment (Jondeau & Rockinger, 2006). Such utility preferences may be summed up into a single expected utility function using Taylor expansion (Martellini & Ziemann, 2010). The problem at hand may be termed as cardinality-constrained mean-variance-skewness-kurtosis expected utility optimization.

The cardinality constraint transforms the efficient frontier to be discontinuous (Chang, Meade, Beasley, & Sharaiha, 2000) and renders the gradient-based traditional portfolio optimization methods to be no longer suitable for such a mixed integer non-linear programming. To solve portfolio optimization problem with a cardinality constraint, heuristic methods are needed (Crama & Schyns, 2003). Examples include genetic optimization, differential evolution, tabu search, simulated annealing, and particle swarm optimization (PSO), are available.

The PSO algorithm was first put forwards by Kennedy and Eberhart (1995). It has roots in artificial intelligence and animal communication strategy. PSO algorithm is built on optimal communication strategy where each particle in a flock learns from their personal best position and from global best particle. Thus, any particle in the flock will move towards the global best while improving personal positions with a velocity that is a linear function of its inertia, distance from its personal best and from the global best at the same time.

The traditional PSO algorithm does not account for the cardinality constraint. This deficiency was addressed by Deng, Lin, and Lo (2012) through a binary decision variable that sets the smallest weights to zero, hence affecting velocity of particles. Boudt and Wan (2019) show that this sparse approach in the velocity specification may lead to early stagnation and recommend using the non-transformed solution in the form of continuous-velocity PSO algorithm. The chapter proposes a mixed continuous-sparse PSO algorithm that accounts for both sparsity and continuity in particle velocity information and takes the best position.

The chapter extends the existing literature in two ways. First, the cardinality-constrained mean-variance optimization problem is extended to mean-variance-skewness and mean-variance-skewness-kurtosis optimization problems. Second, the velocity equation in sparse-PSO approach is updated to mixed continuous-sparse-velocity driver.

The chapter investigates two different higher-order moment objective functions corresponding to mean-variance-skewness (MVS) and mean-variance-skewness-kurtosis (MVK) utility preferences. The investigation involves six different cardinality-constrained portfolios where k is 10, 15, 20, 30, 40, or 50. Each portfolio

is evaluated for three different particle velocity methods, namely, sparse, continuous, and continuous-sparse, and three different acceleration coefficient methods, namely, constant, time-variance, and constriction. The proposed continuous-sparse-velocity driver, when compared to sparse and continuous drivers, solves the early stagnation problem in some of the considered cases and produces early gains.

The remaining parts of the chapter are organized in three sections. The next section explains the formulation of cardinality-constrained higher-order moment utility optimization problem and PSO algorithm setups to solve it. Section 10.3 illustrates the performance of the various PSO implementations for the portfolio optimization problem. The last section concludes the chapter with final remarks.

10.2 Methodology

This section first defines the cardinality-constrained portfolio optimization problem for higher-order moment preferences. It then introduces the particle swarm optimization algorithms.

10.2.1 *Cardinality-Constrained Portfolio Optimization Problem*

A cardinality-constrained optimization problem may be solved by using a binary decision variable as proposed by Deng et al. (2012). To optimize function f , a decision-maker chooses k out of N -dimensional decision vector w , where k elements of N can be non-zero.

$$\max_w f(w) \text{ s.t. } w \in \{0, 1\}^N, w'l \leq k \quad (10.1)$$

with l the N -dimensional vector of ones.

We focus on objective functions that depend on the first four moments of the portfolio return distribution. For the stock returns $r = (r_1, \dots, r_N)'$, the first four central moments mean μ , covariance Σ , coskewness Φ , and cokurtosis Ψ may be defined as

$$\begin{aligned} \mu &= E[r], \\ \Sigma &= E[(r - \mu)(r - \mu)'], \\ \Phi &= E[(r - \mu)(r - \mu)' \otimes (r - \mu)'], \\ \Psi &= E[(r - \mu)(r - \mu)' \otimes (r - \mu)' \otimes (r - \mu)'], \end{aligned} \quad (10.2)$$

where \otimes is Kronecker product (e.g. Martellini and Ziemann (2010); Boudt, Cornilly, Van Holle, and Willems (2020)).

Then the first four moments of the portfolio return are

$$\begin{aligned} m_1 &= w' \mu, \\ m_2 &= w' \Sigma w, \\ m_3 &= w' \Phi(w \otimes w), \\ m_4 &= w' \Psi(w \otimes w \otimes w). \end{aligned} \quad (10.3)$$

We consider a portfolio selection problem that aims at maximizing the investor's expected utility. If the utility function U is n -th order differentiable, the investors' utility of terminal wealth $1 + w'r$ may be approximated by

$$U(1 + w'r) \approx U(1 + w'\mu) + \sum_{i=1}^n \left[\frac{U^{(i)}(1 + w'\mu)}{i!} (w'r - w'\mu)^i \right], \quad (10.4)$$

where μ is the expected return on investment and w is the vector of weights that define proportion of investments. We use $U^{(i)}(x)$ to denote the i^{th} order derivative of utility function evaluated in x . The expected utility may therefore be defined as function of portfolio's central moments.

The chapter evaluates two objective functions $f_{\text{MVS}}(w)$ and $f_{\text{MVK}}(w)$ which represent utility maximization problem for mean-variance-skewness and mean-variance-skewness-kurtosis preferences, respectively. The corresponding objective functions, for investor with constant absolute risk aversion γ and invested capital proportion w in the portfolio, may be defined by

$$f_{\text{MVS}}(w) = m_1(w) - \frac{\gamma}{2} m_2(w) + \frac{\gamma(\gamma+1)}{6} m_3(w), \quad (10.5)$$

$$\begin{aligned} f_{\text{MVK}}(w) &= m_1(w) - \frac{\gamma}{2} m_2(w) + \frac{\gamma(\gamma+1)}{6} m_3(w) \\ &\quad - \frac{\gamma(\gamma+1)(\gamma+2)}{24} m_4(w). \end{aligned} \quad (10.6)$$

These objective functions have been used in Martellini and Ziemann (2010), Jondeau and Rockinger (2006), and Boudt, Lu, and Peeters (2015), amongst others.

10.2.2 Particle Swarm Optimization

Kennedy and Eberhart (1995) proposed an algorithm of optimization that mimics social behaviour of birds in a flocks or insects in a swarm. It is an evolutionary computing technique (a heuristic method) that assumes coexistence and cooperation amongst the pooled particles to find an optimal solution of given problem.

Each particle i at each time t moves from its current position $w_i(t)$ with velocity $v_i(t)$ towards a better position $w_i(t+1)$. Each particle then shares the information

about their best position $w_{li}(t + 1)$ to direct everyone towards global best position w_g of a swarm. Collectively the particles' position at time t may be represented as

$$w_i(t) = (w_{i1}(t), w_{i2}(t), \dots, w_{iN}(t))', \quad (10.7)$$

and their velocity as

$$v_i(t) = (v_{i1}(t), v_{i2}(t), \dots, v_{iN}(t))'. \quad (10.8)$$

Once the information is shared, each particle adjusts its position towards its current best and global best, which are known as cognitive and social behaviour adjustments, respectively. The improved current position of a particle depends on the inertia θ , its velocity $v_i(t - 1)$, and its acceleration factor for cognitive and social behavioural adjustments defined here as c_1 and c_2 , respectively. The cognitive and social behaviours also include some randomness. This is incorporated by multiplying c_1 and c_2 coefficients with r_1 and r_2 random numbers, respectively. Collectively the particle velocity and position adjustment system may be represented as in the following equation

$$\begin{aligned} v_i(t) &= \theta v_i(t - 1) + c_1 r_1(t)[w_{li}(t - 1) - w_i(t)] + c_2 r_2(t)[w_g(t - 1) - w_i(t - 1)], \\ w_i(t) &= w_i(t - 1) + v_i(t), \end{aligned} \quad (10.9)$$

where $w, c_1, c_2 \geq 0$, and r_1, r_2 are uniform random variables from $[0, 1]$ and c_1, c_2 are the fixed acceleration factors for cognitive and social behaviour adjustments, respectively. For further references on Eq. (10.9), please see Eberhart and Shi (2001), Deng et al. (2012), and Boudt and Wan (2019), amongst others Algorithm 10.1.

Algorithm 10.1: Particle Swarm Optimization (PSO)

-
1. Set n to the total number of N -dimensional particles in the swarm and T to the maximum number of iterations.
 2. Initialize the current positions and velocities of the n particles as random draws from the $[0, 1]$ and $[-1, 1]$ uniform distributions, respectively.
 3. Set the local best position w_{li} of particle i to its current position.
 4. Set $w_g = w_{lg}$ to the global best position, where $g = \arg \max_{1 \leq i \leq n} f(w_{li})$.
 5. For $t = 1 : T$
 - 5.1 For each particle $i = 1, \dots, N$
 - Set $v_i(t) = \theta v_i(t - 1) + c_1 r_1(t)[w_{li}(t - 1) - w_i(t)] + c_2 r_2(t)[w_g(t - 1) - w_i(t - 1)]$
 - Set $w_i(t) = w_i(t - 1) + v_i(t)$
 - End
 - 5.2 For each particle $i = 1, \dots, N$
 - If $f(w_i(t)) > f(w_{li})$, set $w_{li} = w_i(t)$
 - If $f(w_{li}) > f(w_g)$, set $w_g = w_{li}$
 - End
 6. Set $w_G = w_g$.
-

The application of traditional PSO algorithm is not adequate for portfolio optimization problem in Eq. (10.1) as it does not take the cardinality constraint into account. It is modified with the help of a binary decision variable to account for the cardinality constraint as proposed by Deng et al. (2012) and is elaborated in Algorithm 10.2. At each point in time, the algorithm optimizes by selecting the best asset combination that maximizes the utility.

It is further assumed that a cardinality-constrained investor optimizes his/her portfolio invested in k out N assets with full investment constraint. Given the cardinality constraint, a binary mapping function is used to choose K largest values from N assets. The corresponding mapping for each iteration t is given as

$$h_t(w) = I[w > w_{(N-K)}(t)], \quad (10.10)$$

where $w_{(N-K)}(t)$ represents the largest $K - N$ values in $w(t)$ and $I[.]$ the vector valued indicator function which generates one for first K largest values from N assets and zero otherwise. This is the same mapping function as in Boudt and Wan (2019).

Algorithm 10.2: Sparse-Velocity PSO

-
1. Set n to the total number of N -dimensional particles in the swarm and T to the maximum number of iterations.
 2. Initialize the current positions and velocities of the n particles as random draws from the $[0,1]$ and $[-1,1]$ uniform distribution, respectively.
 3. Set $\tilde{w}_i = h(w_i)$.
 4. Set the local best position $\tilde{w}_{li} = \tilde{w}_i$.
 5. Set $w_g = w_{lg}$ to the global best position, where $g = \arg \max_{1 \leq i \leq n} f(\tilde{w}_{li})$.
 6. For $t = 1 : T$
 - 6.1 For each particle $i = 1, \dots, N$
 - Set $v_i(t) = \theta v_i(t-1) + c_1 r_1(t)[\tilde{w}_{li}(t-1) - \tilde{w}_i(t)] + c_2 r_2(t)[\tilde{w}_g(t-1) - \tilde{w}_i(t-1)]$.
 - Set $w_i(t) = \tilde{w}_i(t-1) + v_i(t)$.
 - End
 - 6.2 Set $\tilde{w}_i(t) = h(w_i(t))$.
 - 6.3 For each particle $i = 1, \dots, N$
 - If $f(\tilde{w}_i(t)) > f(\tilde{w}_{li})$, set $\tilde{w}_{li} = \tilde{w}_i(t)$.
 - If $f(\tilde{w}_{li}) > f(\tilde{w}_g)$, set $\tilde{w}_g = \tilde{w}_{li}$.
 - End
 7. Set $w_G = \tilde{w}_g$.

Note: See Boudt and Wan (2019) for further explanation.

However, the results from Algorithm 10.2 have a risk of early stagnation (Boudt & Wan, 2019). This is because the velocity is sparse and only k out of N of its elements is non-zero. Hence, its maximum capacity to change the elements is $2K$

which when less than N may cause stagnation. To solve early stagnation problem, Boudt and Wan (2019) proposed a continuous-velocity (CV-PSO) algorithm that uses continuous solution vector w that is not affected by the cardinality-induced mapping function. This allows for the particle position and velocity information to preserve and pass on from iteration to iteration. Thus, the CV-PSO is more flexible and thus reduces chances of early stagnation. The CV-PSO also shows early gains in small k portfolios compared to N . The continuous-velocity PSO is shown in Algorithm 10.3.

Algorithm 10.3: Continuous-Velocity PSO

-
1. Set n to the total number of N -dimensional particles in the swarm and T to the maximum number of iterations.
 2. Initialize the current positions and velocities of the n particles as random draws from the $[0,1]$ and $[-1,1]$ uniform distribution, respectively.
 3. Set $\tilde{w}_i = h(w_i)$.
 4. Set the local best position $w_{li} = w_i$ and $\tilde{w}_{li} = h(w_{li})$.
 5. Set $w_g = w_{lg}$ to the global best position, where $g = \arg \max_{1 \leq i \leq n} f(w_{li})$.
 6. For $t = 1 : T$
 - 6.1 For each particle $i = 1, \dots, N$
 - Set $v_i(t) = \theta v_i(t - 1) + c_1 r_1(t)[w_h(t - 1) - w_i(t)] + c_2 r_2(t)[w_g(t - 1) - w_i(t - 1)]$.
 - Set $w_i(t) = w_i(t - 1) + v_i(t)$.
 - End
 - 6.2 Set $\tilde{w}_i(t) = h(w_i(t))$.
 - 6.3 For each particle $=1, \dots, N$
 - If $f(\tilde{w}_i(t)) > f(\tilde{w}_{li})$, set $w_{li} = w_i(t)$ and $\tilde{w}_{li} = h(w_{li})$.
 - If $f(\tilde{w}_{li}) > f(\tilde{w}_g)$, set $w_g = w_{li}$ and $\tilde{w}_g = h(w_g)$.
 - End
 7. Set $w_G = h(w_g)$.
-

Note: See Boudt and Wan (2019) for further explanation.

The CV-PSO helps resolve early stagnation in small k portfolios and shows some gains in results, but when k is large enough compared to N , the results from SV-PSO algorithm are more promising (Boudt & Wan, 2019). The SV-PSO algorithm for large k might allow frequent recurrence of same security in each iteration. This phenomenon allows for selective continuity of information due to randomness of asset selection that results in better long-run optimized values.

This chapter proposes to use both continuous solution vector w and the sparse solution vector \tilde{w} for velocity transformation such that the best of w or \tilde{w} updates local and global best positions based on both the information sets. This improves the particle position from iteration to iteration as it is conditioned to select the best performing particle through both information sets when k is smaller than N . We call it continuous-sparse-velocity (CSV-PSO) and explain it in Algorithm 10.4.

Algorithm 10.4: Continuous-Sparse-Velocity PSO

1. Set n to the total number of N -dimensional particles in the swarm and T to the maximum number of iterations.
 2. Initialize the current positions and velocities of the n particles as random draws from the $[0,1]$ and $[-1,1]$ uniform distribution, respectively.
 3. Set $\tilde{w}_i = h(w_i)$.
 4. Set the local best position $w_{li} = w_i$ and $\tilde{w}_{li} = h(w_{li})$.
 5. Set $w_g = w_{lg}$ to the global best position, where $g = \arg \max_{1 \leq i \leq n} f(w_{li})$.
 6. For $t = 1 : T$.
 - 6.1 For each particle $i = 1, \dots, N$
 - Set $v_i(t) = \theta v_i(t-1) + c_1 r_1(t)[\tilde{w}_{li}(t-1) - \tilde{w}_i(t)] + c_2 r_2(t)[\tilde{w}_g(t-1) - \tilde{w}_i(t-1)]$.
 - Set $w_i(t) = \tilde{w}_i(t-1) + v_i(t)$.
 - Set $\bar{v}_i(t) = \theta \bar{v}_i(t-1) + c_1 r_1(t)[w_{li}(t-1) - w_i(t)] + c_2 r_2(t)[w_g(t-1) - w_i(t-1)]$.
 - Set $\bar{w}_i(t) = \bar{w}_i(t-1) + \bar{v}_i(t)$.
 - 6.2 Set $\tilde{w}_i(t) = h(w_i(t))$ and $\tilde{\bar{w}}_i(t) = h(\bar{w}_i(t))$.
 - 6.3 For each particle $i = 1, \dots, N$
 - If $f(\tilde{w}_i(t)) > f(\tilde{w}_{li})$, set $w_{li} = w_i(t)$ and $\tilde{w}_{li} = h(w_{li})$.
 - If $f(\tilde{\bar{w}}_i(t)) > f(\tilde{w}_{li})$, set $w_{li} = \bar{w}_i(t)$ and $\tilde{w}_{li} = h(w_{li})$.
 - If $f(\tilde{w}_{li}) > f(\tilde{w}_g)$, set $w_g = w_{li}$ and $\tilde{w}_g = h(w_g)$.
 7. Set $w_G = h(w_g)$
-

10.3 Simulation-Based Performance Evaluation

In this section, simulated data sets of $N = 100$ and $N = 500$ assets are used to evaluate the cardinality-constrained mean-variance-skewness-kurtosis (MVS K) and mean-variance-skewness (MVS) utility optimization problems, respectively. The investors are assumed to be risk-averse with $\gamma = 10$ for all cases. Further, six different cardinality-constrained portfolios are formed by setting k equal to 10, 15, 20, 30, 40, and 50. The asset position is set to be long only $0 \leq w_i \leq 1$ with minimum number of securities equal to 1. The objective functions are then solved using particle swarm optimization with six parametric variations. The parametric variations include three different velocity methods, namely, sparse-velocity, continuous-velocity, and continuous-sparse-velocity, and three different acceleration methods, namely, constant, time-variant, and constriction coefficient. For each objective function, 54 setups search $T = 1000$ iterations for their optimal solution. The results presented are an average of 10 repetitions.

To explain this, the section is divided into four sub-sections of which the first subsection explains the properties of simulated data, second subsection explains PSO parametric settings, and the third and the final sections present the results.

10.3.1 Properties of Asset Returns

The study considers stylized setup where the stocks in the portfolio are set with mean excess returns (μ) to 3% and zero for the rest. The annualized standard deviation (σ) is set to 15% for all stocks with zero correlation between each other. While their third and fourth moment marginals are set as

$$\begin{aligned} \sigma_i^3 &= (-1)^i, \\ \sigma_i^4 &= \left(3 + i - 4\left\lceil\frac{i}{4}\right\rceil\right), \end{aligned} \quad (10.11)$$

where σ_i is annualized standard deviation of the return of stock i in the portfolio. $\lceil x \rceil$ is a ceiling function that returns the smallest integer value greater than the given number x .

10.3.2 Parameters of PSO

The swarm particle velocity is restricted to be between $[-1, 1]$ and is random for each particle in each iteration. Reaching a target may require many attempts (iterations). It is common practice in PSO algorithm application to set maximum number of iterations T to 1000 and the swarm size n to 1000 particles. The inertia weight for particle velocity is set to 0.4, while the acceleration coefficient is taken from three different studies. The first, the acceleration coefficients are set to be constant with $c_1 = c_2 = 2$, which is a common practice (Boudt & Wan, 2019).

Second, the acceleration coefficient is set using constriction coefficients χ proposed by Clerc and Kennedy (2011), where χ is defined as

$$\chi = \frac{2\kappa}{|2 - \phi - \sqrt{(\phi^2 - 4\phi)}|}, \quad (10.12)$$

with kappa $\kappa \in [0, 1]$ and $\phi = \phi_1 + \phi_2 \geq 4$.

It is common practice to set $\kappa = 1$ and $\phi_1 = \phi_2 = 2.05$. The constriction setting transforms the inertia θ , c_1 , and c_2 to

$$\begin{aligned}\theta &= \chi, \\ c_1 &= \chi\phi_1, \\ c_2 &= \chi\phi_2.\end{aligned}\tag{10.13}$$

Third, the time-varying acceleration coefficients are set as proposed in Ratnaweera, Halgamuge, and Watson (2004) given as follows

$$\begin{aligned}c_1(t) &= \frac{(c_{1,\min} - c_{1,\max})t}{T} + c_{1,\max}, \\ c_2(t) &= \frac{(c_{2,\max} - c_{2,\min})t}{T} + c_{2,\max},\end{aligned}\tag{10.14}$$

where $c_{1,\max} = c_{2,\max} = 2.5$, $c_{1,\min} = c_{2,\min} = 0.5$, T is total number of iterations, and t is the index of iterations.

10.3.3 Mean-Variance-Skewness Utility Optimization

In this section we use the PSO to find the optimal value for the MVS portfolio optimization with 500 securities. We use six different cardinality constrained portfolios by setting k to 10, 15, 20 30, 40, or 50. Three different particle swarm optimization (PSO) methods, *sparse-velocity* (SV), *continuous-velocity* (CV), and *continuous-sparse-velocity* (CSV), are used (as explained in Algorithms 10.2–10.4). Further, three different acceleration coefficient methods, *constant*, *constriction*, and *time-variant* are used (as explained in Sect. 10.3.2). All the results of MVS-utility optimization problem for six cardinality-constrained portfolios are presented in Table 10.1 and Figs. 10.1, 10.2, and 10.3 as the mean over $S = 10$ replications. As such we account for the randomness of the generated data.

The results for cardinality-constrained mean-variance-skewness utility optimization problem using sparse-PSO methods in Algorithms 10.2–10.4 for *constant acceleration coefficient method* $c_1 = c_2 = 2$ and for all cardinality levels are shown in Fig. 10.1. The results show some early gains for CSV-PSO over CV-PSO and SV-PSO methods for $k \leq 15$ and better end-of-iterations results when $k \geq 40$. Further, the early stagnation of SV-PSO method and better continuity of velocity information in CV and CSV PSO are evident for constant acceleration method (see Fig. 10.1).

The results for *constriction acceleration coefficient method* can be seen in Fig. 10.2. The SV-PSO algorithm seems to completely stagnate utility values which may be due to increased inertia with $\theta = \chi = 0.72$ and reduce weights of acceleration coefficients to $c_1 = c_2 = 1.49$. The CV-PSO algorithm generates better MVS-utility values for $k = 20$, $k = 40$, and $k = 40$. For only $k = 30$, the CSV-PSO generates better results. The results for *time-variant acceleration coefficient method* can be seen in Fig. 10.3. It shows better MVS-utility values for SV-PSO in all considered cases except when $k = 10$.

10.3.4 Mean-Variance-Skewness-Kurtosis Utility Optimization

In this section, we use the PSO to find the optimal value for the MVSK portfolio optimization with 100 securities. We take less number of securities in the universe as the number of parameters to be estimated for four comments when $N = 500$ is almost 2.6 billion, which is beyond the memory available in ordinary computing units. With $N = 100$ securities, we still estimate around 2.6 million parameters. We use six different cardinality-constrained portfolios by setting k to 10, 15, 20, 30, 40, or 50. Three different particle swarm optimization (PSO) methods, *sparse-velocity* (SV), *continuous-velocity* (CV), and *continuous-sparse-velocity* (CSV) PSO, are used, as explained in Algorithms 10.2–10.4. Further, three different acceleration coefficient methods, *constant*, *constriction*, and *time-variant coefficients*, are used (as explained in Sect. 10.3.2). To account for the randomness of the generated data, the results of MVSK utility for six cardinality-constrained portfolios are averaged for $S = 10$ repetitions and are presented in Table 10.2 and Figs. 10.4, 10.5, and 10.6.

The early stagnation for SV-PSO method and superior results of CV-PSO and CSV-PSO methods in terms of MVSK utility are prominent using *constriction acceleration coefficient method* which can be seen in Table 10.2 and Fig. 10.4. The CSV-PSO method shows some early gains over CV-PSO method in some cases here.

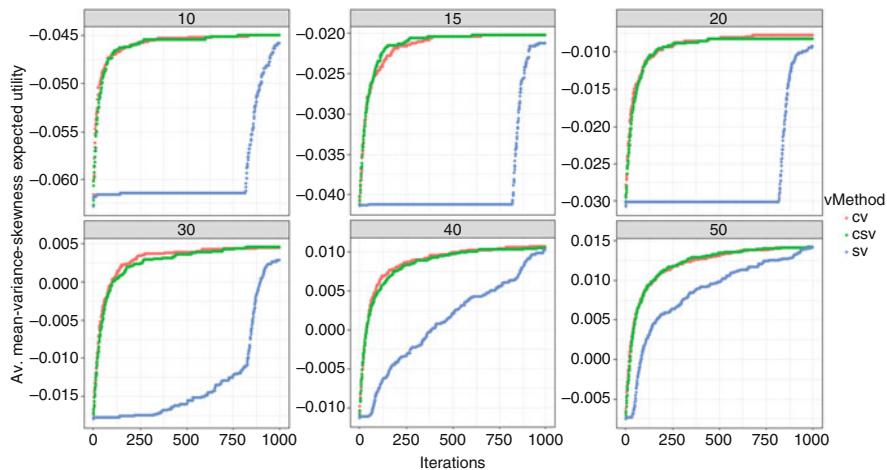
The MVSK expected utility for *constant acceleration coefficient method* where $c_1 = c_2 = 2$ is almost the same for all velocity methods and for all cardinality levels (see Fig. 10.4). The results for *time-variant acceleration coefficient method* show superior overall results for SV-PSO method except when $k = 10$ (see Fig. 10.6).

Table 10.1 The effects of velocity and acceleration coefficients' specifications on average mean-variance-skewness utility for the cardinality constrained portfolio invested in k out of N risky assets, where $k \in \{10, 15, 20, 30, 40, 50\}$ and $N = 500$

Method	Iterations	$K = 10$			$K = 15$			$K = 20$		
		$c = 2$	$c = c(t)$	$c = f\chi$	$c = 2$	$c = c(t)$	$c = f\chi$	$c = 2$	$c = c(t)$	$c = f\chi$
SV-PSO	10	-0.0619	-0.0617	-0.0623	-0.0414	-0.0413	-0.0413	-0.0302	-0.0299	-0.0301
	50	-0.0616	-0.0614	-0.0623	-0.0413	-0.0412	-0.0413	-0.0302	-0.0299	-0.0300
	100	-0.0616	-0.0614	-0.0621	-0.0413	-0.0412	-0.0413	-0.0302	-0.0296	-0.0300
	250	-0.0614	-0.0613	-0.0621	-0.0413	-0.0369	-0.0413	-0.0302	-0.0130	-0.0300
	500	-0.0614	-0.0602	-0.0621	-0.0413	-0.0227	-0.0413	-0.0302	-0.0106	-0.0300
	750	-0.0614	-0.0565	-0.0621	-0.0413	-0.0205	-0.0413	-0.0302	-0.0098	-0.0300
	1000	-0.0458	-0.0473	-0.0620	-0.0213	-0.0205	-0.0413	-0.0093	-0.0097	-0.0300
	10	-0.0562	-0.0553	-0.0584	-0.0381	-0.0344	-0.0386	-0.0247	-0.0221	-0.0275
	50	-0.0488	-0.0475	-0.0530	-0.0277	-0.0266	-0.0328	-0.0149	-0.0148	-0.0216
	100	-0.0471	-0.0463	-0.0511	-0.0246	-0.0245	-0.0306	-0.0118	-0.0132	-0.0183
CV-PSO	250	-0.0458	-0.0457	-0.0480	-0.0217	-0.0220	-0.0269	-0.0094	-0.0120	-0.0147
	500	-0.0453	-0.0452	-0.0465	-0.0205	-0.0215	-0.0231	-0.0083	-0.0113	-0.0123
	750	-0.0451	-0.0451	-0.0457	-0.0203	-0.0215	-0.0219	-0.0078	-0.0113	-0.0111
	1000	-0.0450	-0.0451	-0.0457	-0.0203	-0.0215	-0.0213	-0.0078	-0.0113	-0.0103
	10	-0.0577	-0.0558	-0.0589	-0.0376	-0.0348	-0.0383	-0.0257	-0.0222	-0.0273
	50	-0.0495	-0.0479	-0.0533	-0.0280	-0.0259	-0.0312	-0.0165	-0.0146	-0.0220
	100	-0.0473	-0.0466	-0.0499	-0.0236	-0.0241	-0.0285	-0.0120	-0.0128	-0.0184
	250	-0.0458	-0.0457	-0.0474	-0.0211	-0.0226	-0.0255	-0.0090	-0.0120	-0.0141
	500	-0.0454	-0.0455	-0.0462	-0.0205	-0.0221	-0.0235	-0.0083	-0.0117	-0.0123
	750	-0.0451	-0.0455	-0.0458	-0.0203	-0.0219	-0.0224	-0.0083	-0.0115	-0.0108
	1000	-0.0450	-0.0455	-0.0457	-0.0203	-0.0219	-0.0216	-0.0083	-0.0115	-0.0101

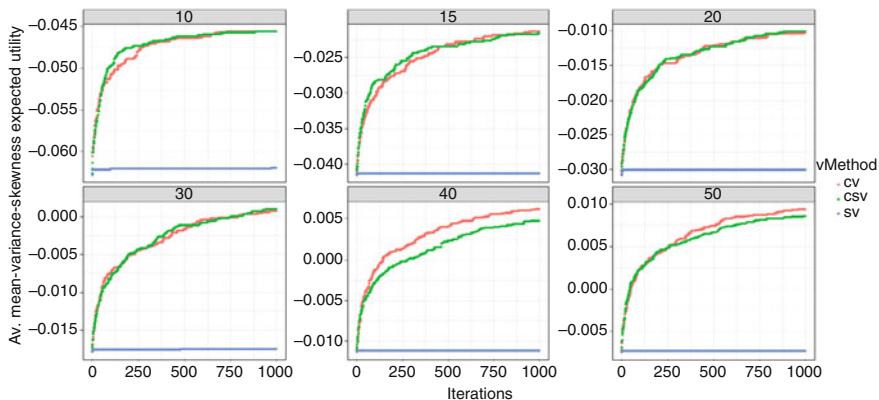
Velocity	Iterations	$K = 30$			$K = 40$			$K = 50$		
		$c = 2$	$c = c(t)$	$c = f(\chi)$	$c = 2$	$c = c(t)$	$c = f(\chi)$	$c = 2$	$c = c(t)$	$c = f(\chi)$
SV-PSO	10	-0.0178	-0.0176	-0.0176	-0.0112	-0.0111	-0.0112	-0.0074	-0.0073	-0.0074
	50	-0.0178	-0.0134	-0.0176	-0.0112	0.0009	-0.0112	-0.0060	0.0071	-0.0074
	100	-0.0178	-0.0022	-0.0176	-0.0078	0.0070	-0.0112	0.0017	0.0119	-0.0074
	250	-0.0174	0.0032	-0.0176	-0.0034	0.0109	-0.0112	0.0064	0.0147	-0.0074
	500	-0.0160	0.0049	-0.0176	0.0021	0.0110	-0.0112	0.0098	0.0149	-0.0074
	750	-0.0126	0.0049	-0.0176	0.0056	0.0112	-0.0112	0.0124	0.0150	-0.0074
	1000	0.0027	0.0050	-0.0176	0.0102	0.0113	-0.0112	0.0142	0.0150	-0.0074
	10	-0.0138	-0.0120	-0.0152	-0.0072	-0.0056	-0.0088	-0.0036	-0.0023	-0.0045
	50	-0.0037	-0.0062	-0.0103	0.0020	0.0001	-0.0038	0.0050	0.0024	-0.0002
	100	-0.0002	-0.0047	-0.0075	0.0058	0.0013	-0.0013	0.0084	0.0038	0.0022
CV-PSO	250	0.0024	-0.0025	-0.0045	0.0086	0.0028	0.0013	0.0115	0.0045	0.0048
	500	0.0038	-0.0023	-0.0018	0.0099	0.0028	0.0039	0.0130	0.0046	0.0073
	750	0.0043	-0.0023	-0.0002	0.0104	0.0029	0.0055	0.0138	0.0046	0.0087
	1000	0.0044	-0.0021	0.0007	0.0107	0.0029	0.0061	0.0141	0.0046	0.0093
	10	-0.0147	-0.0116	-0.0148	-0.0072	-0.0049	-0.0087	-0.0039	-0.0017	-0.0048
	50	-0.0050	-0.0044	-0.0103	0.0015	-0.0003	-0.0049	0.0045	0.0034	0.0005
	100	-0.0005	-0.0021	-0.0083	0.0049	0.0014	-0.0029	0.0085	0.0046	0.0022
	250	0.0023	-0.0009	-0.0043	0.0083	0.0021	-0.0003	0.0117	0.0054	0.0047
	500	0.0036	-0.0005	-0.0012	0.0097	0.0024	0.0023	0.0135	0.0055	0.0066
	750	0.0042	-0.0005	-0.0001	0.0103	0.0024	0.0039	0.0140	0.0055	0.0080
	1000	0.0045	-0.0005	0.0010	0.0105	0.0024	0.0047	0.0142	0.0055	0.0085

Note: $f(\chi)$ refers to constriction coefficient for acceleration method as explained in Sect. 10.3.2



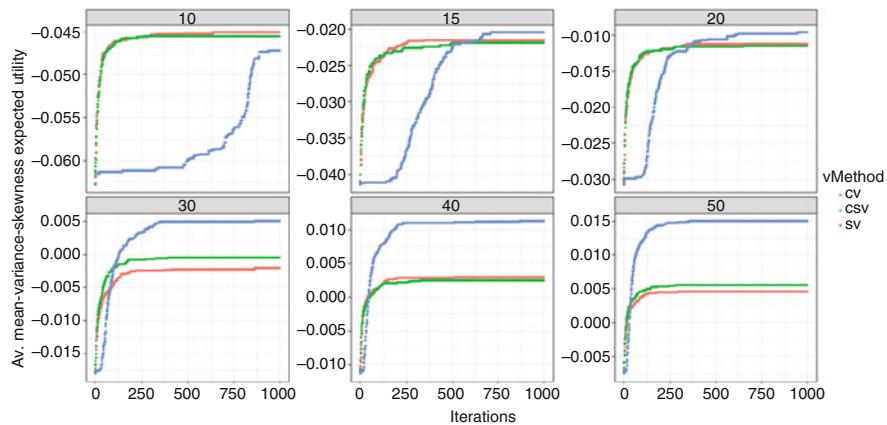
The results for constant acceleration method with $c_1=c_2=2$.

Fig. 10.1 The average MVSK-utility obtained using SV, CV, and CSV PSO algorithms with constant acceleration method ($c_1 = c_2 = 2$) for portfolios with at most 10, 15, 20, 30, 40, and 50 assets selected from $N = 500$ assets. Source: Authors' Own Creation



The results for constriction acceleration method as in Equations 12–13.

Fig. 10.2 The average MVS-utility obtained using SV, CV, and CSV PSO algorithms with constriction acceleration method (Clerc & Kennedy, 2011) for portfolios with at most 10, 15, 20, 30, 40, and 50 assets selected from $N = 500$ assets. Source: Authors' Own Creation



The results for time-variant acceleration method as in Equations 14.

Fig. 10.3 The average MVS-utility obtained using SV, CV, and CSV PSO algorithms with *time-variant acceleration method* $c_1 = c_2 = c(t)$ for portfolios with at most 10, 15, 20, 30, 40, and 50 assets selected from $N = 500$ assets. Source: Authors' Own Creation

10.4 Conclusion

The chapter contributes towards further application of particle swarm optimization in solving cardinality-constrained asset-selection problem under non-normality. It also extends the continuous-velocity PSO algorithm of Boudt and Wan (2019) to continuous-sparse-velocity approach.

The chapter investigates two different objective functions corresponding to mean-variance-skewness (MVS) and mean-variance-skewness-kurtosis (MVK) utility preferences. The investigation involves six different cardinality-constrained portfolios where k is 10, 15, 20, 30, 40, or 50. Each portfolio is evaluated for three different particle velocity methods, namely, sparse, continuous, and continuous-sparse, and three different acceleration coefficient methods, namely, constant, time-variance, and constriction.

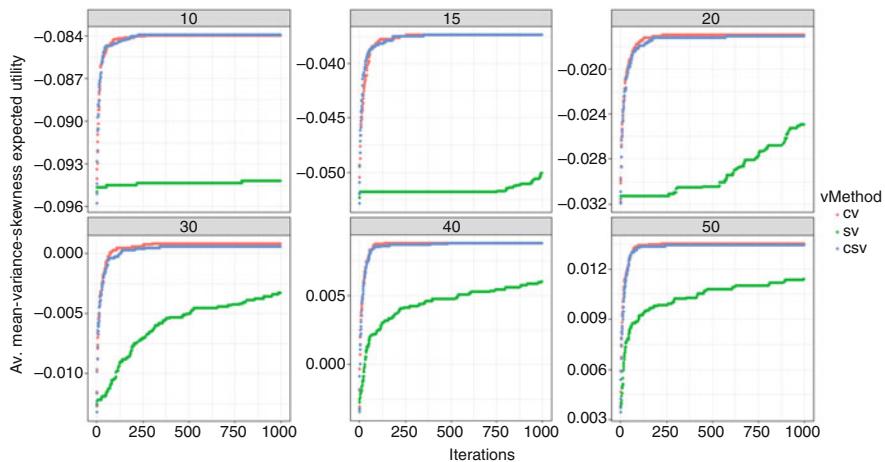
The proposed continuous-sparse-velocity PSO algorithm generates early gains in some of the cases presented. It particularly performs better when used in combination with constant or constriction acceleration methods when k is considerably small compared to N for both objective functions. The early stagnation of sparse-velocity PSO algorithm was marked in MVS-utility optimization problems in all cases, but the same was missing in case of MVSK-utility optimization except when used with constriction method. It was also marked that the sparse-velocity PSO algorithm when set up with constriction coefficient produced early stagnation for both objective functions.

Table 10.2 The effects of velocity and acceleration coefficients' specifications on average mean-variance-skewness-kurtosis expected utility for the cardinality-constrained portfolios invested in k out of N risky assets, where $k \in \{10, 15, 20, 30, 40, 50\}$ and $N = 100$

Method	Iterations	$K = 10$			$K = 15$			$K = 20$		
		$c = 2$	$c = c(t)$	$c = f(\chi)$	$c = 2$	$c = c(t)$	$c = f(\chi)$	$c = 2$	$c = c(t)$	$c = f(\chi)$
SV-PSO	10	-0.0953	-0.0946	-0.0946	-0.0524	-0.0478	-0.0518	-0.0313	-0.0267	-0.0313
	50	-0.0861	-0.0841	-0.0946	-0.0387	-0.0374	-0.0518	-0.0181	-0.0170	-0.0313
	100	-0.0845	-0.0840	-0.0945	-0.0379	-0.0374	-0.0518	-0.0173	-0.0170	-0.0313
	250	-0.0841	-0.0840	-0.0943	-0.0377	-0.0374	-0.0518	-0.0171	-0.0170	-0.0313
	500	-0.0841	-0.0840	-0.0943	-0.0374	-0.0374	-0.0518	-0.0170	-0.0170	-0.0305
	750	-0.0841	-0.0840	-0.0943	-0.0374	-0.0374	-0.0518	-0.0170	-0.0170	-0.0277
	1000	-0.0840	-0.0840	-0.0942	-0.0374	-0.0374	-0.0501	-0.0170	-0.0170	-0.0250
	10	-0.0873	-0.0860	-0.0902	-0.0434	-0.0446	-0.0458	-0.0235	-0.0257	-0.0254
	50	-0.0843	-0.0842	-0.0851	-0.0381	-0.0411	-0.0398	-0.0174	-0.0237	-0.0192
	100	-0.0840	-0.0841	-0.0843	-0.0377	-0.0408	-0.0380	-0.0171	-0.0234	-0.0177
CV-PSO	250	-0.0840	-0.0840	-0.0841	-0.0374	-0.0408	-0.0375	-0.0170	-0.0230	-0.0170
	500	-0.0840	-0.0840	-0.0840	-0.0374	-0.0408	-0.0374	-0.0170	-0.0230	-0.0170
	750	-0.0840	-0.0840	-0.0840	-0.0374	-0.0408	-0.0374	-0.0170	-0.0230	-0.0170
	1000	-0.0840	-0.0840	-0.0840	-0.0374	-0.0408	-0.0374	-0.0170	-0.0230	-0.0170
	10	-0.0873	-0.0870	-0.0890	-0.0427	-0.0409	-0.0453	-0.0233	-0.0225	-0.0257
	50	-0.0841	-0.0843	-0.0848	-0.0378	-0.0383	-0.0390	-0.0174	-0.0190	-0.0198
	100	-0.0840	-0.0842	-0.0846	-0.0374	-0.0379	-0.0383	-0.0170	-0.0181	-0.0181
	250	-0.0840	-0.0840	-0.0840	-0.0374	-0.0377	-0.0375	-0.0170	-0.0177	-0.0173
	500	-0.0840	-0.0840	-0.0840	-0.0374	-0.0377	-0.0374	-0.0170	-0.0175	-0.0173
	750	-0.0840	-0.0840	-0.0840	-0.0374	-0.0375	-0.0374	-0.0170	-0.0175	-0.0171
CSV-PSO	1000	-0.0840	-0.0840	-0.0840	-0.0374	-0.0375	-0.0374	-0.0170	-0.0175	-0.0171

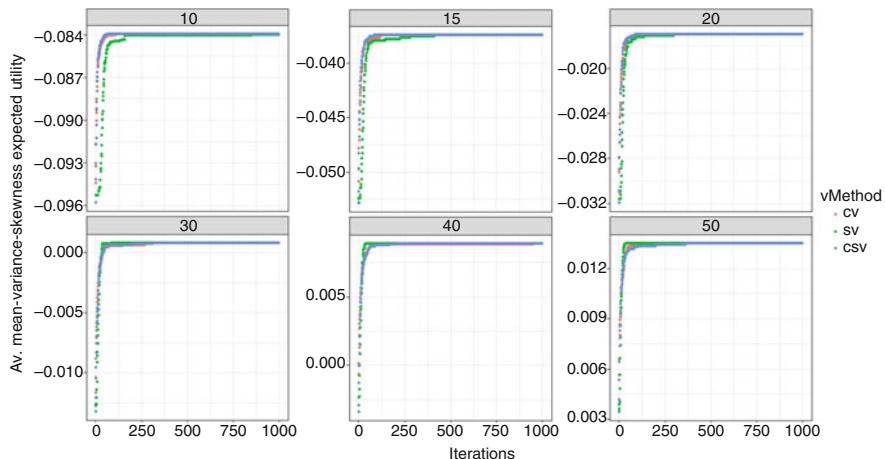
Method	Iterations	$K = 30$			$K = 40$			$K = 50$		
		$c = 2$	$c = c(t)$	$c = f(\chi)$	$c = 2$	$c = c(t)$	$c = f(\chi)$	$c = 2$	$c = c(t)$	$c = f(\chi)$
SV-PSO	10	-0.0091	0.0067	-0.0122	0.0015	0.0019	-0.0021	0.0079	0.0076	0.0049
	50	0.0007	0.0008	-0.0118	0.0089	0.0013	0.0135	0.0135	0.0082	
	100	0.0008	0.0008	-0.0105	0.0089	0.0025	0.0135	0.0135	0.0090	
	250	0.0008	0.0008	-0.0071	0.0089	0.0041	0.0135	0.0135	0.0098	
	500	0.0008	0.0008	-0.0050	0.0089	0.0047	0.0135	0.0135	0.0108	
	750	0.0008	0.0008	-0.0043	0.0089	0.0054	0.0135	0.0135	0.0110	
	1000	0.0008	0.0008	-0.0033	0.0089	0.0060	0.0135	0.0135	0.0114	
CV-PSO	10	-0.0046	-0.0042	-0.0065	0.0040	0.0038	0.0032	0.0093	0.0095	0.0088
	50	0.0003	-0.0011	-0.0010	0.0083	0.0067	0.0077	0.0131	0.0117	0.0125
	100	0.0005	-0.0006	0.0003	0.0088	0.0072	0.0087	0.0135	0.0121	0.0134
	250	0.0006	-0.0001	0.0006	0.0088	0.0076	0.0088	0.0135	0.0123	0.0135
	500	0.0008	0.0001	0.0008	0.0088	0.0078	0.0088	0.0135	0.0126	0.0135
	750	0.0008	0.0001	0.0008	0.0088	0.0079	0.0088	0.0135	0.0126	0.0135
	1000	0.0008	0.0001	0.0008	0.0089	0.0079	0.0088	0.0135	0.0126	0.0135
CSV-PSO	10	-0.0054	-0.0045	-0.0060	0.0038	0.0039	0.0025	0.0095	0.0091	0.0083
	50	0.0003	-0.0011	-0.0014	0.0083	0.0070	0.0075	0.0130	0.0117	0.0126
	100	0.0006	-0.0009	-0.0003	0.0088	0.0073	0.0085	0.0133	0.0122	0.0133
	250	0.0007	-0.0003	0.0003	0.0089	0.0078	0.0086	0.0134	0.0127	0.0134
	500	0.0008	-0.0002	0.0005	0.0089	0.0078	0.0088	0.0135	0.0128	0.0134
	750	0.0008	-0.0002	0.0005	0.0089	0.0078	0.0088	0.0135	0.0128	0.0134
	1000	0.0008	-0.0002	0.0005	0.0089	0.0078	0.0088	0.0135	0.0128	0.0134

Note: $f(\chi)$ refers to constriction coefficient for acceleration method as explained in Sect. 10.3.2



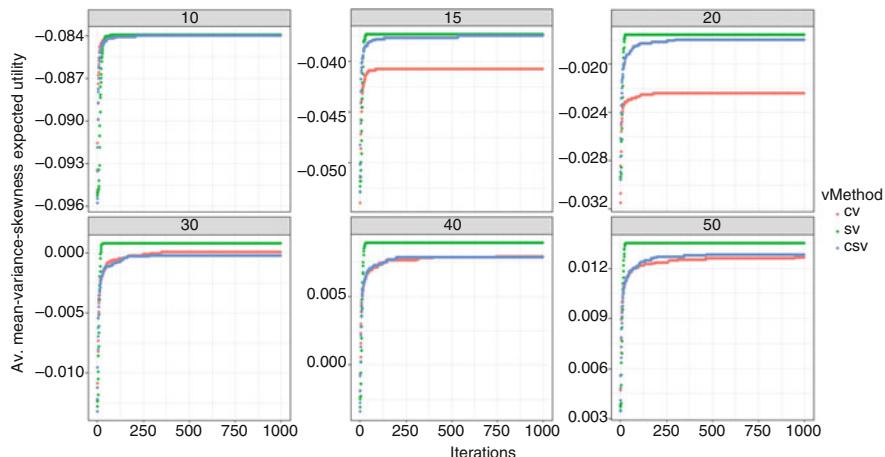
The results for constriction acceleration method as in Equations 12–13.

Fig. 10.4 The average MVSK-utility obtained using SV, CV, and CSV PSO algorithms with constriction acceleration method (Clerc & Kennedy, 2011) for portfolios with at most 10, 15, 20, 30, 40, and 50 assets selected from $N = 100$ assets. Note: The results in figure show early stagnation for SV-PSO approach, while CSV-PSO shows some early gains in some cases. Source: Authors' Own Creation



The results for constant acceleration method with $c_1=c_2=2$.

Fig. 10.5 The average MVSK-utility obtained using SV, CV, and CSV PSO algorithms with constant acceleration method ($c_1 = c_2 = 2$) for portfolios with at most 10, 15, 20, 30, 40, and 50 assets selected from $N = 100$ assets. Source: Authors' Own Creation



The results for time-variant acceleration method as in Equations 14.

Fig. 10.6 The average MVSK-utility obtained using SV, CV, and CSV PSO algorithms with time-variant acceleration method $c_1 = c_2 = c(t)$ for portfolios with at most 10, 15, 20, 30, 40, and 50 assets selected from $N = 100$ assets. Source: Authors' Own Creation

References

- Boudt, K., Cornilly, D., Van Holle, F., & Willems, J. (2020). Algorithmic portfolio tilting to harvest higher moment gains. *Heliyon*, 6(3), e03516.
- Boudt, K., Lu, W., & Peeters, B. (2015). Higher order comoments of multifactor models and asset allocation. *Finance Research Letters*, 13, 225–233.
- Boudt, K., & Wan, C. (2019). The effect of velocity sparsity on the performance of cardinality constrained particle swarm optimization. *Optimization Letters*, 14, 747–758.
- Chang, T., Meade, N., Beasley, J., & Sharaiha, Y. (2000). Heuristics for cardinality constrained portfolio optimization. *Computers and Operations Research*, 27(13), 1271–1302.
- Clerc, M., & Kennedy, J. (2011). The particle swarm—Explosion, stability, and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation*, 6(1), 2011.
- Crama, Y., & Schyns, M. (2003). Simulated annealing for complex portfolio selection problems. *European Journal of Operational Research*, 150(3), 546–571.
- Deng, G. F., Lin, W. T., & Lo, C. C. (2012). Markowitz-based portfolio selection with cardinality constraints using improved particle swarm optimization. *Expert Systems with Applications*, 39 (4), 4558–4566.
- Eberhart, R. C., & Shi, Y. (2001). Tracking and optimizing dynamic systems with particle swarms. *Proceedings of the IEEE Conference on Evolutionary Computation, ICEC*, 1, 94–100.
- Jondeau, E., & Rockinger, M. (2006). Optimal portfolio allocation under higher moments. *European Financial Management*, 12(1), 29–55.
- Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. In *Paper Presented at IEEE International Conference on Neural Networks, Perth, Australia*. Piscataway, NJ: IEEE Service Center.
- Martellini, L., & Zieman, V. (2010). Improved estimates of higher-order comoments and implications for portfolio selection. *Review of Financial Studies*, 23(4), 1467–1502.
- Ratnaweera, A., Halgamuge, S., & Watson, H. (2004). Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients. *IEEE Transactions on Evolutionary Computation*, 8(3), 240–255.

Part II

Different Applications of PSO

Chapter 11

Different Applications of PSO



Altaf Q. H. Badar

Abstract Particle swarm optimization (PSO) is an evolutionary optimization algorithm. PSO is a robust and well researched optimization technique. There are a large number of applications of PSO. “Applications of PSO” chapter tries to present a classified literature review for the applications of PSO in different fields. The applications are classified into different sections based on the area of implementation.

The chapter also presents a table with references to multiple other applications over and above those covered in the chapter. References of some largely cited review papers dealing with the applications of PSO are also mentioned at the end of the chapter.

Keywords Particle swarm optimization · Evolutionary algorithms · Application of PSO

11.1 Introduction

Optimization is a part of our daily life where we want to optimize each and every task. We want to optimize our travel plans, our monetary spending, our time and resources, and what not. In reality, we solve these problems by our own conscience, our past experiences, and the experiences of our known ones. If these problems can be converted into a model, then these problems can also be solved through an optimization technique.

What is optimization? In simple words, optimization is making the best use of available situation or resources. In reality, we can only process some of the available solutions that we know or can think of. But there are multiple optimization problems which have a lot of possible solutions which we, as humans, cannot analyze. This

A. Q. H. Badar (✉)

Electrical Engineering Department, National Institute of Technology, Warangal, India
e-mail: altafbadar@nitw.ac.in

happens because of a lot of possible combinations between the input variables. Just to get an idea, let us assume that if there are three input variables and each variable can have four different allowable values, then the number of possible solutions will be 64. If these numbers of input variables or their possible values are increased, then the number of possible solutions shall increase by a huge number.

The magnitude of the above-discussed optimization problem increases merely due to the increase in the number of possible solutions. It becomes impossible for a human to evaluate these optimization problems for all the possible solutions. Thus, we need methods to solve such optimization problems.

Evolutionary-based optimization techniques provide very simple and robust methods to solve these problems while providing the advantage of consuming less time. One of such evolutionary optimization technique is particle swarm optimization (PSO). PSO was introduced in 1995 and has been developed to perform better and better over a period of years. The same has been covered in earlier chapters of the book.

In this chapter, we shall look into various applications of PSO and its variants in different fields like engineering, medical, etc. This book itself covers an application of PSO into portfolio optimization. PSO and its applications have a huge number of publications, and it is not possible to include all of them. The chapter is divided into different sections based on the field of applications.

Since the numbers of references for some of the applications are so numerous, that all cannot be referred to; thus, one reference per application has been cited. The total number of applications has been limited to 40. Some of the review papers are cited at the end of the chapter for the benefit of the readers.

11.2 Electrical Engineering

Electrical engineering has a large number of PSO applications.

11.2.1 *Reactive Power Optimization*

In reactive power optimization problem, the objective is to reduce the total amount of current flowing in the transmission system. Various inputs considered for this problem are voltage magnitude, transformer tap settings, and capacitor bank settings. By optimal selection of input variables, the reactive part of the current, flowing in the transmission system, is reduced (Badar, Umre, & Junghare, 2012).

11.2.2 *Load Forecasting*

A very important part of job in a power system engineer's work life is the prediction of load in the future. The prediction of load is done for long term (some years),

medium term (some months), and short term (some days). PSO is used in combination of various methods like support vector machines (SVM), wavelets, etc. to predict load characteristics for short term. In (Huang, Huang, & Wang, 2005), short-term load forecasting is done using PSO along with autoregressive-moving-average model with exogenous inputs (ARMAX) model. Akaike's final prediction error (FPE) and loss function represent the objective function for the problem discussed. The basic aim is to reduce the error in the prediction process.

11.2.3 Maximum Power Point Tracking in Photovoltaic System

The power generated by the solar panel is adjusted to be maximum for the given condition by tracking the graphs representing the relations between voltage and power and voltage and current. PSO is found to give faster results with lesser oscillations and optimal results in different conditions (Ishaque, Salam, Amjad, & Mekhilef, 2012).

11.2.4 Proportional-Integral-Derivative (PID) Controller

PID controllers are used for various control applications in electrical engineering and other fields. The optimal operation of PID controller is obtained through proper tuning of its parameters, namely, K_p , K_i , and K_d . In (Solihin, Tack, & Kean, 2011), PID controller is used to control the working of a DC motor.

11.2.5 Phasor Measurement Unit (PMU) Placement

Phasor measurement units are a very important part of measuring instruments being utilized in the modern-day electrical systems. Their placement is limited with less number of PMUs being utilized while covering the whole system (Ahmadi, Alinejad-Beromi, & Moradi, 2011).

11.2.6 Economic Dispatch

One of the long-term applications of PSO in electrical engineering is the problem of economic dispatch. This problem tries to find the power plants that should supply power at a given time and how much power each plant should generate. In

(Al Bahrani & Patra, 2017), this problem is applied to modern smart grid conditions with new constraints.

11.2.7 Home Energy Management System

Home energy management is a very important field of research. It takes into account the renewable energy sources, energy storage systems, appliances (controllable and uncontrollable), tariff rates, and many other factors to reduce the cost of energy bills for a home. PSO is applied for reduction in energy costs of a home in (Pedrasa, Spooner, & MacGill, 2010). The home energy management systems usually apply demand response techniques like peak shaving, load scheduling, etc. to achieve these goals.

11.2.8 State Estimation

In an electrical system, there are numerous measurements which are taken and transmitted every second. There are different reasons due to which this measured data may contain uncertainties or noise. State estimation is applied for the estimation of power system variables through the identification of errors. In (Tungadio, Numbi, Siti, & Jimoh, 2015), state estimation is performed using PSO through two objective functions of weighted least square and weighted least absolute value.

11.2.9 Congestion Management

In electrical systems, the power is transmitted from the generating stations to the loads through transmission lines. It is necessary to follow the power transfer limits of the transmission lines. When high power flows through transmission lines, congestion happens. To avoid such situations, congestion management is required. In (Kamaraj, 2011), congestion management is used for pool-based electricity market through PSO. The security constraints considered are line loading and voltages of buses. The method is applied to two standard systems.

11.2.10 Induction Cooker Design

PSO is applied for induction heating design in (Hosseini, Kashtiban, & Alizadeh, 2006). Finite element method is applied alongside PSO to get optimal design. The objective is to minimize the cost as a function of leakage flux and electromagnetic

forces of induction heating winding. The parameters considered are DC voltage source, output power, switching frequency, and load resistance and inductance.

11.3 Electronics Engineering

11.3.1 Swarm Robots

PSO finds a very interesting application of searching a given target through a swarm of robots in (Hereford, 2006). PSO eliminates the requirement of having a central leader for coordinating the movement of the robots and is able to find the target in lesser time.

11.3.2 Antenna Design

PSO has been used to optimize antenna design for quite a long time now. In (Robinson & Rahmat-Samii, 2004), PSO is used to design a horn antenna. The antenna design is applied for different types of walls. The simulation output computes beam width, weight, return loss, and peak cross-polarization. There are large numbers of publications using PSO on antenna design optimization problem.

11.3.3 Channel Allocation

In communications, it is very necessary that high bandwidth, absence of delay, and channel availability are provided to users for better quality of service. In (Scott-Hayward & Garcia-Palacios, 2014), PSO is used for multiple application resource allocation. The objective function is based on utility function of channel time allocation for video on demand and Internet protocol television.

11.3.4 Filter Design

PSO has been applied for electronic filters in a large number of research works. In (Krusienski & Jenkins, 2004), PSO is applied on infinite impulse response filters. The mean squared error between the output of unknown system and adaptive filter is related to the cost function which acts as an objective function for the given problem.

11.3.5 Very-Large-Scale Integration (VLSI) Routing

Very-large-scale integration is used to create integrated circuits through the combination of a large number of transistors on a single chip. For such circuits, it is very important to optimize the wire lengths to limit the power delay. Rectilinear Steiner minimal tree is used to obtain the cost of such a circuit. PSO solves this problem in (Khan, Laha, & Sarkar, 2013) with approximately 20% reduction in the cost/length of wire.

11.3.6 Wireless Sensor Network

In wireless sensor network, allocation of workload for each task to proper nodes efficiently is termed as task allocation (Yang, Zhang, Ling, Pan, & Sun, 2013). The constraints for the problem are taken as task workload and connectivity. The fitness function includes time, energy, and lifetime components. The problem has been solved through many different considerations and constraints in the research publications.

11.4 Mechanical Engineering

11.4.1 Machinery Fault Detection

Machines are a very important part of our lives, without which nothing would move and these machines need bearing to run properly. PSO is used to obtain optimal input features for classifiers like SVM or artificial neural network (ANN) while finding machine bearing faults (Samanta & Nataraj, 2009).

11.4.2 Cell Formation

Formation of cells in the manufacturing of machine parts is an important step for optimal process implementation. PSO is used to reduce the setup time and travel of parts in between different cells in (Anvari, Mehrabad, & Barzinpour, 2010). A number of problems are solved to prove the performance of PSO. A similar problem for placement of inventory is handled in (Hochmuth, Lassig, & Thiem, 2011).

11.4.3 Injection Molding

Injection molding is a process of making parts by injecting molten material into a mold. In (Bensingh, Machavaram, Boopathy, & Jebaraj, 2019), PSO is used to predict optimal process parameters, and the whole problem is solved through ANN and PSO.

11.4.4 Milling

Milling is similar to metal cutting having an objective of obtaining optimal surface roughness at micro level and economic performance at macro level. In (Pare, Agnihotri, & Krishna, 2011), surface roughness is optimized, while cutting speed, feed rate, radial rake angle, and depth of cut are taken as input variables.

11.4.5 Multi-hole Extrusion

A Taguchi method and PSO combination is used to find optimal process parameters for a multi-hole extrusion process (Chen, Su, Nian, Lin, & Chen, 2013). The parameters are eccentricity ratio, extrusion velocity, friction coefficient, and billet temperature. Better values of exit tube bending angles and mandrel eccentricity are also obtained. Various softwares are used to verify the results.

11.4.6 Nuclear Power Plant Component Design

To improve the thermal efficiency of a nuclear power plant, modified PSO is proposed in (Liu, Yan, & Wang, 2014). The optimal design is generated for vertical electrical heating pressurizer in reactor coolant system. Optimization variables are primary loop operation pressure, reactor inlet and outlet coolant temperature, and pressurizer inner diameter.

11.5 Computer Engineering

11.5.1 Rule Mining

Association rule mining is a procedure which aims to observe frequently occurring patterns, correlations, or associations from datasets found in various kinds of

databases such as relational databases, transactional databases, and other forms of repositories. In (Kuo, Chao, & Chiu, 2011), PSO is applied on standard database to mine association rules for stock selection behavior.

11.5.2 Sentiment Analysis

Artificial intelligence has been developing at a very fast pace. Sentiment analysis is used to analyze the emotions expressed in the text by a human. It is an extremely helpful tool for multiple applications like social networking, product feedback, etc. PSO is used for sentiment analysis in (Gupta, Reddy, Ekbal, et al., 2015) automatic feature selection for aspect term extraction and sentiment classification.

11.5.3 Data Clustering

Clustering is used to partition data based on their similarities. The major problems in this method are compactness and separating data in clusters. In (Armano & Farmani, 2016), clustering is defined as a multi-objective problem, on the basis of connectivity and cohesion. The result gives well-defined, connected, and compact clusters. Around 27 datasets have been implemented in (Armano & Farmani, 2016).

11.5.4 Cloud Computing

Task scheduling is one of the most important requirements in the cloud computing environment. The task scheduling is responsible for efficient working of cloud computing facilities. The parameters to be considered for optimal task scheduling are time, cost, make span, availability, scalability, reliability, throughput, utilization of resources, etc. In (Awad, El-Hefnawy, & Abdel Kader, 2015), account reliability, execution and roundtrip time, make span, transmission time, cost of transmission, and balancing of load between tasks and virtual machine are considered through the implementation of PSO.

11.5.5 Fuzzy Cognitive Maps

Neural networks and fuzzy modeling methodology make up fuzzy cognitive maps, which are used to simulate complex systems. The main objective is to obtain optimal values of fuzzy cognitive map weights so as to get desirable steady-state behavior from the system while following the constraints of the system. In (Petralas,

Parsopoulos, & Vrahatis, 2009), PSO is used to find these weights, and it is verified against a number of fuzzy cognitive map applications.

11.5.6 Video-Based Face Recognition

For artificial intelligence systems, it is important to keep learning, especially when the new data becomes available during operations. Classifier systems are applied for the same, and PSO is applied to guide these classifiers in (Connolly, Granger, & Sabourin, 2012). The proposed method is applied on video-based facial recognition through a combination of fuzzy logic and artificial neural networks. PSO is used to generate and evolve diversified pools of classifiers. Real-world video streams are used to assess the method on the basis of classification rate and resources required.

11.6 Others

11.6.1 Traffic Flow Forecasting

A hybrid combination of PSO and support vector regression (SVR) model is used to predict the flow of traffic in (Hu, Yan, Liu, & Wang, 2016). In the traffic forecasting application, PSO is used to optimize the parameters of SVR.

11.6.2 Oil Well Location and Type

For the development of oil and gas fields, it is important to find optimal location and type for the new wells (Onwunalu & Durlofsky, 2010). The objective function evaluates the net present value of a location and type of well, which includes all types of costs including drilling and other costs.

11.6.3 Maintenance and Inventory Management

A novel application of PSO is found in (Samal & Pratihar, 2015) for maintenance and inventory management of spare parts. It tries to optimize the cost related to combined cost of maintenance and for purchase of spare parts. PSO provided results at par with conventional methods.

11.6.4 Vehicle Routing Problem

Vehicle routing problem is an evolving optimization problem, with practical implementation and various versions appearing over a period of years. The input data consists of number of nodes, distance between them, vehicle type, speed, etc. The objective of the problem is to reduce the time of travel while visiting all the nodes and reducing the traveling distance. In (Ai & Kachitvichyanukul, 2009), a vehicle routing problem is solved through modified PSO. In newer versions of the problem, delivery and pickup details of each node are required.

11.6.5 Submission Decision Process

A multi-objective problem is designed around article submission process and solved through PSO in (Adewumi & Popoola, 2018). The objectives are denoted by “C” for the expected number of citations, “P” for the time required from submission to acceptance for a paper, and “R” for the expected number of submissions.

11.6.6 Ship Design

Barebones PSO has been applied to design the ship in (Yao & Han, 2013). The optimization problem has three objective functions, six design variables, and nine inequality constraints. Design variables considered are length, depth, beam, draft, block coefficient, and speed in knots. Objective functions for the problem are minimization of transportation cost and lightship weight and maximization of annual cargo.

11.6.7 Propylene Reactor Application

An industrial process of Ziegler-Natta propylene polymerization in propylene reactors is solved through non-linear dynamic data reconciliation. PSO is used in (Prata, Schwaab, Lima, & Pinto, 2010) for parameter estimation for error detection. Real and system data for different systems are implemented for steady and dynamic states. The output of PSO leads to a more robust and reliable process. A Welch estimator is used in the process along with non-linear data reconciliation and parameter estimation for detection of error.

11.6.8 Fault Selection in Chemical Process

Fault classification based on a large number of variables in a chemical process industry like Tennessee Eastman process is a tedious task. Authors in (L. Wang &

Yu, 2005) solve this problem having a large amount of data along with irrelevant variables through PSO. SVM is also implemented for classification of fault. The proposed work was executed for around 25 h.

11.6.9 Vapor-Liquid Equilibrium

PSO is used for parameter estimation in (H. Zhang, Kennedy, Rangaiah, & Bonilla-Petriciolet, 2011) while developing mathematical models for analyzing vapor-liquid equilibrium process. These problems are non-linear in nature. The research work is applied on 16 problems. Approaches used in the research work are least squares and error in variable.

11.6.10 Leukemia Detection

Diagnosis of microscopic images can lead to decision support system for the detection of acute lymphoblastic leukemia. PSO is used to identify characteristics that discriminate between healthy and blast cells in (Srisukkham, Zhang, Neoh, Todryk, & Lim, 2017). It applies two PSO methods with different characteristics for better results. Around 180 microscopic images and cross-domain sonar dataset from the UCI Machine Learning Repository are evaluated.

11.6.11 Oil Recovery

Authors in (X. Wang & Qiu, 2013) deal with oil recovery problem from a large heavy oil reservoir. A comparison of three different versions of PSO is presented. Even though many parameters can affect the optimization problem, five parameters selected in this case are injecting fluid temperature, stream and additive gas injection, rate of injection wells, amount of CO₂ in additive gas, and liquid production rate of production wells.

11.6.12 Microscope Autofocusing

A method to autofocus the microscope for keeping micro-objects within the field of the lens is studied in (Bahadur & Mills, 2013). The image should have sharp edges, and the features of the micro-objects should be included in it. The performance of PSO is validated through image variance for sharpness quotient in the presence of noise. Micro-beads are used for experimentation.

A summary of the above applications is presented in Table 11.1.

Table 11.1 Summary of PSO applications

Sr. No.	Topic	References
<i>Electrical engg</i>		
1	Reactive power optimization	Badar et al. (2012)
2	Load forecasting	Huang et al. (2005)
3	Maximum power point tracking in photovoltaic system	Ishaque et al. (2012)
4	PID controller	Solihin et al. (2011)
5	PMU placement	Ahmadi et al. (2011)
6	Economic dispatch	Al Bahrani and Patra (2017)
7	Home energy management systems	Pedrasa et al. (2010)
8	State estimation	Tungadio et al. (2015)
9	Congestion management	Kamaraj (2011)
10	Induction cooker design	Hosseini et al. (2006)
<i>Electronics engg</i>		
1	Swarm robots	Hereford (2006)
2	Antenna design	Robinson and Rahmat-Samii (2004)
3	Channel allocation	Scott-Hayward and Garcia-Palacios (2014)
4	Filter design	Krusienski and Jenkins (2004)
5	VLSI routing	Khan et al. (2013)
6	Wireless sensor network	Yang et al. (2013)
<i>Mechanical engg</i>		
1	Machinery fault detection	Samanta and Nataraj (2009)
2	Cell formation	Anvari et al. (2010)
3	Injection molding	Bensingh et al. (2019)
4	Milling	Pare et al. (2011)
5	Multi-hole extrusion	Chen et al. (2013)
6	Nuclear power plant component design	Liu et al. (2014)
<i>Computer engg</i>		
1	Rule mining	Kuo et al. (2011)
2	Sentiment analysis	Gupta et al. (2015)
3	Data clustering	Armano and Farmani (2016)
4	Cloud computing	Awad et al. (2015)
5	Fuzzy cognitive maps	Petalas et al. (2009)
6	Video-based face recognition	Connolly et al. (2012)
<i>Other</i>		
1	Traffic flow forecasting	Hu et al. (2016)
2	Oil well location and type	Onwunalu and Durlofsky (2010)
3	Maintenance and inventory management	Samal and Pratihar (2015), Hochmuth et al. (2011)
4	Vehicle routing problem	Ai and Kachitvichyanukul (2009)
5	Submission decision process	Adewumi and Popoola (2018)
6	Ship design	Yao and Han (2013)
7	Propylene reactor application	Prata et al. (2010)

(continued)

Table 11.1 (continued)

Sr. No.	Topic	References
8	Fault selection in chemical process	Wang and Yu (2005)
9	Vapor-liquid equilibrium	Zhang et al. (2011)
10	Leukemia detection	Srisukkham et al. (2017)
11	Oil recovery	Wang and Qiu (2013)
12	Microscope autofocusing	Bahadur and Mills (2013)

As a last part of the chapter, a collection of review papers dealing with the applications of PSO is listed below Table 11.2.

11.7 Conclusion

Optimization is an ongoing process, whether it is evolution (long term) or any other small process. There are various methods from optimizing a process which may vary from simple observation to very complex mathematical processes.

For problems where large numbers of dimensions are involved along with a complex search space, it becomes impossible for finding an optimal solution through simple classical methods. In such cases, evolutionary algorithms like PSO are utilized. PSO is a robust, dynamic, and adaptive evolutionary optimization technique. The method has been in the research domain for around two and a half decades. It has been applied for optimizing a wide variety of problems. It is one of the most researched optimization techniques. PSO imitates the behavior of flocking animals/human brain to search to optimal results.

PSO is a multidimensional optimization technique which can handle different kinds of problems from various fields. PSO is also capable of implementing different varieties of optimization problems like multi-objective problems, discrete and continuous problems, complex problems, etc.

The operation of PSO is simple and is dependent on two equations: velocity calculation and updating of particle position. PSO has been implemented in a wide variety of fields to solve optimization problems.

This chapter covers PSO applications in the fields of electrical engineering, electronics engineering, mechanical engineering, computer engineering, and a combination of other topics. A large number of references are also cited related to the applications of PSO.

Table 11.2 List of review papers

Sr. No.	Title	Author	Year	References
1	An analysis of publications on particle swarm optimization applications	R. Poli	2008	Poli (2008)
2	Applications of particle swarm optimization in geotechnical engineering: a comprehensive review	M. Hajihassani, D. Jahed Armaghani, R. Kalatehjari	2017	Hajihassani, Armaghani, and Kalatehjari (2018)
3	A comprehensive survey on particle swarm optimization algorithm and its applications	Yudong Zhang, Shuihua Wang, and Genlin Ji	2015	Zhang, Wang, and Ji (2015)
4	Particle swarm optimization: basic concepts, variants and applications in power systems	Yamille del Valle et al.	2008	Del Valle, Venayagamoorthy, Mohagheghi, Hernandez, and Harley (2008)
5	Particle swarm optimization: developments, applications and resources	R.C. Eberhart, Yuhui Shi	2001	Shi and Eberhart (2001)
6	Particle swarm optimization and differential evolution algorithms: technical analysis, applications and hybridization perspectives	Swagatam Das, Ajith Abraham, and Amit Konar	2008	Das, Abraham, and Konar (2008)
7	Particle swarm optimization and intelligence: advances and applications	Konstantinos E. Parsopoulos, Michael N. Vrahatis	2010	Parsopoulos and Vrahatis (2010)
8	Review on applications of particle swarm optimization in solar energy systems	A. H. Elsheikh, M. Abd Elaziz	2018	Elsheikh and Elaziz (2019)
9	Review on the cost optimization of microgrids via particle swarm optimization	Sengthavy Phommixay et al.	2019	Phommixay, Doumbia, and St-Pierre (2020)
10	A review on particle swarm optimization algorithms and their applications to data clustering	Sandeep Rana, Sanjay Jasola, Rajesh Kumar	2010	Rana, Jasola, and Kumar (2011)
11	A review on particle swarm optimization algorithm and its variants to clustering high-dimensional data	Ahmed A. A. Esmin, Rodrigo A. Coelho, Stan Matwin	2015	Esmin, Coelho, and Matwin (2015)
12	A review of particle swarm optimization. Part II: hybridisation, combinatorial, multicriteria and constrained optimization, and indicative applications	Alec Banks, Jonathan Vincent, Chukwudi Anyakoha	2007	Banks, Vincent, and Anyakoha (2008)
13	A survey of particle swarm optimization applications in electric power systems	M. R. AlRashidi, M. E. El-Hawary	2009	AlRashidi and El-Hawary (2008)

(continued)

Table 11.2 (continued)

Sr. No.	Title	Author	Year	References
14	A review of real-world applications of particle swarm optimization algorithm	Michal Pluhacek et al.	2017	Pluhacek, Senkerik, Viktorin, Kadavy, and Zelinka (2017)
15	Particle swarm optimization applications to mechanical engineering-A review	Ninad Kulkarni et al.	2015	Kulkarni et al. (2015)

References

- Adewumi, A. O., & Popoola, P. A. (2018). A multi-objective particle swarm optimization for the submission decision process. *International Journal of Systems Assurance Engineering and Management*, 9(1), 98–110.
- Ahmadi, A., Alinejad-Beromi, Y., & Moradi, M. (2011). Optimal PMU placement for power system observability using binary particle swarm optimization and considering measurement redundancy. *Expert Systems with Applications*, 38(6), 7263–7269.
- Ai, T. J., & Kachitvichyanukul, V. (2009). A particle swarm optimization for the vehicle routing problem with simultaneous pickup and delivery. *Computers & Operations Research*, 36(5), 1693–1702.
- Al Bahrani, L. T., & Patra, J. C. (2017). Orthogonal pso algorithm for economic dispatch of thermal generating units under various power constraints in smart power grid. *Applied Soft Computing*, 58, 401–426.
- AlRashidi, M. R., & El-Hawary, M. E. (2008). A survey of particle swarm optimization applications in electric power systems. *IEEE Transactions on Evolutionary Computation*, 13(4), 913–918.
- Anvari, M., Mehrabad, M. S., & Barzinpour, F. (2010). Machine–part cell formation using a hybrid particle swarm optimization. *The International Journal of Advanced Manufacturing Technology*, 47(5-8), 745–754.
- Armano, G., & Farmani, M. R. (2016). Multiobjective clustering analysis using particle swarm optimization. *Expert Systems with Applications*, 55, 184–193.
- Awad, A., El-Hefnawy, N., & Abdel Kader, H. (2015). Enhanced particle swarm optimization for task scheduling in cloud computing environments. *Procedia Computer Science*, 65, 920–929.
- Badar, A. Q., Umre, B., & Junghare, A. (2012). Reactive power control using dynamic particle swarm optimization for real power loss minimization. *International Journal of Electrical Power & Energy Systems*, 41(1), 133–136.
- Bahadur, I. M., & Mills, J. K. (2013). Robust autofocusing in microscopy using particle swarm optimization. In *2013 IEEE International Conference on Mechatronics and Automation* (pp. 213–218). New York: IEEE.
- Banks, A., Vincent, J., & Anyakoha, C. (2008). A review of particle swarm optimization. Part II: Hybridisation, combinatorial, multicriteria and constrained optimization, and indicative applications. *Natural Computing*, 7(1), 109–124.
- Bensingh, R. J., Machavaram, R., Boopathy, S. R., & Jebaraj, C. (2019). Injection molding process optimization of a bi-aspheric lens using hybrid artificial neural networks (anns) and particle swarm optimization (pso). *Measurement*, 134, 359–374.
- Chen, W.-J., Su, W.-C., Nian, F.-L., Lin, J.-R., & Chen, D.-C. (2013). Application of anova and taguchi-based mutation particle swarm algorithm for parameters design of multi-hole extrusion process. *Research Journal of Applied Sciences, Engineering and Technology*, 6(13), 2316–2325.

- Connolly, J. F., Granger, E., & Sabourin, R. (2012). Evolution of heterogeneous ensembles through dynamic particle swarm optimization for video-based face recognition. *Pattern Recognition*, 45(7), 2460–2477.
- Das, S., Abraham, A., & Konar, A. (2008). Particle swarm optimization and differential evolution algorithms: technical analysis, applications and hybridization perspectives. In *Advances of computational intelligence in industrial systems* (pp. 1–38). Berlin: Springer.
- Del Valle, Y., Venayagamoorthy, G. K., Mohagheghi, S., Hernandez, J.-C., & Harley, R. G. (2008). Particle swarm optimization: basic concepts, variants and applications in power systems. *IEEE Transactions on Evolutionary Computation*, 12(2), 171–195.
- Elsheikh, A., & Elaziz, M. A. (2019). Review on applications of particle swarm optimization in solar energy systems. *International journal of Environmental Science and Technology*, 16(2), 1159–1170.
- Esmir, A. A., Coelho, R. A., & Matwin, S. (2015). A review on particle swarm optimization algorithm and its variants to clustering high-dimensional data. *Artificial Intelligence Review*, 44(1), 23–45.
- Gupta, D. K., Reddy, K. S., Ekbal, A., et al. (2015). Pso-asent: Feature selection using particle swarm optimization for aspect based sentiment analysis. In *International conference on applications of natural language to information systems* (pp. 220–233). Saarbrücken, Germany: NLDB.
- Hajihassani, M., Armaghani, D. J., & Kalatehjari, R. (2018). Applications of particle swarm optimization in geotechnical engineering: a comprehensive review. *Geotechnical and Geological Engineering*, 36(2), 705–722.
- Hereford, J. M. (2006). A distributed particle swarm optimization algorithm for swarm robotic applications. In *2006 IEEE International conference on evolutionary computation* (pp. 1678–1685). New York: IEEE.
- Hochmuth, C. A., Lassig, J., & Thiem, S. (2011). Optimizing complex multilocation inventory models using particle swarm optimization. In *Computational optimization, methods and algorithms* (pp. 101–124). Berlin: Springer.
- Hosseini, S. H., Kashtiban, A. M., & Alizadeh, G. (2006). Particle swarm optimization and finite-element based approach for induction heating cooker design. In *2006 SICE-ICASE International joint conference* (pp. 4624–4627). Busan: IEEE.
- Hu, W., Yan, L., Liu, K., & Wang, H. (2016). A short-term traffic flow forecasting method based on the hybrid pso-svr. *Neural Processing Letters*, 43(1), 155–172.
- Huang, C.-M., Huang, C.-J., & Wang, M.-L. (2005). A particle swarm optimization to identifying the armax model for short-term load forecasting. *IEEE Transactions on Power Systems*, 20(2), 1126–1133.
- Ishaque, K., Salam, Z., Amjad, M., & Mekhilef, S. (2012). An improved particle swarm optimization (pso)-based mppt for pv with reduced steady-state oscillation. *IEEE Transactions on Power Electronics*, 27(8), 3627–3638.
- Kamaraj, N. (2011). Transmission congestion management using particle swarm optimization. *Journal of Electrical Systems*, 7(1), 54–70.
- Khan, A., Laha, S., & Sarkar, S. K. (2013). A novel particle swarm optimization approach for VLSI routing. In *2013 3rd IEEE international advance computing conference (IACC)* (pp. 258–262). Ghaziabad: IEEE.
- Krusienski, D. J., & Jenkins, W. K. (2004). Particle swarm optimization for adaptive IIR filter structures. In *Proceedings of the 2004 congress on evolutionary computation (IEEE cat. no. 04th8753)* (Vol. 1, pp. 965–970). New York: IEEE.
- Kulkarni, M. N. K., Patekar, M. S., Bhoskar, M. T., Kulkarni, M. O., Kakandikar, G., & Nandedkar, V. (2015). Particle swarm optimization applications to mechanical engineering-a review. *Materials Today: Proceedings*, 2(4-5), 2631–2639.
- Kuo, R. J., Chao, C. M., & Chiu, Y. (2011). Application of particle swarm optimization to association rule mining. *Applied Soft Computing*, 11(1), 326–336.

- Liu, C., Yan, C., & Wang, J. (2014). Hybrid particle swarm optimization algorithm and its application in nuclear engineering. *Annals of Nuclear Energy*, 64, 276–286.
- Onwunalu, J. E., & Durlofsky, L. J. (2010). Application of a particle swarm optimization algorithm for determining optimum well location and type. *Computational Geosciences*, 14(1), 183–198.
- Pare, V., Agnihotri, G., & Krishna, C. (2011). Optimization of cutting conditions in end milling process with the approach of particle swarm optimization. *International Journal of Mechanical and Industrial Engineering*, 1(2), 21–25.
- Parsopoulos, K. E., & Vrahatis, M. N. (2010). *Particle swarm optimization and intelligence: advances and applications*.
- Pedrasa, M. A. A., Spooner, T. D., & MacGill, I. F. (2010). Coordinated scheduling of residential distributed energy resources to optimize smart home energy services. *IEEE Transactions on Smart Grid*, 1(2), 134–143.
- Petalas, Y. G., Parsopoulos, K. E., & Vrahatis, M. N. (2009). Improving fuzzy cognitive maps learning through memetic particle swarm optimization. *Soft Computing*, 13(1), 77.
- Phommixay, S., Doumbia, M. L., & St-Pierre, D. L. (2020). Review on the cost optimization of microgrids via particle swarm optimization. *International Journal of Energy and Environmental Engineering*, 11(1), 73–89.
- Pluhacek, M., Senkerik, R., Viktorin, A., Kadavy, T., & Zelinka, I. (2017). A review of real-world applications of particle swarm optimization algorithm. In *International conference on advanced engineering theory and applications* (pp. 115–122). Cham: Springer.
- Poli, R. (2008). Analysis of the publications on the applications of particle swarm optimisation. *Journal of Artificial Evolution and Applications*, 2008, 685175.
- Prata, D. M., Schwaab, M., Lima, E. L., & Pinto, J. C. (2010). Simultaneous robust data reconciliation and gross error detection through particle swarm optimization for an industrial polypropylene reactor. *Chemical Engineering Science*, 65(17), 4943–4954.
- Rana, S., Jasola, S., & Kumar, R. (2011). A review on particle swarm optimization algorithms and their applications to data clustering. *Artificial Intelligence Review*, 35(3), 211–222.
- Robinson, J., & Rahmat-Samii, Y. (2004). Particle swarm optimization in electromagnetics. *IEEE Transactions on Antennas and Propagation*, 52(2), 397–407.
- Samal, N. K., & Pratihar, D. K. (2015). Joint optimization of preventive maintenance and spare parts inventory using genetic algorithms and particle swarm optimization algorithm. *International Journal of Systems Assurance Engineering and Management*, 6(3), 248–258.
- Samanta, B., & Nataraj, C. (2009). Use of particle swarm optimization for machinery fault detection. *Engineering Applications of Artificial Intelligence*, 22(2), 308–316.
- Scott-Hayward, S., & Garcia-Palacios, E. (2014). Channel time allocation PSO for gigabit multimedia wireless networks. *IEEE Transactions on Multimedia*, 16(3), 828–836.
- Shi, Y., & Eberhart, R. (2001). Particle swarm optimization: developments, applications and resources. In *Proceedings of the 2001 congress on evolutionary computation (IEEE cat. no. 01th8546)* (Vol. 1, pp. 81–86). New York: IEEE.
- Solihin, M. I., Tack, L. F., & Kean, M. L. (2011). Tuning of PID controller using particle swarm optimization (PSO). In *Proceeding of the international conference on advanced science, engineering and information technology* (Vol. 1, pp. 458–461). New York: IEEE.
- Srisukkham, W., Zhang, L., Neoh, S. C., Todryk, S., & Lim, C. P. (2017). Intelligent leukaemia diagnosis with bare-bones PSO based feature optimization. *Applied Soft Computing*, 56, 405–419.
- Tungadio, D., Numbi, B., Siti, M., & Jimoh, A. A. (2015). Particle swarm optimization for power system state estimation. *Neurocomputing*, 148, 175–180.
- Wang, L., & Yu, J. (2005). Fault feature selection based on modified binary PSO with mutation and its application in chemical process fault diagnosis. In *International conference on natural computation* (pp. 832–840). Berlin: Springer.
- Wang, X., & Qiu, X. (2013). Application of particle swarm optimization for enhanced cyclic steam stimulation in a offshore heavy oil reservoir. *Computational Engineering, Finance, and Science*. <https://arxiv.org/abs/1306.4092>.

- Yang, J., Zhang, H., Ling, Y., Pan, C., & Sun, W. (2013). Task allocation for wireless sensor network using modified binary particle swarm optimization. *IEEE Sensors Journal*, 14(3), 882–892.
- Yao, J., & Han, D. (2013). Improved barebones particle swarm optimization with neighborhood search and its application on ship design. *Mathematical Problems in Engineering*, 2013, 175848.
- Zhang, H., Kennedy, D. D., Rangaiah, G. P., & Bonilla-Petriciolet, A. (2011). Novel bare-bones particle swarm optimization and its performance for modeling vapor–liquid equilibrium data. *Fluid Phase Equilibria*, 301(1), 33–45.
- Zhang, Y., Wang, S., & Ji, G. (2015). A comprehensive survey on particle swarm optimization algorithm and its applications. *Mathematical Problems in Engineering*, 2015, 931256.

Chapter 12

Particle Swarm Optimization in Global Path Planning for Swarm of Robots



Ritesh Kumar Halder

Abstract Planning of the optimal path of an autonomous swarm of mobile robots is quite challenging since they may need to meet multiple targets while avoiding obstacles. This chapter addresses the problem using a method of global navigation based on particle swarm optimization technique. Since it is a meta-heuristic search technique, the path can be found following any optimization criteria such as shortest distance or minimum time. The chapter explores different traditional path planning approaches in contrast with the evolutionary algorithms. The PSO algorithm is tested in different scenarios. Several modifications were implemented in the algorithm for optimization improvements and faster convergence, leading to better results. Geometrical illustrations were used to explain the changes in position of the particles with respect to the environment and obstacles. Consequently, the experiments are conducted on a simulated robot, and the visualizations demonstrated the feasibility of the technique to solve global path planning problem.

Keywords PSO · Autonomous navigation · Evolutionary algorithm · Obstacle avoidance · Artificial intelligence

12.1 Introduction

Autonomous mobile robots are devices, which use their intelligence to navigate the environment they work in. The environment can be anything such as a crowded street, interiors of a house, or rough terrains of the moon. To navigate such environments, a major challenge encountered by these robots is an optimized, collision-free path planning. Avoiding collisions with stationary or moving objects and reaching the target with minimum cost involved is a wide area of ongoing research. In path planning procedure, the robot needs to find a trajectory connecting

R. K. Halder (✉)
IIT Kanpur, Kanpur, India

the start and endpoints while minimizing the cost or the optimization criteria involved. The trajectory needs to be an optimized collision-free path, according to some optimization criteria such as path involving shortest distance or minimum time. This problem is classified into global and local path planning problem. In global path planning, all the information about the environment is provided to the planner before the operation, whereas in local path planning, the environment is unknown or partly known. Since supplying a new map every time minor changes occur in the environment is costly, local planners are used in combination with the global planner. These does not need the entire environment model and works in locally limited area. In this work, the author proposes a method of global navigation based on particle swarm optimization technique.

Traditional algorithms for global path planning were mostly deterministic in nature, such as visibility graph, artificial potential field, cell decomposition, etc. Particle swarm optimization is a meta-heuristic search technique inspired by the behavior of the flock of birds. PSO is relatively simple and converges faster, and movement is updated using population-based feature. Thus, it is a considerable alternative for solving path planning problem. PSO has been used for target search applications, such as firefighting, landmine detection, and military surveillance, and is an effective technique for swarm robotic search problems.

In the next section, the background and origin of PSO algorithm are discussed in brief along with its modifications over time and its implementation in the domain of autonomous path planning. The meta-heuristic approach with some of the examples of evolutionary algorithms and particle swarm optimization theory is described in Sect. 12.3, along with the justification for its use in path planning. In Sect. 12.4, the setup for PSO in 2D search space, and the modifications in the algorithms implemented for the experiments, is explained. This follows the experimental results obtained from the simulations and the graphical representations, followed by its implementation and visualization in Robot Operating System. Future research directions and conclusion are presented in the last two sections.

12.2 Background

Kennedy and Eberhart introduced particle swarm optimization in 1995 (Eberhart & Kennedy, 1995, November). The optimization algorithm was an effort to integrate the social behaviors found in various living species, with the tools and techniques available in modern computer graphics and ideas from social psychology research.

Initially in the domain of computer graphics, Reeves proposed the idea to model dynamic objects using particle systems (Reeves, 1983), which cannot be represented by simpler surfaces such as polygons. Objects like smoke, water, and fire are such objects, which cannot be modelled by conventional techniques, since the particles are independent of each other. However, we can predict their movement since they are guided by some set of rules (convection current, gravity, wind direction, etc.). Reynolds simulated the behavioral movement of flock of birds using a particle

system (Reynolds, 1987, August). Heppner and Grenander used simulated bird flock model including a roost and simulated birds being attracted to it, following stochastic differential equations (Heppner & Grenander, 1990). The original PSO algorithm used these models as an inspiration to create its own set of rules.

Nowak, Szamrej, and Latané proposed the dynamic theory of social impact, in the domain of social psychology research (Nowak, Szamrej, & Latané, 1990). The rules that apply on human social psychology, and lead individuals to adjust their attitudes and beliefs to be in harmony to those of their peers, are similar to the ones, which govern the particle movements in the search space. This is also an inspiration for the development of the first particle swarm optimization algorithm (Kennedy, 2006).

The algorithm was improved by including additional parameters. Shi and Eberhart introduced the inertia parameter “w,” whose value determines the characteristics of global and local search (Shi & Eberhart, 1998, May). In 2002, Laskari et al. showed that PSO with a slight modification can be used to solve discrete optimization problems, by working with continuous variables and rounding off the real optima to the nearest integer position (Laskari, Parsopoulos, & Vrahatis, 2002, May). Hassan, B. Cohanim, and O. de Weck have shown that PSO is able to accommodate multiple objectives and it’s also faster and more efficient than the GA (genetic algorithm) (Hassan, Cohanim, De Weck, & Venter, 2005, April). Sedighizadeh and Masehian enumerate and describe briefly 102 algorithms that are based on PSO (Sedighizadeh & Masehian, 2009). These authors classified the PSO algorithms according to the type of control variables: continuous, binary, and discrete. They also used different criteria to cluster these algorithms, e.g., attraction (there can be attraction, repulsion, or both), swarm topology, activity (a particle can be in passive state and have no social behavior), and sign of particle trajectories (particles can follow the worst particle instead of the best one). Other criteria were hierarchy (higher-level particles have more effect on the evolution of the swarm), restriction (particles move in a constricted space or not), etc. Hybridized with other heuristic algorithms such as simulated annealing, ant colony optimization, genetic algorithms, differential evolution, etc., optimization objective (single-objective or multi-objective optimization) was also considered. Spears et al. studied the biases in PSO and found that particles tend to concentrate along paths parallel to the coordinate axes and rotational variance is related to the concentration (Spears, Green, & Spears, 2012).

In the application of autonomous navigation, Ezequiel Di Mario, Zeynab Talebpour, and Alcherio Martinoli compared reinforcement learning and PSO for multi-robot obstacle avoidance and showed that time required by PSO as compared to Q-learning can be significantly reduced as the algorithm is distributed in nature (Di Mario, Talebpour, & Martinoli, 2013, June). Guangsheng Li and Wusheng Chou used PSO in combination with self-adaptive learning technique, which would switch over to the suitable search technique based on different stages of the optimization process, thus improving the searching ability (Li & Chou, 2018).

12.3 Meta-Heuristic Algorithms and Particle Swarm Optimization Theory

Traditionally, *heuristic* means “to discover” by trial and error. Heuristic algorithms are designed with the specific problem in mind. However, to solve it in a faster and more efficient manner than conventional stochastic strategies, often optimality, precision, or accuracy is compromised. Thus, mostly these algorithms are used when an exact solution is computationally expensive, whereas approximate solutions are sufficient. Meta-heuristics (*meta-* meaning “beyond”), on the other hand, are problem-independent techniques that can be applied to a broad range of problems. These usually perform better than simple heuristic algorithms. Although optimal solution is not guaranteed, these are suitable for global optimization problems. They know nothing about the problem they are applied to; hence, they treat functions as black boxes.

There is a tradeoff of local search and randomization in most meta-heuristic algorithms. This includes two components, exploitation and exploration. Exploration creates diverse solutions initially, spread over entire search space for better coverage. Wherever there is a higher probability of getting good solution, exploitation focuses the search in that region. To increase the rate of convergence, a good balance between these two components is required during selection of best solutions. Exploration helps in escaping local optima, whereas selecting best solutions leads to convergence at optimum. Thus, these algorithms perform better than conventional methods as well as reaches near-optimal solution better than heuristic algorithms.

12.3.1 Swarm Intelligence

Swarm intelligence systems involve multiple agents, which are capable of interacting with each other locally in the environment. This kind of collective behavior is found naturally in many species, such as birds, bees, ants, etc. Thus, swarm intelligence is used to describe all such decentralized systems, both natural and artificial. Few noteworthy algorithms belonging to this class of systems are mentioned below. Each of these is inspired by the self-organizing behavior of such species.

These algorithms are also known as “evolutionary algorithms,” since they are similar to the evolutionary mechanism of the species. Evolutionary algorithms are based on the process of evolution of species. Just as evolution of any population facing environmental changes and pressure occurs, leading to natural selection, i.e., survival of the fittest, similarly these algorithms evolve by increasing the average fitness of the population. Fitness is the performance criteria for adaptation of the organism; better fitness signifies better adaptation or higher chances of reaching the goal. Generally, these algorithms roughly focus on few mechanisms similar to biological evolutionary process (Figs. 12.1, 12.2, 12.3, and 12.4).

Fig. 12.1 Ant colony optimization algorithm.
Source: Redrawn by author
(Wikimedia, 2020)

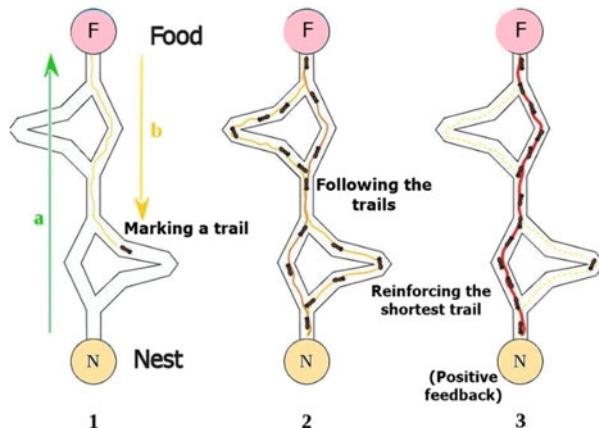
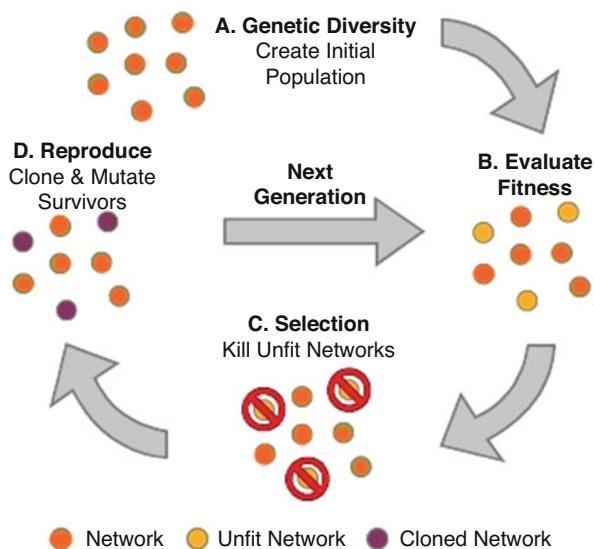


Fig. 12.2 Genetic algorithm. Source: Redrawn by author (Aparra, 2016)



Few algorithms of this class are:

- Ant colony optimization
- Particle swarm optimization
- Genetic algorithms
- Artificial bee colony
- Cuckoo search

Most commonly used algorithm among these is the particle swarm optimization (PSO). Ant colony optimization (ACO) and artificial bee colony algorithm are among other algorithms comparable to PSO, though particle swarm optimization

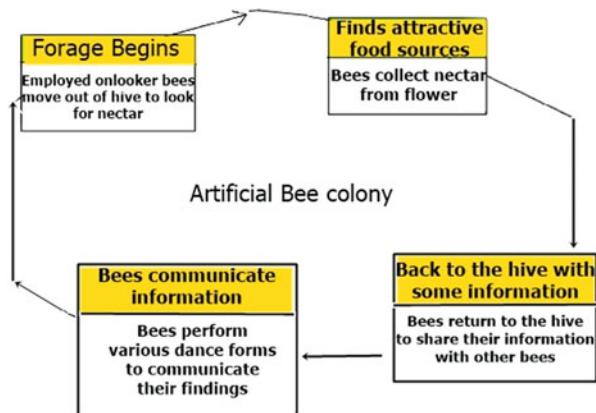


Fig. 12.3 Artificial bee colony algorithm. Source: Redrawn by author (Huang & Liu, 2013)

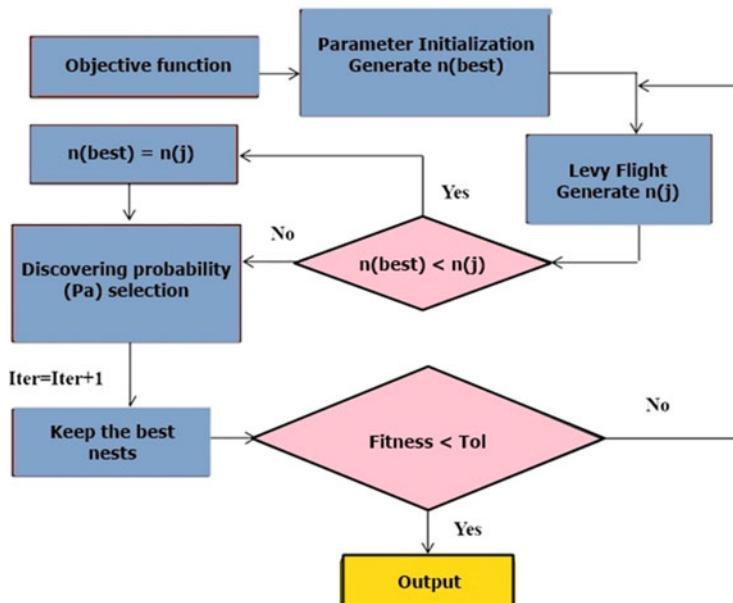


Fig. 12.4 Flowchart of cuckoo search using Lévy flight. Source: Redrawn by author (Xu, Liu, & Lu, 2016)

is preferred for its simplicity and faster convergence. These algorithms are described in brief in the following section.

12.3.1.1 Ant Colony Optimization

This heuristic optimization algorithm is a commonly used solution for the Travelling Salesman Problem (TSP). In this problem, a salesperson is given a set of locations for him to visit and the distances between them. The person needs to find the shortest and closed route visiting all the locations only once. In ACO, a set of artificial agents or ants constructs the best possible solution. The ants probabilistically choose the vertices to follow based on a pheromone update model and keep a memory of the path. Optimal solutions receive higher pheromones and thus are retained until the end until the termination criteria are satisfied (Dorigo & Di Caro, 1999, July).

12.3.1.2 Artificial Bee Colony

The artificial bee colony algorithm is based on the foraging behavior of the bee colonies. In this, three types of agents or bees are defined—employed, onlooker, and scout bees. The employed bees look for food sources and keep a memory of it. The quality of the source is determined by a fitness function (similar to that in a PSO algorithm). If the fitness value is greater than the previous best value, it is retained. The onlooker bees select the higher-quality sources. The employed bees then abandon the sources from memory and become the scout bees. They discover new food sources based on certain parameters and boundaries (Xu, Fan, & Yuan, 2013).

12.3.2 Particle Swarm Optimization

In particle swarm optimization, a fixed number of particles are initialized randomly within the search space. These particles move along the optimization space whose boundaries are usually defined. Each particle's positions are considered the solution of the optimization problem. These particles then follow the velocity and position update formulas and continuously move toward the optimal solution.

The optimization problem to solve is formulated in the form of the fitness function. Each particle solves this fitness function and tries to reach the optimal value. Each i^{th} particle at time t stores the following information—position values (p_i^t) representing the solution and the velocity values (v_i^t). Along with this, the value of the fitness function $f(p_i^t)$ for the particle, the previous local best solution ($p_{p\text{Best}}^t$), the value of the fitness function for local best $f(p_{p\text{Best}}^t)$, and also the global best solution ($p_{g\text{Best}}^t$) and fitness function value for the global best $f(p_{g\text{Best}}^t)$ are stored.

Let N be the number of particles assigned in the search space. Starting positions and velocities of the particles are initialized randomly. Following that, for $i = 1 \dots N$, the velocity and position of the particles are updated using the following formula:

$$v_i^{t+1} = w * v_i^t + c_1 \cdot r_1 \cdot (p_{p\text{Best}}^t - p_i^t) + c_2 \cdot r_2 \cdot (p_{g\text{Best}}^t - p_i^t)$$

$$p_i^{t+1} = p_i^t + v_i^{t+1}$$

The velocities are usually bounded within an interval ($-V_{\max}$, V_{\max}) to avoid extremely large or small update steps. Here r_1 and r_2 are random number multiplier, $\text{rand}() \in [0,1]$. Basic control parameters for PSO operation are discussed below:

- **Inertia weight (w):** This helps in controlling the continuing motion of the particles and balances the convergence speed for global and local search. Higher values help in the global search, while lower values help with the local search.
- **Acceleration coefficients (c_1 and c_2):** These influence the local and social components of the particles. Constant c_1 represents the amount of confidence the particle has on itself, whereas c_2 represents the confidence it has on its neighboring particles. The inclusion of these coefficients gives each particle its own adaptive acceleration based on the value of the fitness function at the current particle position.
- **Swarm size/number of particles:** This is usually problem dependent and needs to be chosen keeping in mind the complexity of the problem. Large number of particles will cover search space easily, but it will also increase time complexity for each iteration. However, a smoother search space will require fewer particles to achieve an optimized solution easily.
- **Number of iterations:** This is also problem dependent. It is usually determined by other stopping conditions such as obtaining a solution within a threshold limit or if the solution saturates and does not improve with further iterations. Large number of iterations may also increase computational complexity (Fig. 12.5).

12.3.2.1 Pseudo Code for Particle Swarm Optimization

- For each i^{th} particle:
 - Initialize position p_i^t randomly within the search space
 - Initialize velocity v_i^t randomly within the search space
- End For
- Iteration $k = 1$:
- Do:
 - For each i^{th} particle:
 - Calculate fitness value $f(p_i^t)$
 - If fitness value is better than local fitness value; $f(p_i^t) > f(p_{p\text{Best}}^t)$

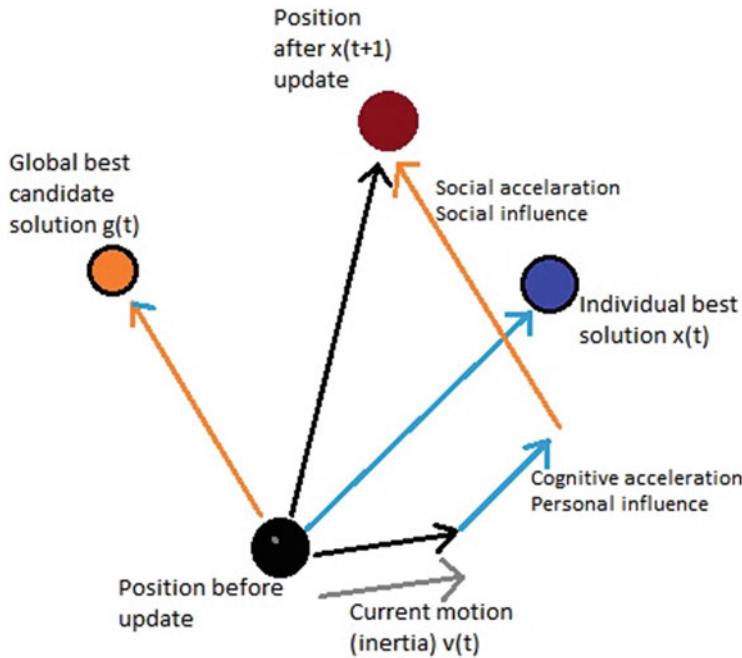


Fig. 12.5 Updated particle position represented as a sum of vectors. Source: Redrawn by author (Macedo, 2018)

$$\text{Set } p_{p\text{Best}}^t = p_i^t$$

- End For
- If the local best fitness value is better than global fitness value; $f(p_{p\text{Best}}^t) > f(p_{g\text{Best}}^t)$:

$$\text{Set } p_{g\text{Best}}^t = p_{p\text{Best}}^t$$

- For each i^{th} particle:

Calculate particle velocity according to the equation:

$$v_i^{t+1} = w * v_i^t + c_1 \cdot r_1 \cdot (p_{p\text{Best}}^t - p_i^t) + c_2 \cdot r_2 \cdot (p_{g\text{Best}}^t - p_i^t)$$

Update particle position according to the equation:

$$p_i^{t+1} = p_i^t + v_i^{t+1}$$

– End For

$$k = k + 1$$

- While maximum iterations or minimum error criteria is not attained.

12.3.3 Particle Swarm Optimization in Path Planning

Classical path planning algorithms are deterministic in nature. Either a solution exists or it does not. Thus, these algorithms are unable to cope with uncertainty and are computationally intensive. For the problem statement at hand, few of the classical algorithms were applied to check for efficiency. A major difference was observed when the planning algorithm needed to achieve multiple targets in an efficient and optimized way, where the meta-heuristic or evolutionary algorithms fared better. Few classical path planning algorithms previously explored were:

- Dijkstra's algorithm.
- A-star, D-star.
- Probability roadmaps.
- Rapidly exploring random trees (RRT) and RRT-star (Fig. 12.6).

All these algorithms are deterministic in nature, suited for navigation of a single bot scenario, where the sensory information is used to determine the costmap and find out the path.

12.4 Solutions and Recommendations

To overcome the problems faced by classical deterministic algorithms, the evolutionary algorithm particle swarm optimization is applied to the problem at hand. Here the implementation of a basic PSO algorithm in a 2D search space is constrained in many aspects. The following assumptions are taken into consideration:

- The object to traverse is a point object (particle) with no dimensions.
- The movement of the object is holonomic (capable of movement in any direction).
- Obstacles are circular in shape, and their center and radius values are known.

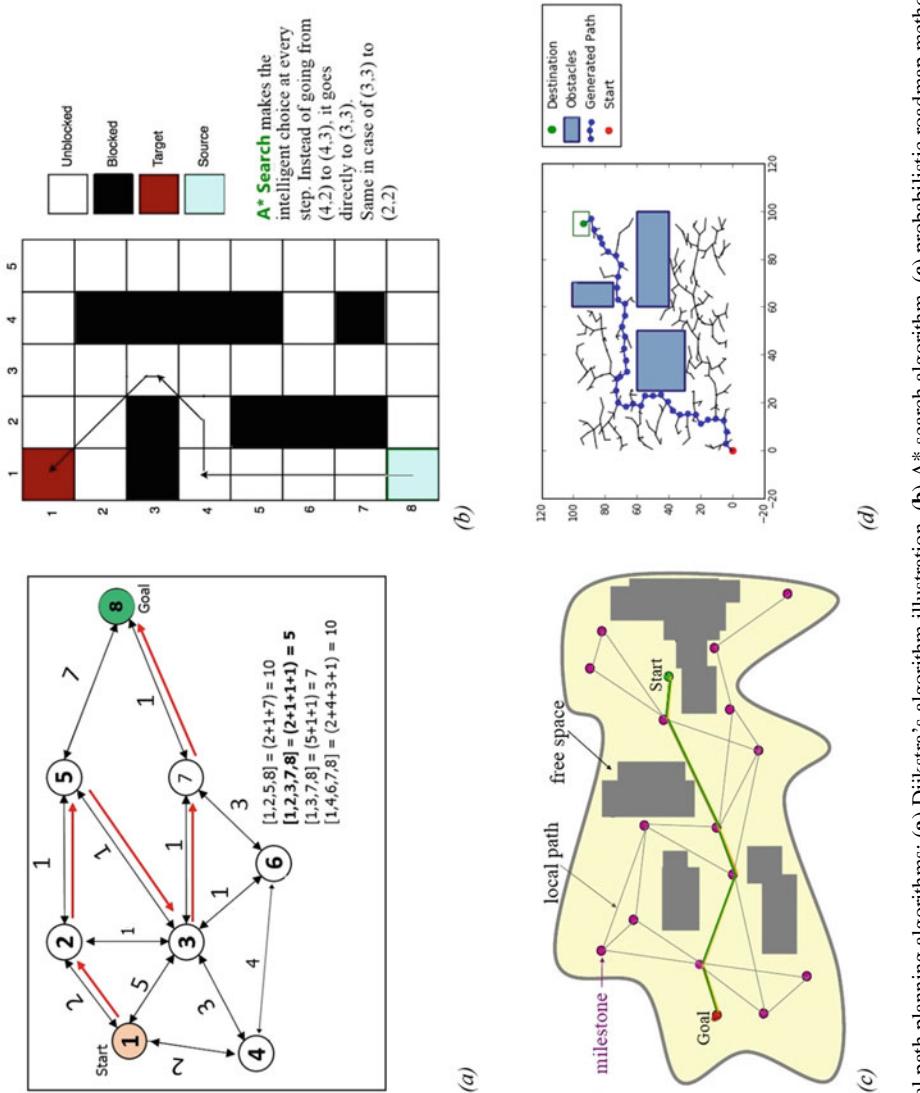


Fig. 12.6 Classical path planning algorithms: (a) Dijkstra's algorithm illustration, (b) A* search algorithm, (c) probabilistic roadmap method (PRM), and (d) rapidly exploring random tree. Source: Redrawn by author (Belwarier, 2018; LaValle, 1998; Rehman, Awuah-Offei, Baker, & Bristow, 2019)

In real-world scenario, these assumptions may not hold true as every object has certain dimension and degrees of freedom that may need consideration. For simplicity, the ideal case conditions are taken into consideration.

12.4.1 PSO Algorithm Implementation in 2D Search Space

The implementation was done in a 2D search space to verify the working PSO algorithm. Since this is a holonomic case, the dependency on orientation was nullified. The path generation from a given set of start and goal position was observed in the presence of various circular obstacles in the search area. The start and end cells were user given variables. The user can also choose the number of obstacles and their radius. The algorithm generates a set of obstacles randomly placed in the entire search space. Certain coefficients such as inertial weight coefficient and individual and social coefficients were chosen via trial and error method to determine the possible set of parameters which gives the best results. The entire program was prepared in C++ and the data gathered was stored in a spreadsheet. This was used to plot the data using python's plotting tools, to visualize the particles movement and the path generated.

Since the objective in path planning situation is to get the shortest path between the start and goal, the distance between two points was taken as the fitness function, with the objective to minimize this function.

Nomenclature:

$p_i^t = (x_i^t, y_i^t)$ is the i^{th} particle position coordinates at time t .

$p_{gb}^t = (x_{gb}^t, y_{gb}^t)$ is the current global best particle position.

v_i^t = i^{th} particle velocity at time t .

v_{gb}^t = velocity of the global best particle.

12.4.2 Basic Fitness Function

The objective is to traverse the vehicle via the shortest path. Thus, the shortest distance or the Euclidean distance between any two points in the search space is taken as the fitness function. Formula for the distance function is given by:

$$F_{1i} = \sqrt{(x_i^t - x_{\text{goal}})^2 + (y_i^t - y_{\text{goal}})^2} \quad (12.1)$$

where x_i^t, y_i^t are the present coordinates of the particle and $x_{\text{goal}}, y_{\text{goal}}$ are the coordinates of the goal. Further modifications were done to include the angle between the goal and the two successive particle position, as discussed in the next section.

12.4.3 Modifications and Improvements to the Basic Algorithm

To improve the performance of the basic algorithm, certain modifications are suggested by various research works. The improvements in the path planning algorithm implemented are as follows:

- Applying a slight perturbation to the global best particles so that it is not stuck in local minima (Adamu, Jegede, Okagbue, & Oguntunde, 2018). This helps in convergence faster by exploring options near the local minima or maxima. The particles are updated according to the given set of operations:

Start operation for a fixed number of iterations:

$$v_{gb}^{t+1} = (\alpha \cdot v_{gb}^t) + (\beta \cdot r_3) \quad (12.2)$$

$$p_{gb}^{t+1} = p_{gb}^t + v_{gb}^{t+1} \quad (12.3)$$

Set $i = i + 1$.

Terminate on convergence or when the iteration limit is reached.

where:

v_{gb}^{t+1} is the velocity of the next global best particle and p_{gb}^t and v_{gb}^t are the position and velocity of the current global best particle, respectively.

$\alpha = 0.1$, $\beta = 0.2$ are arbitrarily taken constants and r_3 is a random variable.

- Inertial weight is an important factor responsible for the convergence of the algorithm. Thus, instead of using a fixed inertial weight value by trial and error, non-linear decreasing inertial weight (NDIW) is taken into consideration (Gao, 2018). The formulation takes in boundary conditions of weight values (maximum and minimum weight value allowed) and changes weight at each iteration. The formula takes higher weight value in the initial iterations for faster convergence and lower value near the end to prevent overshoot and improves local optimization ability. The formulation is given as follows:

$$w_i = w_{\min} + (w_{\max} - w_{\min}) * e^{-(2.3*i/T_{\max})^2} \quad (12.4)$$

where w_{\max} and w_{\min} are the upper and lower boundaries for the inertial weight, respectively, usually taken in the range of (0.4, 0.9). i is the present iteration, and T_{\max} represents maximum number iterations.

- Basic shortest distance fitness function makes erratic orientation changes and may cause problems in practical cases. Thus to generate a smooth path, additional fitness function is used taking the angle between two hypothetical lines connecting the goal point to the vehicle's two successive positions in each iteration, i.e., $gbest(t)$ and $gbest(t - 1)$, into consideration (Masehian & Sedighizadeh, 2010, March). The formulation is given as follows:

$$F_{2i} = \cos^{-1} \frac{(x_i^t - x_{\text{goal}}) \cdot (x_{gb}^t - x_{\text{goal}}) + (y_i^t - y_{\text{goal}}) \cdot (y_{gb}^t - y_{\text{goal}})}{\sqrt{(x_i^t - x_{\text{goal}})^2 + (y_i^t - y_{\text{goal}})^2} * \sqrt{(x_{gb}^t - x_{\text{goal}})^2 + (y_{gb}^t - y_{\text{goal}})^2}}$$
(12.5)

To combine the two fitness functions, weighted sums of both the fitness values are considered.

Let the weights be $\alpha = 1, \beta = 0.25$.

Thus, total fitness is defined by the formula:

$$F_i = \alpha * F_{1i} + \beta * F_{2i} \quad (12.6)$$

12.4.3.1 Obstacle Avoidance

For obstacle avoidance, relocation method is used instead of penalty method. In penalty method, we remove the particles located inside an obstacle, or the path connecting new and old point passing through the obstacle is removed. However, in relocation method, particles' positions are relocated if they are located within the obstacle. In this way, no particle and in turn no information are lost (Islam, Tajmiruzzaman, Muftee, & Hossain, 2014).

The algorithm for relocation of particles' positions is:

```

for i=1 to M, M is no. of obstacles do
  for j=1 to N, N is no. of particles do
    Find I, I is matrix of intersection points of
    obstacle and line connecting the new position
    of the particle to the robot's current position.
    while ( I ~= empty) do
      Find new position of the particle
      for k=1 to M do
        Find I
        if(I==empty)
          Break
        end if
      end for k
    end while
  end for j
end for i

```

12.4.4 Results and Discussion

2D square search space of fixed height and width was taken for the experiment. A wide range of parameters were used on a trial and error basis to determine the set of parameters giving the best results in terms of faster convergence, efficient obstacle avoidance, feasible path generation, avoiding local minima, preventing high overshoots, etc. The console printed values for global best position of particle and fitness are displayed at each iteration for different cases. The final parameters that balance a trade-off between all these criteria are described here.

12.4.4.1 Metrics Used

The following key metrics are used to measure the input parameters and the outcomes of the simulation:

12.4.4.2 Simulation Parameters

Input Metrics

- Search space dimensions: height, 3000; width, 3000, unit inches.
- Starting coordinates: (1000, 1000).
- Goal coordinates: (-500, -1000).
- Number of obstacles: 10.
- Swarm size: 100.
- Number of max iterations: 200.
- Inertial weight coefficients: w_{\min} , 0.2; w_{\max} , 0.7.
- Local and social coefficients: c_1 , 2; c_2 , 2.
- Max allowed velocity: width/10 inches
- Distance threshold to reach near goal for success criteria: within 10% of shortest distance.
- Distance threshold to avoid near obstacles (safe distance): radius of obstacle * 0.1 inches.

Output Metrics

- Global best fitness value at each iteration.
- Global best position at each iteration.
- Final global best position (within the threshold to reach near goal).

12.4.4.3 Simulation Results

The experiment is executed using different parameters to check for convergence. Finally using the above parameters along with the modifications implemented to the basic algorithm has resulted in faster convergence. Four such cases/executions are shown as an example here. The console printed values of the output metrics are displayed.

Points to Note

- In case (a) and case (d), it took 11 and 5 iterations, respectively, to reach within 10% of the distance to the goal. At iterations 7 and 10 in case (a) and iteration 3 in case (d), the algorithm is stuck at local minima. The global best particle is then slightly perturbed according to the modification in Eqs. (12.2) and (3), which forces the particle to get out of the local minima and converge faster. This can be observed clearly since after the perturbation, the algorithm did not take much iterations to converge.
- In case (b) and case (c), the starting point or target points were located inside obstacles. This happened since the obstacles are randomly placed for the convenience of simulation. There will not be any such case in a practical environment. The obstacle avoidance module checks for these anomalies and generates these messages and changes the obstacle positions accordingly (Fig. 12.7).

12.4.4.4 Graphical Representations

The following graphs were plotted using python with the particle position data gathered from the above experiment.

- Initial point: Point A at top right.
- Final point: Point B at bottom left.
- Obstacles: The red circles represent the obstacles in the area under consideration.
- Final path: The blue line represents the final path from initial to final point.
- Color-coding of particles: The colors of the particles are based on the iterations. With each iteration, the particles change color from blue (initial particle distribution) to red (final particle distribution).

Points to Note

- As per the algorithm, the particles are restricted from being placed inside the boundaries of the obstacles on further iteration using the obstacle avoidance module.
- Initial particles (blue in color) are randomly placed, thus are more scattered. The final particles (red in color) converge near the goal point.
- The algorithm finds out the least cost path even when the obstacles are very closely placed, thus proving efficient in such scenarios (Fig. 12.8).

```

Iteration :1.....
New Global Best fitness: 983.131
New Global Best Position: (232.027,-344.382)
Iteration :2.....
New Global Best fitness: 983.131
New Global Best Position: (232.027,-344.382)
Iteration :3.....
New Global Best fitness: 983.131
New Global Best Position: (232.027,-344.382)
Iteration :4.....
New Global Best fitness: 975.864
New Global Best Position: (189.749,-309.879)
Iteration :5.....
New Global Best fitness: 960.622
New Global Best Position: (116.245,-263.191)
Iteration :6.....
New Global Best fitness: 957.519
New Global Best Position: (179.034,-325.268)
Iteration :7.....
New Global Best fitness: 942.717
New Global Best Position: (100.002,-273.041)
Local Minima detected:
Iteration :8.....
New Global Best fitness: 639.705
New Global Best Position: (-101.486,-500.247)
Local Minima detected:
Iteration :9.....
New Global Best fitness: 364.861
New Global Best Position: (-284.023,-706.114)
Local Minima detected:
Iteration :10.....
New Global Best fitness: 187.207
New Global Best Position: (-403.306,-840.631)
Local Minima detected:
Iteration :11.....
New Global Best fitness: 48.2397
New Global Best Position: (-546.614,-1002.29)

Reached near goal. SUCCESS
Final Global Best Position: (-546.614,-1002.29)
Values written to csv file
Run Successful

```

(a)

```

Iteration :1.....
New Global Best fitness: 1339.96
New Global Best Position: (833.549,-872.195)
Iteration :2.....
New Global Best fitness: 627.126
New Global Best Position: (-1124.87,-989.451)
Iteration :3.....
New Global Best fitness: 471.484
New Global Best Position: (-967.636,-936.053)
Iteration :4.....
New Global Best fitness: 192.296
New Global Best Position: (-634.156,-863.526)
Iteration :5.....
New Global Best fitness: 178.587
New Global Best Position: (-518.146,-823.556)
Iteration :6.....
New Global Best fitness: 42.3198
New Global Best Position: (-505.271,-958.282)

Reached near goal. SUCCESS

```

```

Final Global Best Position: (-505.271,-958.282)
Values written to csv file
Run Successful

```

(b)

```

start or target in obstacle.....
Iteration :1.....
New Global Best fitness: 441.27
New Global Best Position: (-473.635,-1438.98)
Iteration :2.....
New Global Best fitness: 194.075
New Global Best Position: (-655.696,-887.296)
Iteration :3.....
New Global Best fitness: 55.4197
New Global Best Position: (-551.416,-1019.88)

Reached near goal. SUCCESS
Final Global Best Position: (-551.416,-1019.88)
Values written to csv file
Run Successful

```

(c)

```

start or target in obstacle.....
Iteration :1.....
New Global Best fitness: 264.619
New Global Best Position: (-292.003,-1161.94)
Iteration :2.....
New Global Best fitness: 247.249
New Global Best Position: (-284.132,-1120.06)
Iteration :3.....
New Global Best fitness: 223.547
New Global Best Position: (-322.239,-1135.42)
Local Minima detected:
Iteration :4.....
New Global Best fitness: 164.529
New Global Best Position: (-436.869,-1151.81)
Iteration :5.....
New Global Best fitness: 78.3348
New Global Best Position: (-423.286,-1006.34)

Reached near goal. SUCCESS
Final Global Best Position: (-423.286,-1006.34)
Values written to csv file
Run Successful

```

(d)

Fig. 12.7 Output log messages of four different cases displayed during execution of the experiment, showing the values of global best-fit position and its fitness value

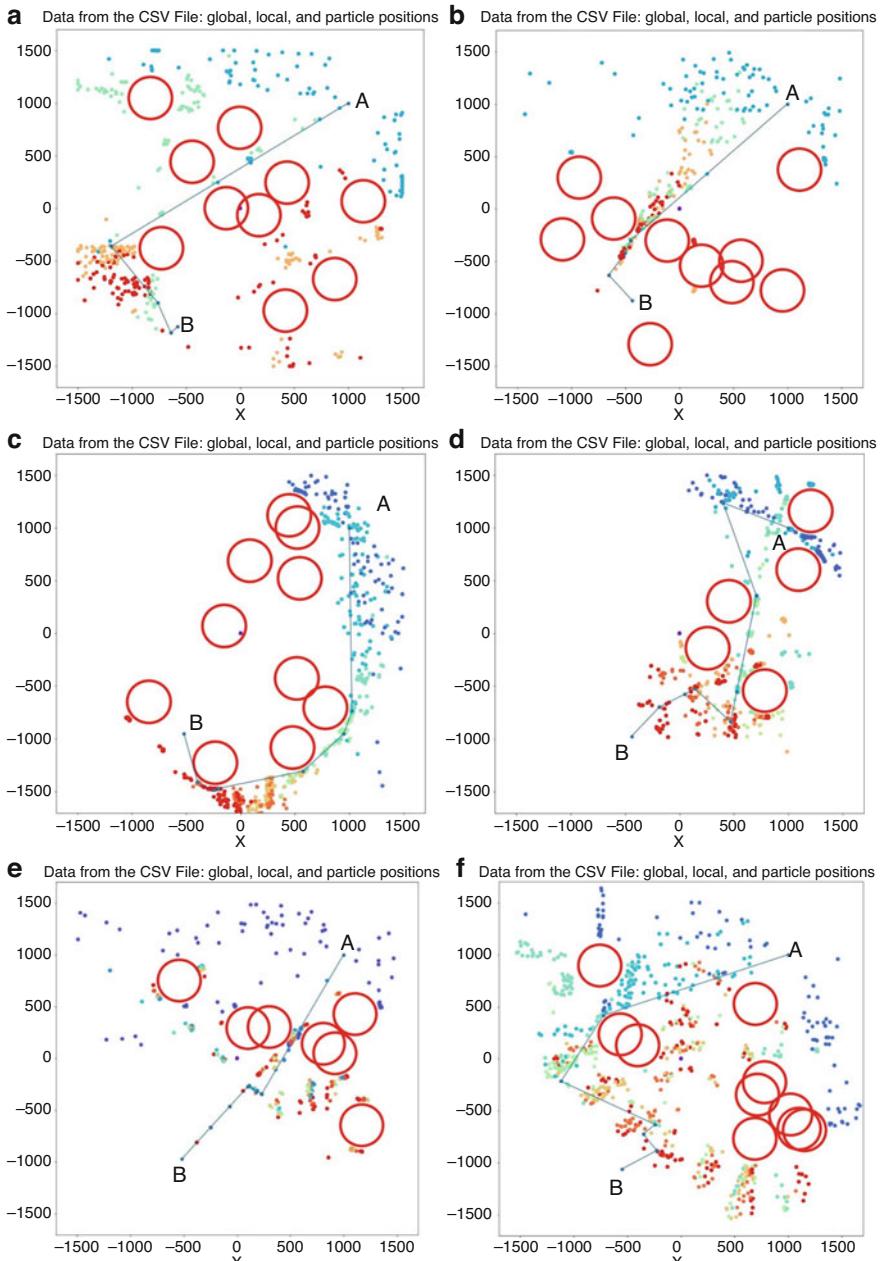


Fig. 12.8 Graphical representation of the final path connecting the start and goal point for six different and complex scenarios. Randomly placed obstacles are included to simulate a real-world environment. Particles are displayed with color-coding to demonstrate the convergence of the algorithm. Source: Author's own creation

12.4.5 Implementation in ROS (*Robot Operating System*)

The Robot Operating System (ROS) is a flexible framework for writing robot software. ROS uses RViz (Robot Vision) framework for visualization of robots. The above algorithm was implemented in ROS as a global planner for path planning in a known environment. The planner works on grid-world environment, where the map of the search space is already made available. With the provided map, we can generate a path between our starting and goal coordinates.

Assumptions

- The PSO algorithm only acts as the guiding agent to determine the points on the search space through which the vehicle should move to the desired point.
- Navigating from one point to another was carried on by the underlying A* planner.
- Another local planner was working to avoid any dynamic obstacles or obstacles not mapped earlier.

This kind of setup was created to allow a swarm of robots' setup, where multiple bots would be placed in the environment and a single PSO planner would command all the bots to get to the target position following their individual optimum path.

12.4.5.1 Result Visualizations in ROS

The package utility was tested on a custom four-wheeled non-holonomic robot, mounted with a laser scanner, IMU, camera, and wheel encoders.

Visualization tool: RViz

Parameters Considered

- Boundary dimensions: 4000*4000 pixels.
- Swarm size: 30.
- Number of max iterations: 200.
- Inertial weight coefficients: w_{\min} , 0.4; w_{\max} , 0.9.
- Local and social coefficients: c1, 2; c2, 2.
- Max allowed velocity: width/10.
- Distance threshold to reach near goal for success criteria: width/60.

Points to Note

- The green dots represent the particles distributed all over the search space.
- The red blob is our custom robot, surrounded by particles helping in localizing the bot.
- The blue line is the shortest distance (straight-line) plan generated between robot position and the target.

- The global planner currently does not handle the obstacle avoidance; instead, the local planner does it.
- It is observed that the particles are denser near the goal than in any other location. This is because the images were captured near the end of the run when convergence happens (Fig. 12.9).

12.5 Future Research Directions

As PSO is a part of the evolutionary algorithms, which is further a branch of artificial intelligence, there are plenty of scope of its applications in optimization problems involving learning strategies. As a future step, the application can be extended to a multi-agent multi-target system. Such cases with high dimensionality and multi-modality will be difficult be solve using classical methods and require improved meta-heuristic techniques. The algorithm needs to be modified to accommodate such requirements. The implementation can be further improved by using additional features to it. The initialization of the particles can be done using uniform random distribution of particles. This will help in avoiding local optima in the beginning of the search and help in better global search. Further, the local and social components c_1 and c_2 are usually fixed and determined using trial and error. This can be avoided if these components are considered as function of inertia weight. It will reduce the dependency of parameters and improve the uniformity of the algorithm.

As a future research, the fitness function dependencies can be modified to make the algorithm generic for any optimization problem and not just for path planning purposes. Research has shown that by using different fitness function, it can be used in neural network architectures for updating weights replacing back-propagation method. The fitness function can also be modified to assign different weights to multiple targets. In a multi-agent system, this will avoid cluttering of particles at any single target, and all the targets can be reached efficiently.

Presently, the algorithm assumes a fixed number of circular obstacles of equal sizes randomly placed in the search area. Since this is not very practical, obstacles of different shapes and sizes need to be implemented. Along with that, moving or dynamic obstacles also need to be considered, and circumference of the obstacle in the field of vision of the particle must be used as obstacle markers. This will further improve the obstacle avoidance module.

In this work, it is assumed that the target is static. But in real-world multi-target situations such as search and rescue scenario, the victims may move from their initial position. Thus, the algorithm can be extended to handle dynamic goal position criteria. Cubic spline curves can be implemented to make the orientation changes smoother for a non-holonomic robot. At each global best particle position, the algorithm should decide on the best coefficients for the cubic spline curves. This will ultimately make all cubic spline curves converge on the optimal spline connecting the start and goal positions. The path followed by the optimal spline will be used for the path planning.

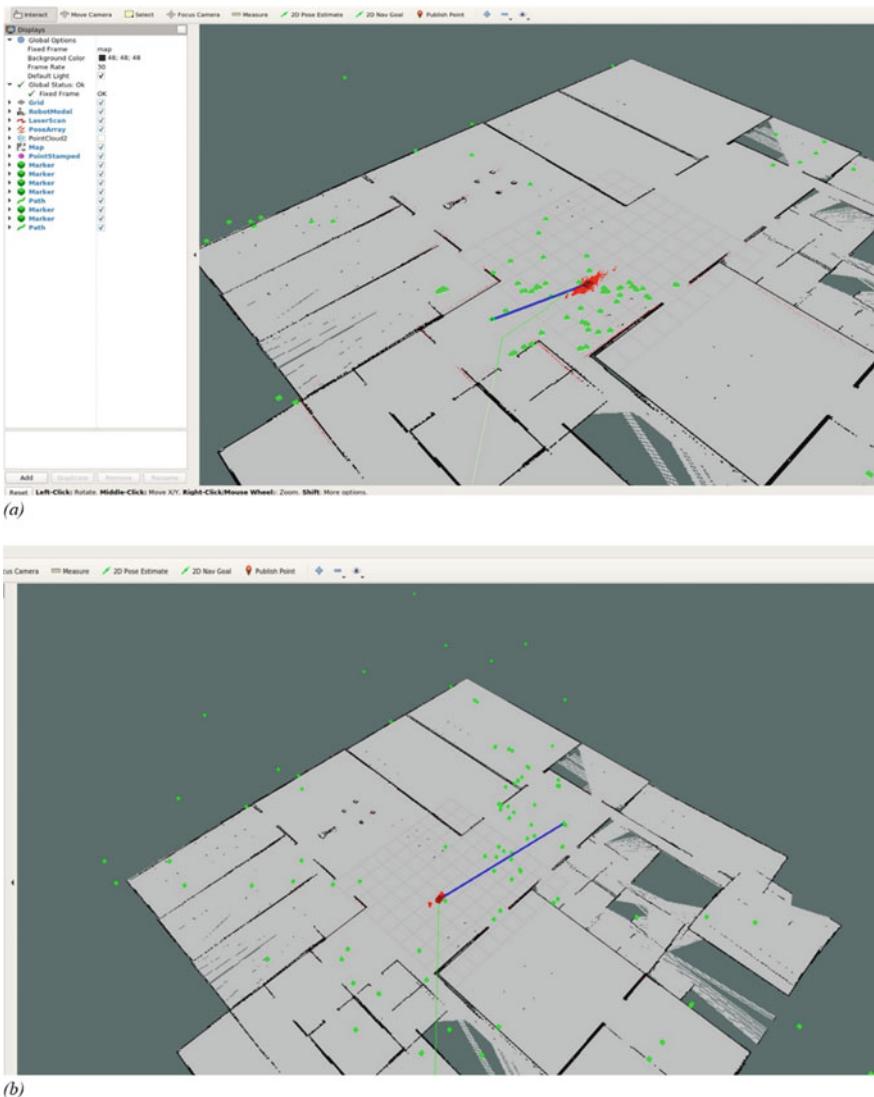


Fig. 12.9 Visualizing the robot's position changes in RViz via ROS at two different time instances.
Source: Author's own creation

Finally, this work can be implemented into a real-world miniature ground bot swarm or on a swarm of micro aerial vehicles. Its working in ROS Gazebo simulation already brings it one-step closer to real-world implementation. With minor calibrations and model changes, this can be done as a future work. In this way, the algorithm's efficiency can be further compared against other advanced swarm robotic algorithms.

12.6 Conclusion

In this paper, an approach for global path planning of autonomous robots is demonstrated using particle swarm optimization. In view of the problems of classical path planning methods, such as being deterministic in nature and computationally intensive, the meta-heuristic method of particle swarm optimization is used, which can be further extended in the case of multi-agent swarm system. The working of few other meta-heuristic methods such as ant colony optimization and artificial bee colony is discussed in brief. In comparison, PSO was considered for simplicity and faster convergence. Since basic PSO version also has the problems of falling into local optimal solutions and low convergence accuracy, it was further modified following various research works. These included applying perturbations to the global best particles to avoid local optimal solutions, using non-linear decreasing inertial weight, and modifying basic fitness function to include the smoothness criteria. This improved algorithm is tested in various simple and complex environment, which included up to ten circular obstacles. From the simulation results, it can be verified that the implemented method is able to deal with the global planning requirements.

- The convergence is fast and takes very few iterations as can be observed from the output log messages.
- The obstacles were randomly placed in the environment and the algorithm avoided all of them.
- None of the generated particles is placed inside the obstacle boundaries after updating their positions at every step.
- In few cases, the algorithm passed through narrow passages in between bigger obstacles to reach the goal following the optimal path.

Thus, our algorithm provided faster convergence and performed well in tricky situations even when the environment was cluttered. For the current ideal scenario, the optimal global path planning was achieved with high accuracy, but in case of dynamic and unknown environments, the accuracy may vary slightly. By varying the parameters, the algorithm can be tweaked according to the problem statement. In further study, this will be extended in robotic swarm system on ROS or real hardware platform, to improve its practical applications.

References

- Adamu, P., Jegede, J., Okagbue, H., & Oguntunde, P. (2018). *Shortest path planning algorithm—A particle swarm optimization (PSO) approach*.
- Aparra. (2016, Dec 22). *Applying genetic algorithms to define a trading system*. Retrieved from QuantDare: <https://quantdare.com/ga-to-define-a-trading-system/>.
- Belwariar, R. (2018, Sep 07). *A* search algorithm*. Retrieved from Geeksforgeeks: <https://www.geeksforgeeks.org/a-search-algorithm/>.

- Di Mario, E., Talebpour, Z., & Martinoli, A. (2013, June). A comparison of PSO and reinforcement learning for multi-robot obstacle avoidance. In *IEEE congress on evolutionary computation* (pp. 149–156). New York: IEEE.
- Dorigo, M., & Di Caro, G. (1999, July). Ant colony optimization: A new meta-heuristic. In *Proceedings of the 1999 congress on evolutionary computation-CEC99 (Cat. No. 99TH8406)*. 2 (pp. 1470–1477). New York: IEEE.
- Eberhart, R., & Kennedy, J. (1995, November). Particle swarm optimization. In *Proceedings of the IEEE international conference on neural networks* (pp. 1942–1948). New York: IEEE.
- Gao, Q. (2018). Intelligent vehicle global path planning based on improved particle swarm optimization. *Open Access Library Journal*, 5, e4587.
- Hassan, R., Cohanim, B., De Weck, O., & Venter, G. (2005, April). A comparison of particle swarm optimization and the genetic algorithm. In *46th AIAA/ASME/ASCE/AHS/ASC structures, structural dynamics and materials conference*, p. 1897.
- Heppner, F., & Grenander, U. (1990). A stochastic nonlinear model for coordinated bird flocks. In *The ubiquity of chaos* (pp. 233, 238).
- Huang, S., & Liu, X. (2013). Application of artificial bee colony-based optimization for fault section estimation in power systems. *International Journal of Electrical Power & Energy Systems*, 44(1), 210–218.
- Islam, M., Tajmiruzzaman, M., Muftee, M., & Hossain, M. (2014). Autonomous robot path planning using particle swarm optimization in dynamic environment. *International conference on mechanical, industrial and energy engineering*.
- Kennedy, J. (2006). Swarm intelligence. In *Handbook of nature-inspired and innovative computing* (pp. 187–219). Boston, MA: Springer.
- Laskari, E., Parsopoulos, K., & Vrahatis, M. (2002, May). Particle swarm optimization for minimax problems. In *Congress on Evolutionary Computation (Cat. No. 02TH8600)* (pp. 1576–1581). New York: IEEE.
- LaValle, S. (1998). Rapidly-exploring random trees: A new tool for path planning. *Technical Report, Computer Science Department, Iowa State University*.
- Li, G., & Chou, W. (2018). Path planning for mobile robot using self-adaptive learning particle swarm optimization. *SCIENCE CHINA Information Sciences*, 61(5), 052204.
- Macedo, I. (2018, Dec 25). *Implementing the Particle Swarm Optimization (PSO) Algorithm in Python*. Retrieved from Medium: <https://medium.com/analytics-vidhya/implementing-particle-swarm-optimization-pso-algorithm-in-python-9efc2eb179a6>.
- Masehian, E., & Sedighizadeh, D. (2010, March). A multi-objective PSO-based algorithm for robot path planning. In *IEEE International Conference on Industrial Technology* (pp. 465–470). Via del Mar: IEEE.
- Nowak, A., Szamrej, J., & Latané, B. (1990). From private attitude to public opinion: A dynamic theory of social impact. *Psychological Review*, 97(3), 362.
- Reeves, W. T. (1983). Particle systems—a technique for modeling a class of fuzzy objects. *ACM Transactions on Graphics*, 2(2), 91–108.
- Rehman, A., Awuah-Offei, K., Baker, D., & Bristow, D. (2019). *Emergency evacuation guidance system for underground mines* (pp. 19–100). Denver, CO: SME Annual Meeting.
- Reynolds, C. W. (1987, August). Flocks, herds and schools: A distributed behavioral model. *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, (pp. 25–34).
- Sedighizadeh, D., & Masehian, E. (2009). Particle swarm optimization methods, taxonomy and applications. *International Journal of Computer Theory and Engineering*, 1(5), 486.
- Shi, Y., & Eberhart, R. (1998, May). A modified particle swarm optimizer. In *IEEE international conference on evolutionary computation proceedings (Cat. No. 98TH8360)* (pp. 69–73). New York: IEEE World Congress on Computational Intelligence.
- Spears, W., Green, D., & Spears, D. (2012). Biases in particle swarm optimization. In *Innovations and developments of swarm intelligence applications* (pp. 20–43). Hershey: IGI Global.

- Wikipedia. (2020). *Ant colony optimization algorithms*. Retrieved from Wikipedia. https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms.
- Xu, H., Liu, J., & Lu, Z. (2016). Structural damage identification based on cuckoo search algorithm. *Advances in Structural Engineering*, 19(5), 849–859.
- Xu, Y., Fan, P., & Yuan, L. (2013). A simple and efficient artificial bee colony algorithm. *Mathematical Problems in Engineering*.

Additional Readings

- Arana-Daniel, N., Gallegos, A. A., López-Franco, C., & Alanis, A. Y. (2012). Smooth path planning for mobile robot using particle swarm optimization and radial basis functions. In *Proceedings of the international conference on genetic and evolutionary methods (GEM)* (p. 1). The steering committee of the world congress in computer science, computer engineering and applied computing (WorldComp).
- Carvalho, M., & Ludermir, T. B. (2007, September). Particle swarm optimization of neural network architectures and weights. In *7th International Conference on Hybrid Intelligent Systems (HIS 2007*) (pp. 336–339). New York: IEEE.
- Chen, X., & Li, Y. (2006, June). Smooth path planning of a mobile robot using stochastic particle swarm optimization. In *2006 International conference on mechatronics and automation* (pp. 1722–1727). New York: IEEE.
- Fetanat, M., Haghzad, S., & Shouraki, S. B. (2015, April). Optimization of dynamic mobile robot path planning based on evolutionary methods. In *2015 AI & Robotics (IRANOPEN)* (pp. 1–7). New York: IEEE.
- Kavraki, L., Svestka, P., Latombe, J., & Overmars, M. (1996). Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4), 566–580.
- Kumar, A. S., Manikutty, G., Bhavani, R. R., & Couceiro, M. S. (2017, September). Search and rescue operations using robotic darwinian particle swarm optimization. In *2017 International conference on advances in computing, communications and informatics (ICACCI)* (pp. 1839–1843). New York: IEEE.
- KumarPatel, P., Sharma, V., & Gupta, K. (2013). Guaranteed convergence particle swarm optimization using personal best. *International Journal of Computer Applications*, 73(7), 6–10.
- Li, Y., & Chen, X. (2005, August). Mobile robot navigation using particle swarm optimization and adaptive NN. In *International conference on natural computation* (pp. 628–631). Berlin, Heidelberg: Springer.
- Rastgoo, M. N., Nakisa, B., & Ahmad Nazri, M. Z. (2015). A hybrid of modified PSO and local search on a multi-robot search system. *International Journal of Advanced Robotic Systems*, 12 (7), 86.
- Rehman, A., Awuah-Offei, K., Baker, D., & Bristow, D. (2019). Emergency evacuation guidance system for underground mines. SME Annual Meeting, Denver, CO (pp. 19-100).
- Saska, M., Macas, M., Preucil, L., & Lhotska, L. (2006, September). Robot path planning using particle swarm optimization of Ferguson splines. In *2006 IEEE conference on emerging technologies and factory automation* (pp. 833–839). New York: IEEE.
- Shiltagh, N. A., & Jalal, L. D. (2013). Optimal path planning for intelligent mobile robot navigation using modified particle swarm optimization. *International Journal of Engineering and Advanced Technology*, 2(4), 260–267.
- Smith, L. L., Venayagamoorthy, G. K., & Holloway, P. G. (2006). *Obstacle avoidance in collective robotic search using particle swarm optimization*.

Chapter 13

Training Multi-layer Perceptron Using Hybridization of Chaotic Gravitational Search Algorithm and Particle Swarm Optimization



Sajad Ahmad Rather, P. Shanthi Bala, and Pillai Lekshmi Ashokan

Abstract A novel amalgamation strategy, namely, chaotic gravitational search algorithm (CGSA) and particle swarm optimization (PSO), has been employed for training multi-layer perceptron (MLP) neural network. It is called CGSAPSO. In CGSAPSO, exploration is carried out by CGSA, and exploitation is performed using PSO. The sigmoid activation function is utilized for training MLP. Besides, a matrix encoding strategy has been used for providing a synergy between neural biases, weights, and CGSAPSO searcher agents. To validate the effectiveness of the hybrid framework, CGSAPSO is applied to three different classification datasets, namely, XOR, Iris, and Balloon. The investigation of results is carried out through various performance metrics like average, standard deviation, median, convergence speed, execution time, and classification rate analysis. Besides, a pair-wise non-parametric signed Wilcoxon rank-sum test has also been conducted for statistical verification of simulation results. In addition, the numerical outcomes of CGSAPSO are also compared with standard GSA, PSO, and hybrid PSOGSA. The experimental results indicate that CGSAPSO provides better results in the form of recognition accuracy and global optima as compared to competing algorithms.

Keywords Gravitational Search Algorithm · Chaotic maps · Hybridization · Optimization · Swarm intelligence · Multi-layer perceptron (MLP) · Particle swarm optimization (PSO) · Exploration · Exploitation

Nomenclature

Symbol	Extension
CGSAPSO	Chaotic GSA and Standard PSO
ABC	Artificial Bee Colony algorithm
MSE	Mean square error

S. A. Rather (✉) · P. S. Bala · P. L. Ashokan
Pondicherry University, Kalapet, India

DE	Differential evolution
CFO	Central force optimization
WOA	Whale optimization algorithm
BSA	Bat-swarm algorithm
ChOA	Chimp optimization algorithm
FNN	Feedforward neural network
ECA	Evolutionary centers algorithm
FA	Firefly algorithm
ACO	Ant colony optimization
CS	Cuckoo search algorithm
CPSOGSA	Constriction Coefficient Based PSO and GSA
GWO	Grey wolf optimizer
C _{Rate}	Classification rate

13.1 Introduction

Neural networks (NN) are powerful information processing systems utilized in the area of meta-heuristics. The first principal mathematical model for NNs was invented in 1943 by McCulloch and Pitts. Actually the neural network concept is inspired by the parallelism mechanism of the human brain. The researchers have introduced different variants of NNs including radial basis function (RBF) (Dorffner, 1996), FNN (Bebis & Georgopoulos, 1994), and so on. In fact, FNN is the most well-known and highly used NN as far as practical real-life problem-solving is concerned.

It is obvious that various types of NNs have a number of differences in their structure and numbers of hidden layers; however, learning has been found to be a common feature among them. Primarily, learning is defined as the process through which a NN learns from the training samples. Moreover, commonly, there are two forms of learning, namely, supervised learning (SL) and unsupervised learning (UL). In SL, the NN needs human-labeled data for prediction. In contrast, UL does not require any pre-existing labels and external human supervisor for learning purposes. It is a form of self-organization that creates a channel between input samples and probability densities.

Likewise, the trainer is another essential element of the neural network information processing which is used for learning a NN. Actually, the effectiveness and accuracy of a NN depend on the trainer because it helps in finding the correct input-output combinations from the test samples. Moreover, training is the foremost step in the recognition process in SL in which the performance of the NN increases iteratively.

It has been found that there are two main mechanisms, namely, deterministic and stochastic, for training a NN. In deterministic methods, the output of the training phase is related to the input values and initial conditions. The examples include back

propagation (BP) and gradient descent (GD) methods. However, stochastic methods consist of randomness and probabilistic values which help to increase the performance.

Additionally, it has been proved that deterministic methods have the advantages of simple design and high exploitation power. Also, it has been reported that the deterministic algorithms provide better convergence toward optimal solutions than heuristic techniques (Jacobs, 1988; Ooyen & Nienhuis, 1992; Weir, 1991). But they have the drawbacks of local minima entrapment and sensitivity in initialization. Moreover, the algorithm revolves around the sub-optimal regions while finding the best solution. It leads to the unnecessary increase in processing time of the training process.

On the other hand, the best characteristic of the stochastic methods is the random initialization of the candidate solutions when optimization process starts. It helps in the diversification and local searching of the solution domain which in turn makes the candidate solutions to avoid local minima traps. Moreover, it has been found that stochastic algorithms are quite popular among researchers due to their solution quality, less sensitivity in initialization, and the ability to remain away from infeasible solutions.

In literature, there have been a number of stochastic algorithms employed for training MLP NNs such as (Mendes, Cortez, Rocha, & Neves, 2002), ABC (Karaboga, Akay, & Ozturk, 2007), GSA (Mirjalili, Mohd Hashim, & Moradian Sardroodi, 2012), and DE (Llonen, Kamarainen, & Lampinen, 2003). The recent inclusions into the list of stochastic algorithms for MLP training are CRO (chemical reaction optimization) (James, Lam, & Li, 2011), CFO-PSO (Green II, Wang, & Alam, 2012), CSS (charges system search) (Pereira et al., 2013), and SSO (Social Spider Optimization) (Pereira, Rodrigues, Ribeiro, Papa, & Weber, 2014).

In this chapter, the chaotic gravitational search algorithm and particle swarm optimization (CGSAPSO) are employed for training multi-layer perceptron neural network. In CGSAPSO, exploration is carried out by CGSA, and exploitation is performed using PSO. The sigmoid activation function is utilized for training MLP. To validate the effectiveness of the hybrid framework, CGSAPSO is applied to three different classification datasets, namely, XOR, Iris, and Balloon.

The rest of the chapter is structured as follows: First, the literature survey regarding MLP training algorithms is covered. Then, MLP and chaos theory concepts are explained. Next, a brief discussion of CGSA, PSO, and CGSAPSO is carried out appropriately. Subsequently, the proposed CGSAPSO-MLP trainer is introduced. Afterward, the simulation analysis of experimental outcomes is carried out. Finally, the conclusion and future direction of the chapter are provided.

13.2 Literature Survey

ANN consists of neurons connected with each other resembling the structure of the brain. Each neuron is provided with some input data for processing and communication. Also, a weight is associated with each of the connections. In mathematical formulation, training NN is to adjust weight coefficients that fulfill certain criteria. In other words, the ANN is a program that emulates the human brain. The ANN learns through training and identifies the pattern from the input data depending on the design structure. Hence, training and layer structure are important steps for an efficient ANN.

A multi-layer perceptron (MLP) is one of the most popular types of NN. MLP consists of neurons that are ordered into input, output, and hidden layers, respectively. The backpropagation algorithm (BP) is one of the gradient descent techniques that is widely used in training MLP. However, BP suffers from slow convergence rate and trapping in the local minima due to its gradient nature. Therefore, researchers have made some improvements in the conventional BP algorithm to increase its intensification. The improvement is made by using modified gain in the objective function. The simulated results were verified on four recognition benchmarks. The improved BP provided better results on all four datasets (Nawi, Ransing, Salleh, Ghazali, & Hamid, 2010).

As compared to BP, heuristic algorithms (HAs) have the property of randomness and stochasticity. These algorithms find the best solution among the pool of candidate solutions. For a meta-heuristic algorithm, it is important to maintain a balance between exploration and exploitation (Hussain, Salleh, Cheng, & Shi, 2019). Genetic algorithm (GA) is one of the first heuristic algorithms (HA) used for training the neural networks among all the evolutionary algorithms (EA) (Itano, DeSousa, & Hernandez, 2018). GA is inspired by the concept of evolution that belongs to a larger class of EA. GA uses the biology operators, namely, selection, mutation, and crossover, to find the best solution. But, it has the drawback of low exploitation.

Many heuristic algorithms have been inspired by nature. Swarm intelligence is a new approach to problem-solving which mimics the lifestyle of birds, ants, and fishes. ACO is a HA based on the group behavior of ants in nature proposed by Marco Dorigo, Birattari, and Stutzle (2006). On the other hand, PSO is inspired by the migrating flock of birds. In nature, the birds communicate with one another by changing position and move towards the one having the best position. In the same way, the PSO algorithm works considering particles as solutions. The particle with the optimal position acts as the best solution. The PSO guarantees the exploitation of the search space. It has been used to train MLP for the classification of the Iris dataset (Kennedy & Eberhart, 1995). In another study (Gudise & Venayagamoorthy, 2003), PSO and BP have been compared for training the neural network. The results proved that PSO is better for applications that require fast learning algorithms. The PSO has been modified in several ways to improve its exploration ability. One of the modifications was to add a constriction coefficient to the mathematical equation of PSO. The constriction coefficient converges the solutions to the global optimum, and

it has been mathematically modeled by Clerc & Kennedy in 2002. An improved version of PSO and GSA for training MLP has been recently proposed (Rather & Bala, 2019a, 2020a) where nine different datasets and five benchmark metrics have been utilized for measuring the performance of CPSOGSA in training MLP. The CPSOGSA provided better outcomes. However, it was behind BBO in the overall simulation analysis. Also, the MLP has been widely used in decision-making systems in the medical sector to help doctors. But, MLP sometimes get trapped in local optima. To alleviate the aforementioned problem, fuzzy rules were embedded in PSOGSA for the optimization of PSO and GSA parameters to improve the performance of the MLP classifier. The simulation results confirm 100% classification accuracy of PSOGSA for all the datasets (Huang & Chou, 2019).

The GWO is another HA based on the swarm hunting behavior of the wolves. It has been utilized for training MLP. UCI datasets were used to test the performance of GWO (Mirjalili, 2015). The WOA is a recent HA which models the behavior of humpback whales. In WOA, the best search agent is used to chase the prey. Besides, exploration is carried out by simulating the bubble hunting technique of the humpback whale. It has been applied to a number of engineering problems (Mirjalili & Lewis, 2016).

Another well-regarded algorithm in the hierarchy of HAs is GSA. It is based on the law of gravity and motion. In GSA, all the masses are considered as solutions. The one having the larger mass is treated as the best solution. The masses get attracted to each other due to their weights, and the one having the heavier weight attracts other masses toward itself. Hence, all the solutions are in dynamic motion. The solution having a large magnitude of mass will have a slower movement. The position of the optimal mass determines the best solution. It has been reported that GSA has local minima issues. To overcome this problem, a study was done by Mirjalili and Gandomi (2017) in which chaos theory concepts were introduced in GSA. The experiments were conducted on 23 benchmark functions. The hybrid chaotic GSA has been compared with many other heuristic algorithms including conventional GSA. The results proved that CGSA has better exploration and exploitation capability as compared to GSA. In a recent work (Rather & Bala, 2020b), CGSA was used for engineering optimization. The CGSA provided optimal outcomes.

The researchers have employed hybridization of the heuristic algorithms for training MLP. Taking the advantages of the GSA and PSO, the hybrid PSOGSA works effectively for training MLP. The PSO is the algorithm that promises exploitation and GSA provides exploration. In MLP, training is important to minimize the error. During the training, the goal is to find the connection weight and bias. Using PSOGSA, the minimum error is achieved by doing experiments on three problems. The PSO is used to exploit the global best from the solutions explored by the GSA through updating mass, force, and acceleration of the searcher agents. The simulation results indicated that PSO and PSOGSA have more accuracy compared to GSA. Moreover, PSOGSA provided the best convergence rate (Mirjalili et al., 2012).

The BBO was introduced by (Simon, 2008) and has also been used to train MLP. It is inspired by the concepts of biodiversity and the ecosystem. It is obvious that the

balance of the ecosystem is maintained by the habitats and the habitants. The habitants emigrate and immigrate from one habitat to another. In BBO, HSI (habitat suitability index) acts as an objective function. Habitants living in the habitat with high HSI value tend to migrate to the habitat with low HSI value. This mechanism provides the exploration power of the algorithm. Higher exploration power avoids trapping in local minima. Besides, the HSI is also improved over generations which guarantee the exploitation. On the other side, mutation helps in improving the exploitation of the algorithm, whereas elitism saves the best solutions. The recognition datasets and mathematical functions were used for benchmarking. In all the experiments, the BBO trained MLP outperformed conventional BP (Saremi, Mirjalili, & Lewis, 2014).

The BA (bat algorithm) (Yang & Gandomi, 2012) is one of the novel HAs based on the bat echolocation. Bats in nature are different from each other in their physical characteristics, but when it comes to hunting and chasing prey, all of them have similar behavior. They have natural sonar used for navigating and hunting the prey. These characteristics of bat have been used in designing the BA. The mathematical formulae which represent the vectors guarantee the exploration ability. Besides, a random walk procedure provides intensification. A binary version of the BA has been proposed by Mirjalili et al. (Mirjalili, Mirjalili, & Lewis, 2014; Mirjalili, Mirjalili, & Yang, 2014) in which results were compared with PSO and other HAs. Binary BA outperformed by giving efficient results. Moreover, chaos theory concepts have also been introduced into BA. From the 13 chaotic maps, the sinusoidal map showed optimal performance (Gandomi & Yang, 2014).

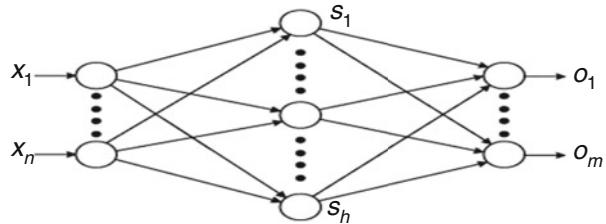
In the family of HAs, Chimp Optimization algorithm (ChOA) is a novel stochastic algorithm which models the hunting tactics of chimps and has been used to train the MLP to classify the underwater acoustical datasets. They hunt in groups by categorizing themselves as the attackers, barriers, chasers, and drivers, all designated to do different tasks. The ChOA classifier outcomes were compared with the hybrid PSOGSA, GWO, and IMO (ion motion algorithm). The measured metrics were the convergence speed, trapping in local minima, and classification accuracy (Khishe & Mosavi, 2020). A newly proposed algorithm is the bird-swarm algorithm (BSA) inspired by the behavior of the birds. It is quite common that HAs normally suffer from trapping in local minima and slower convergence, and the newly proposed BSA also has a similar problem in some situations. So, chaos was introduced into BSA. Moreover, CBSA has been applied to engineering design problems. The experimental studies depicted that embedding of chaotic maps into standard BSA has shown significantly better results (Altay & Alatas, 2020). There are also physics-based algorithms that are used to train the MLP, viz., GSA, CFO, and so on. Another such algorithm is the newly proposed evolutionary centers algorithm (ECA). The algorithm is based on the center of the mass of any object. The center of the mass concept is used to move the worst elements to a better place for the navigation of the solution space. ECA provided optimal results for mathematical functions confirming its global optimization capability (Mejía-de-Dios & Mezura-Montes, 2019).

It is now quite clear that researchers prefer stochastic techniques over traditional GD methods for MLP training as shown in Table 13.1. It is because well-known

Table 13.1 GD and stochastic algorithms for MLP training

Cited researcher	Learning method	Training algorithm	Behavior of training algorithm	Simulation verification
Kennedy et al.	1995	PSO	HA	Benchmark functions
Svozil et al.	1997	HAs and BP	Gradient descent and HA	Literature survey
Walczak & Cerpa	1999	Swarm algorithms	HA	Literature survey
Gudise et al.	2003	BP PSO	Gradient descent HA	Nonlinear functions
Marco Dorigo	2006	ACO	HA	Benchmark functions
Rashedi et al.	2009	GSA	HA	Benchmark functions
Mirjalili et al.	2010	PSO and GSA	HA	Benchmark functions
Green II et al.	2012	CFO and PSO	HA	Recognition datasets
Mirjalili et al.	2012	PSO and GSA	HA	XOR, Iris, mathematical functions
Mirjalili et al.	2013	BBA	HA	Benchmark functions
Gandomi et al.	2014	CBA	HA	Benchmark functions
Mirjalili et al.	2014	BBO	HA	Recognition datasets, mathematical functions
Mirjalili et al.	2015	GWO	HA	Recognition datasets, function approximation
Mirjalili et al.	2016	WOA	HA	Optimization problems, structural design problems
Mirjalili et al.	2017	CGSA	HA	Benchmark functions
Hussain et al.	2019	PSO, ACO, FA, ABC, and CS	HA	Benchmark functions
Mejía-de-Dios & Mezura-Montes	2019	ECA	HA	Benchmark functions
Altay & Alatas	2020	BSA CBSA	HA HA	Benchmark functions and real-life engineering design problems
Khishe and Mosavi	2020	ChOA	HA	Underwater acoustical dataset
Huang & Chou	2019	PSO, GSA, PSOGSA fuzzy-GSA, and fuzzy-PSOGSA	HA	Chronic kidney disease, mesothelioma disease
Rather & Bala	2020b	CGSA	HA	Mechanical engineering design problems
Rather & Bala	2020a	CPSOGSA	HA	Iris, XOR, Balloon, breast cancer, heart, sigmoid, cosine, sine

Fig. 13.1 Simple MLP system



deterministic techniques like BP provide fair enough convergence behavior; however, they have issues with candidate solution quality during the optimization process. In contrast, stochastic techniques, like GA, PSO, and so on, have a simple design and well-defined intensification and diversification operators which help searcher agents to avoid sub-optimal search space locations. That is why, in this study, newly proposed stochastic hybrid strategy, namely, CGSAPSO has been proposed to train MLP to overcome exploitation and solution quality difficulties in traditional GD techniques like BP.

13.3 FNN and Multi-layer Perceptron (MLP)

The neural networks in which information flows from inputs to outputs are called as FNN systems. In fact, FNN is a unidirectional computational information network. Actually, MLP is the most common type of FNN with I-H-O topology in which I and O represent input and output layers while H indicates hidden layer(s). Figure 13.1 presents the schematic diagram of an MLP system.

In Fig. 13.1, n and h represent inputs and hidden nodes, while m depicts output nodes of the MLP system. The neural error is determined by the fitness function. The procedure for fitness function calculation is provided as follows:

It is quite obvious that weights and biases are represented by the first two layers of MLP. So, the sum of neural inputs is given by Eq. (13.1).

$$s_p = \sum_{i=1}^n w_{ip} \cdot x_i - \emptyset_p, \quad p = 1, 2, \dots, n \quad (13.1)$$

where x is the input and w is the weight of the MLP. Moreover, bias is represented by \emptyset_p .

In MLP, the fitness of inputs is determined by sigmoid function as shown in Eq. (13.2).

$$S_p = \text{sigmoid}(s_p) = \frac{1}{1 + \exp(-s_p)}, \quad p = 1, 2, \dots, n \quad (13.2)$$

Now, Eqs. (13.3) and (13.4) calculate the output of the trained MLP.

$$O_l = \sum_{i=1}^h w_{pl} \cdot s_p - \emptyset_l, \quad l = 1, 2, \dots, m \quad (13.3)$$

$$O_l = \frac{1}{1 + \exp(-o_l)}, \quad l = 1, 2, \dots, m \quad (13.4)$$

It is evident from mathematical equations that the neural biases and weights are important for proper functioning of the MLP system. So, we have employed the CGSAPSO algorithm for MLP training to find the proper values of neural inputs.

13.4 Chaos Theory

Chaos theory, broadly speaking, is a branch of mathematics dealing with the study of dynamic systems. The chaotic systems are highly sensitive to the changes in the initial conditions. In other words, small transformation in the inputs creates large variations in the output. It is obvious that chaotic systems have randomized behavior; however chaotic patterns are also shown by deterministic systems. In the past decade, the researchers have utilized chaos theory for resolving premature convergence and local search issues in HAs.

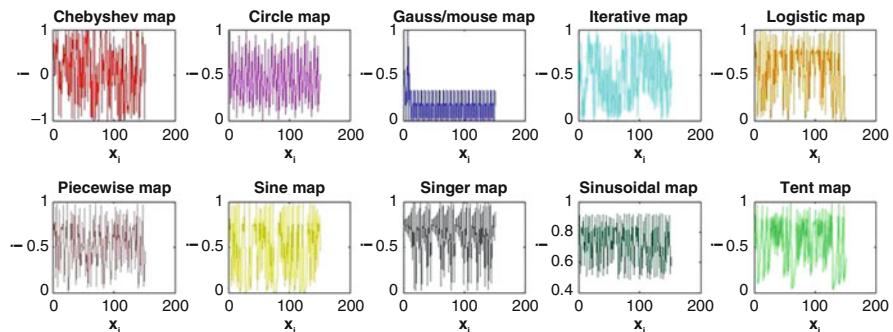
In this work, ten chaotic maps have been utilized to enhance the performance of GSA while intensifying its convergence rate. The mathematical equations of ten chaotic maps are provided in Table 13.2. Also, it can be clearly seen that chaotic maps show random behavior as shown in Fig. 13.2.

13.5 Chaotic Gravitational Search Algorithm

GSA is one of the highly regarded physics-based HAs. It is inspired by the law of gravitation and motion. In fact, gravity is one of the four basic forces in nature. The other three forces are weak nuclear force, electromagnetic force, and the strong nuclear force (Halliday, Resnick, & Walker, 2000; Rashedi, Nezamabadi-pour, & Saryazdi, 2009). Moreover, the law of gravitation states that “the attractive force between two masses is directly proportional to the product of their masses and inversely proportional to the square of the distance between them” (Rather & Bala, 2019b, 2019c; Rather & Sharma, 2017).

Table 13.2 Mathematical equations of chaotic maps

Serial Number	Name of the map	Chaotic equation	Domain
I	Chebyshev	$l_{i+1} = \cos(i \cos^{-1}(l_i))$	(1, -1)
II	Circle	$l_{i+1} = \text{mod}(l_i + q - (\frac{p}{2\pi}) \sin(2\pi l_k), 1)$, $p = 0.5$ and $q = 0.2$	(0, 1)
III	Gauss	$l_{i+1} = \begin{cases} -l, l_i = 0 \\ \frac{1}{\text{mod}(l_i, 1)}, \text{otherwise} \end{cases}$	(0, 1)
IV	Iterative	$l_{i+1} = \sin\left(\frac{p\pi}{l_i}\right)$, $p = 0.7$	(-1, 1)
V	Logistic	$l_{i+1} = p l_i (1 - l_i)$, $p = 4$	(0, 1)
VI	Piecewise	$l_{i+1} = \begin{cases} \frac{l_i}{r}, 0 \leq l_i \leq 0 \\ \frac{l_i - r}{0.5 - r}, r \leq l_i \leq 0.5 \\ \frac{1 - r - l_i}{0.5 - r}, 0.5 \leq l_i \leq 1 - r \\ \frac{1 - l_i}{r}, 1 - r \leq l_i \leq 1 \end{cases}, r = 0.4$	(0, 1)
VII	Sine	$l_{i+1} = \frac{p}{4} \sin(\pi l_i)$, $p = 4$	(0, 1)
VIII	Singer	$l_{i+1} = z (7.86 l_i - 23.31 l_i^2 + 28.75 l_i^3 - 13.302875 l_i^4)$, $z = 1.07$	(0, 1)
IX	Sinusoidal	$l_{i+1} = p l_i^2 \sin(\pi l_i)$, $p = 2.3$	(0, 1)
X	Tent	$l_{i+1} = \begin{cases} \frac{l_i}{0.7}, l_i \leq 0.7 \\ \frac{10}{3}(1 - l_i), l_i \geq 0.7 \end{cases}$	(0, 1)

**Fig. 13.2** Random nature of chaotic maps

The GSA is first initialized with a random distribution of searcher agents in the form of masses. The force between the point masses is calculated in Eq. (13.5).

$$F_{ij} = G(t) \frac{m_{pi}(t)m_{aj}(t)}{R_{ij}(t)+\epsilon} \left(x_j^d(t) + x_i^d(t) \right) \quad (13.5)$$

where $m_{pi}(t)$ and $m_{aj}(t)$ are passive and active gravitational masses, respectively. The Euclidian distance is represented as $R_{ij}(t)$, while ϵ is a small constant.

To get a proper balance between exploration and exploitation, the GSA utilizes an important parameter called gravitational constant represented by “ G .” Besides, it helps in the accuracy of the search. It is given by Eq. (13.6).

$$G(t) = G(t_0) e^{(-\alpha_{MI}^{CI})} \quad (13.6)$$

where $G(t)$ and $G(t_0)$ are the values of the gravitational constant at time interval t and t_0 , respectively. Also, α is an exponentially decreasing coefficient, whereas CI and MI correspond to the current iteration and the maximum number of iteration(s).

In standard GSA, the gravitational constant (G) is the main parameter that specifies the intensity of the gravitational field as shown in Eq. (13.6). In fact, during the initial iteration phase, the value of G decreases exponentially which results in the diversification. Moreover, at last iterations, the value of G changes slowly, hence promoting the exploitation of the candidate solutions toward the global optimum. In CGSA, ten different chaotic maps have been employed to overcome slow convergence and the local searching issue of standard GSA. Chaotic maps create huge changes in the output when the initial conditions of the map(s) are modified. This helps searcher agents to move out of the local minima traps. Besides, chaotic normalization (Mirjalili & Gandomi, 2017) helps in the proper search of the solution space. It is mathematically calculated as shown in Eq. (13.7).

$$C_i^{\text{norm}}(t) = \frac{(C_i(t) - a) * (d - c)}{(b - a)} + c \quad (13.7)$$

In Eq. (13.7), (a, b) and (c, d) are chaotic limits and normal range values, respectively. Moreover, $c = 0$ and d is calculated using Eq. (13.8).

$$d = MI - \frac{CI}{MI} (\text{Max} - \text{Min}) \quad (13.8)$$

Here, MI and CI represent the maximum number of iterations and the current iteration. Besides, adaptive intervals are indicated by Max and Min.

Hence, the chaotic version of gravitational constant is the sum of Eqs. (13.6) and (13.7).

$$G^c(t) = C_i^{\text{norm}}(t) + G(t_0)e^{(-\alpha_{\text{MT}}^{\text{CI}})} \quad (13.9)$$

Equation (13.9) shows that $G^c(t)$ has the interesting properties of randomness, chaotic stochasticity, and adaptive learning capability.

Furthermore, after a number of iterations, the heavy masses will be scattered throughout the search space which represents feasible solutions. So, it is important to preserve the quality of the best solutions. Therefore, elitism criterion, i.e., kbest strategy, is used in GSA. It means that only optimal and efficient heavy mass is having highest intensity after the fulfillment of stopping criteria. It is shown in Eq. (13.10).

$$F_i^d(t) = \sum_{j=\text{kbest}, j \neq i}^m \gamma_j F_{ij}^d(t) \quad (13.10)$$

Moreover, the acceleration of the masses is calculated according to the second law of motion as given in Eq. (13.11).

$$a_i^d(t) = \frac{F_i^d(t)}{m_i(t)} \quad (13.11)$$

In GSA, the point masses get attracted towards heavy masses because they have the highest intensity and strong force of attraction. Hence, it is pivotal to calculate the velocity and position of the best solution in order to find the global optimum as provided in Eqs. (13.12) and (13.13), respectively.

$$v_i^d(t+1) = \gamma_j v_i^d(t) + a_i^d(t) \quad (13.12)$$

$$x_i^d(t+1) = x_i^d(t) + v_i^d(t+1) \quad (13.13)$$

13.6 Particle Swarm Optimization

The PSO is one of the classical techniques in heuristic optimization. It is inspired by the migrating flock of birds. In nature, the birds communicate with one another by changing position and move toward the one having the best position. In the same way, the PSO algorithm works considering particles as solutions. Actually, the position and velocity of the best solutions are calculated using Eqs. (13.14) and (13.15).

$$v_i^d(t+1) = w(t)v_i^d(t) + c_1r_{i1}(\text{pbest} - x_i^d(t)) + c_2r_{i2}(\text{gbest} - x_i^d(t)) \quad (13.14)$$

$$x_i^d(t+1) = x_i^d(t) + v_i^d(t+1) \quad (13.15)$$

such that personal and global constants are represented by c_1 and c_2 . Besides, random numbers for initialization are depicted as (r_{i1}, r_{i2}) .

13.7 Hybrid CGSAPSO

It is essential for every HA to have a one-one correspondence between diversification and exploitation processes because the former one is important for overall navigation while the latter one finds the suitable zones of the domain space. Therefore, in CGSAPSO, CGSA provides global searching support, whereas PSO makes sure that neighborhood regions are exploited optimally. Moreover, velocity of the candidate solutions is as under:

$$V_i^d(t+1) = w(t)V_i^d(t) + c_1r_{i1}(a_i^d(t) - x_i^d(t)) + c_2r_{i2}(\text{gbest} - x_i^d(t)) \quad (13.16)$$

In Eq. (13.16), the velocity of CGSAPSO searcher agents is represented by V_i^d . The location of the best candidate solution is calculated by using Eq. (13.17).

$$X_i^d(t+1) = X_i^d(t) + V_i^d(t+1) \quad (13.17)$$

The pseudo-code of CGSAPSO is presented in Algorithm 13.1. Besides, Fig. 13.3 shows the block diagram of the algorithm.

13.8 MLP Training Using CGSAPSO

All stochastic algorithms consist of searcher agents that are essential for global searching and intensification of the solution space. So, in order to train MLP, searcher agents are encoded as inputs of a neural network. This can be done in three ways. First is vector encoding, in which agents of an optimization algorithm are represented by vectors, while bit encoding gives 0 and 1 values to searcher agents. Moreover, matrix encoding portrays agents as rows and columns of a linear matrix.

When it comes to employ stochastic algorithms for MLP training, the first step is to provide proper values for inputs by reducing activation function overhead. Besides, HAs can also be employed for uncovering an efficient MLP neural structure. In addition, parameter tuning like speed and momentum of neural networks can also be achieved through HAs.

In this study, matrix encoding has been utilized for representing neural inputs by CGSAPSO agents. In Fig. 13.4, we have used 2-3-1 MLP model to explain matrix

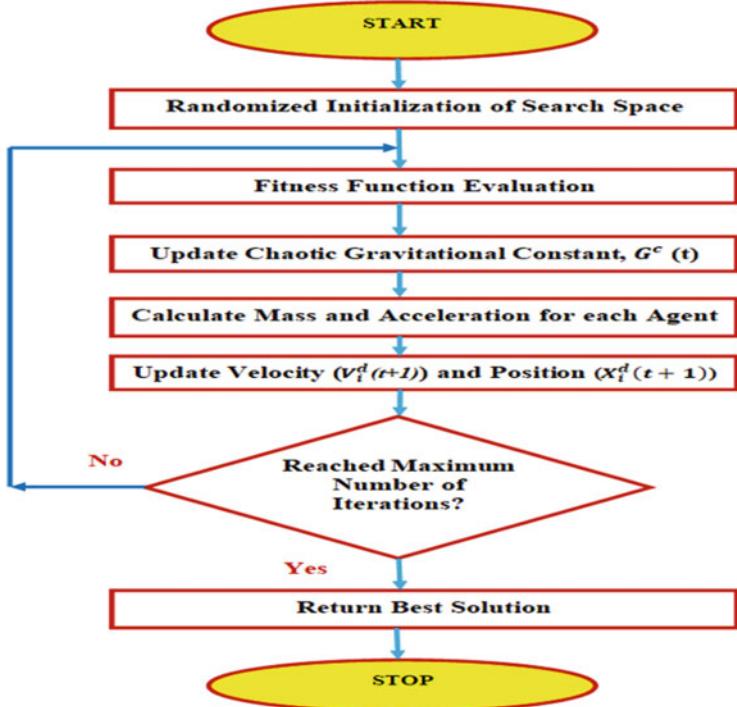


Fig. 13.3 Hybrid CGSAPSO algorithm

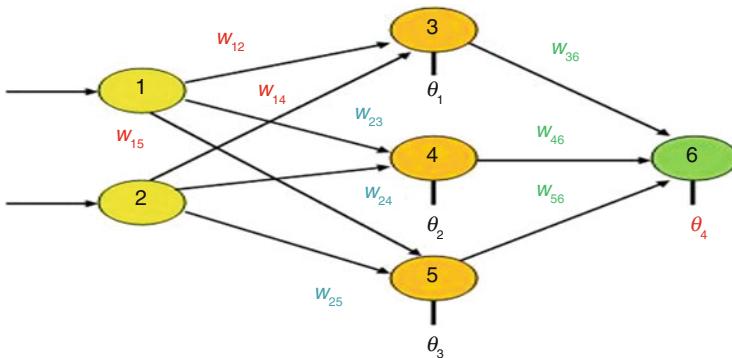


Fig. 13.4 (2-3-1) MLP neural network

encoding scheme. The matrix representation of neural weights and biases is shown in Eqs. (13.18) and (13.19).

Algorithm 13.1: Hybrid CGSAPSO Algorithm

- 1: Initialize the upper and lower limits of the solution space.
 - 2: Evaluate the fitness of searcher agents.
 - 3: Initialize the parameters including the total iterations (T), $G(t_0)$, and α .
 - 4: Start the iteration counter at $t = 0$.
 - 5: **while** $t < T$ **do**,
 - 6: **for** each candidate solution **do**,
 - 7: Find the chaotic behavior, $C_i^{\text{norm}}(t)$ of each point mass with the help of Eq. (13.7).
 - 8: Calculate the chaotic gravitational constant, $G^c(t)$.
 - 9: Using Eq. (13.10) find the gravitational force, $F_i^d(t)$
 - 10: Calculate the mass acceleration $a_i^d(t)$ by using Eq. (13.11)
 - 11: Update the mass velocity $V_i^d(t + 1)$ with the help of Eq. (13.16)
 - 12: Update the mass position $X_i^d(t + 1)$ using Eq. (13.17)
 - 13: **end for**
 - 14: $t = t + 1$
 - 15: **end while**
 - 16: Return the optimal candidate solution
-

$$\text{Particle}(:, :, i) = [w_1, b_1, w'_2, b_2] \quad (13.18)$$

$$w_1 = \begin{bmatrix} w_{13} & w_{23} \\ w_{14} & w_{24} \\ w_{15} & w_{25} \end{bmatrix}, b_1 = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix}, w'_2 = \begin{bmatrix} w_{36} \\ w_{46} \\ w_{56} \end{bmatrix}, b_2 = [\theta_4] \quad (13.19)$$

such that neural weights are represented by $\langle w_1, w_2 \rangle$ while neural biases are shown as $\langle b_1, b_2 \rangle$.

Basically, correct labeling of testing samples is the main goal of MLP learning process. In other words, the classification accuracy of an MLP is crucial for the minimization of the neural error. Besides, MSE is used to calculate the neural error as reported in Eq. (13.20).

$$\text{MSE} = \sum_{i=1}^m (o_i^l - D_i^l)^2 \quad (13.20)$$

such that o and D are system and target MLP outcomes whereas l is the feature sample and i is the neural input.

Also, it is essential for MLP to cover the complete training feature vectors for the better performance. To do that, the average of MSE is calculated as indicated in Eq. (13.21).

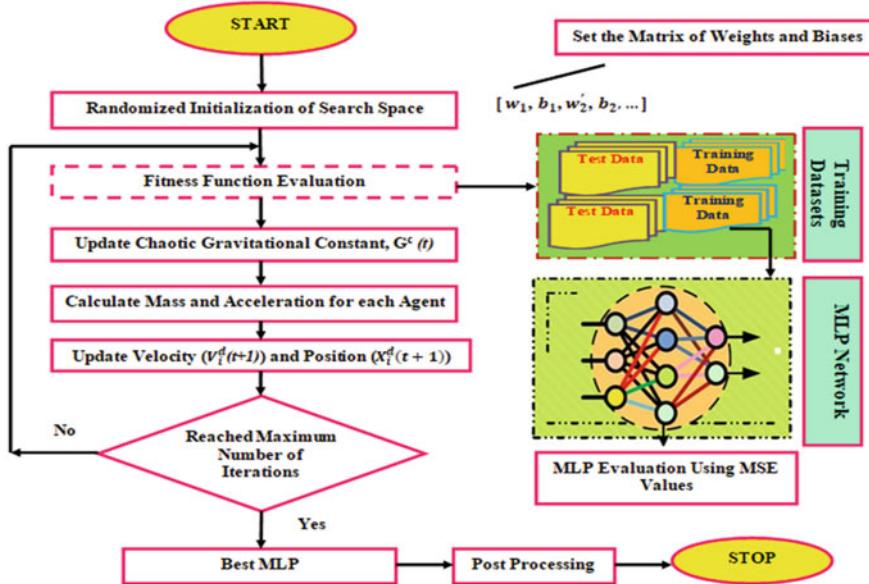


Fig. 13.5 An amalgamation of CGSAPSO and MLP

$$\text{MSE}_{\text{Avg}} = \sum_{l=1}^R \frac{\sum_{i=1}^m (o_i^l - D_i^l)^2}{R}. \quad (13.21)$$

In Eq. (13.21), “ R ” denotes the number of feature samples. Thus, Eq. (13.22) clearly conveys that CGSAPSO MLP training is a minimization process in which solutions (point masses) are modeled as neural weights and biases to increase the prediction accuracy of the input attributes.

$$\text{Minimize : } F(\text{Particle}) = \text{MSE}_{\text{Avg}} \quad (13.22)$$

The MLP training by CGSAPSO is presented in Fig. 13.5. In this work, CGSAPSO has been applied to three classification benchmark functions for verifying its pattern recognition ability. In the next section, the simulation outcomes of different benchmarks using CGSAPSO and other stochastic techniques are reported.

13.9 Experimental Results and Simulation Analysis

Basically, three recognition datasets (Blake & Merz, 1998) are utilized to check the implementation conduct of the proposed CGSAPSO-MLP in finding the optimal neural inputs for training a MLP system. The three datasets are XOR, Balloon, and Iris.

13.9.1 Experimental Setup and Datasets Used

The simulation results of ten versions of CGSAPSO have been compared with three different HAs including GSA, PSO, and PSOGSA (Mirjalili & Hashim, 2010).

The experiments were performed on a computer system having system specifications as mentioned in Table 13.3. Moreover, MATLAB codes are available in <https://github.com/SAJADAHMADI1>.

It is quite obvious that optimization algorithms need a solution space to find the global optimal of the problem. Accordingly, the agents of HAs have been initialized in a search space having limits of $[-10, 10]$. For an impartial comparative analysis, all the algorithms have the same population size of 30. And total iterations of 100 have been chosen as end condition for HAs. The classification datasets are presented in Table 13.4.

In Table 13.4, it is clearly seen that the minimum number of attributes are present in XOR dataset. While Iris recognition dataset is somewhat complex as it has 4 attributes and 150 testing samples. Besides, XOR and Balloon have two classes, whereas the Iris dataset consists of three classes. In addition, the initialization parameters of HAs have been presented in Table 13.5.

The experiments were repeated 20 times to calculate different statistical measures. The statistical analysis has been performed considering the stochastic and random nature of simulation results. Furthermore, CGSAPSO1 to CGSAPSO10 are

Table 13.3 Computer system specifications

System feature	Configuration
Operating system	Windows 10 enterprise
Processor	High speed Intel CPU
RAM	Four giga bytes
Hard disk	500GB
Programming language	MATLAB R2013b

Table 13.4 Classification datasets

Dataset	Features	Training features	Test features	Class	Topology
XOR	3	8	8	2	3-7-1
Balloon	4	16	16	2	4-9-1
Iris	4	150	150	3	4-9-3

Table 13.5 Initialization of HA parameters

Algorithm	Parameter	Value
GSA	Rpower	1
PSO	Intensification constants	2
	Wmax	0.9
	Wmin	0.2
PSOGSA	Coefficient (α)	20
	$G(t_0)$	100

Table 13.6 XOR logic values

X (I/P)			Y (O/P)
F	F	F	F
F	F	T	T
F	T	F	T
F	T	T	F
T	F	F	T
T	F	T	F
T	T	F	F
T	T	T	T

ten chaotic versions of CGSAPSO. The minimum value of mean and standard deviation does not imply that an algorithm is efficient than others (Derrac, García, Molina, & Herrera, 2011). However, statistical tests should be performed on the simulation results to find the optimal competitive algorithm. Therefore, a pair-wise non-parametric signed Wilcoxon rank-sum test (WRST) has been conducted at a 5% significance level to statistically verify the outcomes (Wilcoxon, 1945). The reason behind selecting a Wilcoxon WRST is that it uses median as a statistical measure which is better than average and STD. Moreover, in the WRST the distribution of dataset is not considered. Also, p value is calculated for retaining or rejecting the null hypothesis.

Null hypothesis (H_0): Best performing algorithm does not provide optimal values for the neural inputs of the MLP.

Alternate hypothesis (H_1): Best performing algorithm provides optimal values for the neural inputs of the MLP.

In simpler terms, H_0 consists of an algorithm having p value greater than 0.05, while $P \leq 0.05$ is enough to reject the null hypothesis, that is, H_1 . Besides, N/A indicates the best competing algorithm (PSO, GSA, PSOGSA, or CGSAPSO) which has minimum values for mean and STD, while NN represents chaotic versions of CGSAPSO having large values for mean and STD. The next sections of the chapter deal with the experimental verification of three different HAs including ten versions of CGSAPSO.

13.10 Classification Problems

Generally, the main aim of the hybridization of heuristic algorithm and MLP neural network is to increase the classification accuracy through randomized initialization. In simpler terms, successful training of a neural network means correct labeling of the classification data. So, if the number of features in a dataset is less, that means MLP trained algorithm will take minimum iterations and, hence, less computational time to find the correct values for neural weights. In this work, three classification datasets have been utilized, namely, XOR, Balloon, and Iris, in which XOR has less features while Iris has highest attributes. Therefore, in the next sub-sections, experimental results of all the three datasets have been recorded and discussed.

13.10.1 *XOR Dataset*

The hidden layers in a neural network are mandatory to solve any nonlinear separable problem. Likewise, XOR is a nonlinear mathematical benchmark. The logic behind exclusive OR gate is that the output will be a truth value (1), if the input terminals contain odd number of truth values (1's); otherwise the value of the output terminal will be false (0).

13.10.2 *Simulation Results of XOR Dataset*

The topology used for XOR gate is 3-7-1. Table 13.7 clearly shows that CGSAPSO6 has better values for statistical measures. Also, PSOGSA and other versions of CGSAPSO also depict efficient performance.

Besides, the classification rate (C_{Rate}) of CGSAPSO versions like CGSAPSO1 to CGSAPSO5 is 100% indicating high local searching power. Similarly, CGSAPSO takes less execution time than PSOGSA and GSA. However, PSO takes minimum running time for global optimization as compared to other algorithms. In addition, the PSOGSA simulation values are more statistically significant than CGSAPSO as p values are greater than zero.

The convergence curves of the XOR dataset are presented in Fig. 13.6. It clearly indicates that CGSAPSO has a better convergence rate than GSA and PSO indicating its efficient intensification power in getting global optimum. The box plots (Fig. 13.7) convey that CGSAPSO provides better values for XOR fitness function while PSO has unsatisfactory simulation values for inter quartile range and median.

Table 13.7 Outcomes of XOR dataset

Method	MSE _{best}	MSE _{worst}	MSE _{average}	MSE _{STD}	MSE _{median}	C _{Rate} (%)	P values	Time taken
GSA	0.094158	0.2567	0.21831	0.044457	0.23219	75	0.000103	22.3572
PSO	0.19911	0.374	0.28761	0.047519	0.28514	0	8.857457	17.146
PSOGSA	3.8292e-08	0.12501	0.011434	0.030323	5.89e-05	100	0.411464	22.1368
CGSAPSO1	1.6931e-06	0.092814	0.012634	0.025474	0.002390	100	NN	22.6019
CGSAPSO2	2.2725e-05	0.12499	0.01683	0.036147	0.001401	100	NN	22.5136
CGSAPSO3	2.7087e-07	0.06432	0.006616	0.019723	0.000130	100	NN	22.6765
CGSAPSO4	4.4424e-08	0.03545	0.005689	0.010549	0.000467	100	NN	22.6383
CGSAPSO5	6.982e-09	0.13919	0.014149	0.034944	0.001737	100	NN	22.495
CGSAPSO6	5.9147e-06	0.045017	0.004505	0.010667	0.000378	50	NA	23.022
CGSAPSO7	1.4158e-08	0.11451	0.009610	0.025411	0.007372	87.5	NN	22.6636
CGSAPSO8	8.2972e-07	0.06454	0.011764	0.018981	0.000935	87.5	NN	22.6435
CGSAPSO9	1.3349e-06	0.12524	0.019693	0.034097	0.001966	87.5	NN	22.3277
CGSAPSO10	3.4866e-07	0.054816	0.012743	0.017427	0.003206	100	NN	24.0922

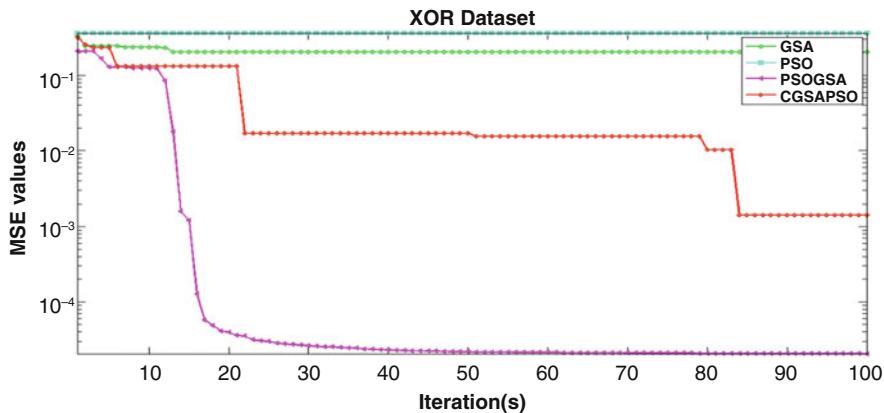


Fig. 13.6 Convergence curve of XOR dataset

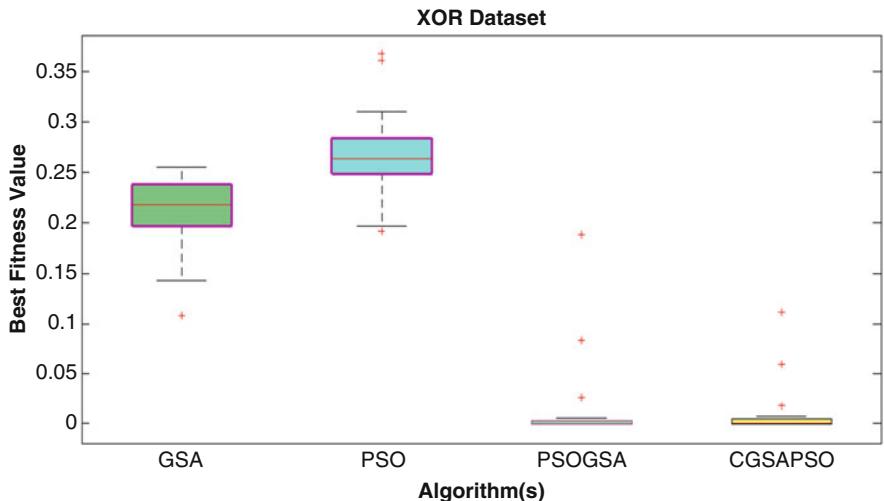


Fig. 13.7 Box plots of XOR dataset

13.10.3 Balloon Dataset

The Balloon dataset is another famous classification benchmark. Actually, Balloon dataset consists of four attributes. During training process, the result of the algorithm is one and, if the balloon inflates, otherwise zero. Besides, the topology employed for Balloon classification benchmark is 4-9-1.

13.10.4 Simulation Results of Balloon Dataset

Table 13.8 provides the simulation outcomes of the participating algorithms. Clearly, CGSAPSO depicts efficient values for $MSE_{average}$ and MSE_{STD} as compared to GSA and PSO. Moreover, PSOGSA also gives good values for the Balloon dataset. When considering the C_{Rate} , most of the CGSAPSO versions including CGSAPSO3 provide 100% recognition accuracy showing its diversification and exploitation capability. Besides, CGSAPSO takes less time than GSA. However, PSO and PSOGSA are faster than CGSAPSO in getting global optimum. In addition, the p values of PSOGSA are statistically more significant than CGSAPSO.

The convergence graph and box plots of the balloon dataset are shown in Figs. 13.8 and 13.9, respectively. The convergence curve of PSOGSA is at the bottom of Fig. 13.8 conveying high exploitation power. On the other hand, CGSAPSO is also a better optimizer than GSA and conventional PSO because it provides better MSE values than both of them, and convergence curves clearly validate this interpretation. Also, box plots indicate that PSO has sub-optimal values, whereas CGSAPSO and PSOGSA have minimum values indicating global optimization capability.

13.10.5 Iris Dataset

In 1936, Donald Fisher introduced iris dataset having 150 samples. This dataset is inspired from the floriculture which includes *setosa*, *versicolor*, and *virginica* as classes. Moreover, the Iris dataset has five features, namely, petal length, petal width, sepal length, sepal width, and species. The topology employed is 4-9-3.

13.10.6 Simulation Results of Iris Dataset

The simulation outcomes of competing algorithms are presented in Table 13.9. The $MSE_{average}$ values of CGSAPSO are better than PSO. However, GSA and PSOGSA also provide competitive values for $MSE_{average}$ and MSE_{STD} . Moreover, the C_{Rate} of CGSAPSO1 (80%) is more than the standard GSA (64%). Besides, CGSAPSO takes less time to find global optimum as compared to GSA and PSOGSA showing its high convergence power. Additionally, the p values indicate that GSA simulation values are more valid than PSOGSA.

Also, Figs. 13.10 and 13.11 present convergence graphs and box plots of the Iris dataset. It can be seen that convergence curves of PSOGSA and GSA overlap with each other indicating symmetrical intensification power. Moreover, the global optimization capability of CGSAPSO is better than PSO because it provides better MSE values than PSO, and convergence curves also validate it. In addition, PSOGSA

Table 13.8 Outcomes of Balloon dataset

Method	MSE _{best}	MSE _{worst}	MSE _{average}	MSE _{STD}	MSE _{median}	C _{Rate} (%)	P values	Time taken
GSA	1.832e-08	0.067699	0.021084	0.023407	0.011228	40	8.857e-05	81.9875
PSO	0.075215	0.31469	0.20675	0.059478	0.20296	0	8.857e-05	76.9748
PSOGSA	7.796e-18	1.117e-06	7.73e-08	2.51e-07	4.90e-11	40	NA	83.6332
CGSAPSO1	2.939e-09	0.50841	2.81e-04	0.01132	2.64e-05	85	NN	88.7842
CGSAPSO2	7.311e-10	0.010119	9.09e-04	0.00252	4.09e-06	70	NN	88.7089
CGSAPSO3	1.074e-11	0.0003295	3.83e-05	7.85e-05	6.27e-06	100	0.000593	87.8833
CGSAPSO4	6.405e-11	1.070e-04	1.62e-04	3.47e-04	1.01e-05	100	NN	87.2306
CGSAPSO5	3.877e-10	2.708e-04	1.98e-04	6.19e-04	1.28e-05	100	NN	87.4532
CGSAPSO6	9.100e-10	6.300e-04	6.62e-04	1.58e-04	6.67e-06	100	NN	112.0174
CGSAPSO7	1.590e-08	9.222e-04	9.65e-04	2.21e-04	5.20e-06	100	NN	121.4285
CGSAPSO8	4.412e-09	0.0243	1.76e-04	5.44e-04	5.31e-06	85	NN	132.6447
CGSAPSO9	1.379e-08	2.729e-04	3.81e-04	7.57e-04	2.02e-05	100	NN	100.7677
CGSAPSO10	7.655e-09	6.242e-04	6.72e-04	1.55e-04	1.70e-05	100	NN	114.4901

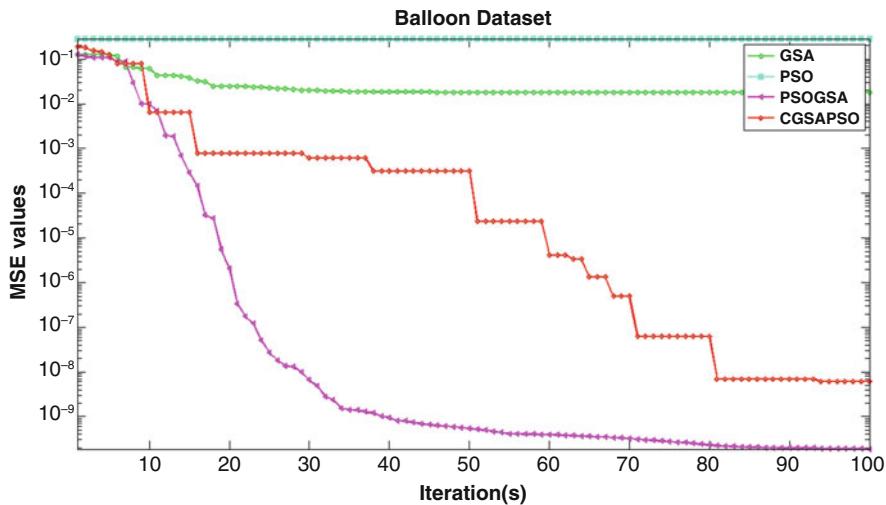


Fig. 13.8 Convergence curve of Balloon dataset

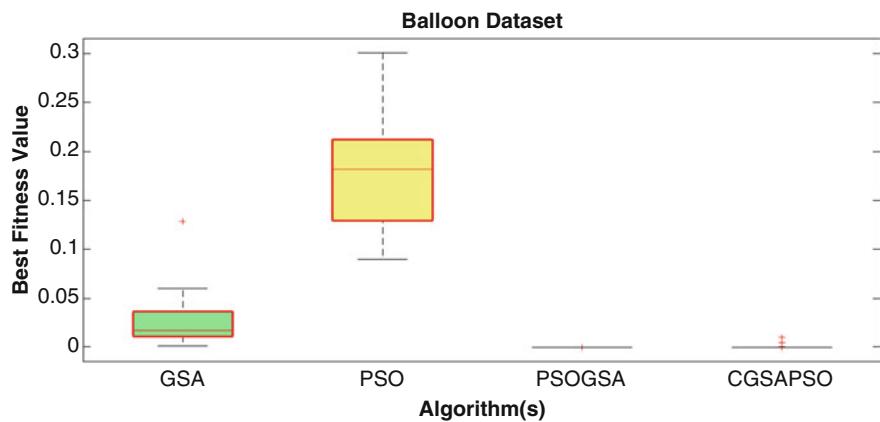


Fig. 13.9 Box plots of Balloon dataset

depicts minimum values for the median statistical measure, while PSO gives maximum quartile and median values showing its sub-optimal performance.

13.10.7 Overall Simulation Results Discussion

It is clearly evident from the simulation results that CGSAPSO provides the best classification accuracy for XOR and Balloon datasets while PSOGSA shows a finest C_{Rate} of 87.33% for Iris dataset. Moreover, MSE values are also minimal for

Table 13.9 Outcomes of Iris dataset

Method	MSE _{best}	MSE _{worst}	MSE _{average}	MSE _{STD}	MSE _{median}	C _{Rate} (%)	P values	Time taken
GSA	0.67079	0.79531	0.70285	0.035853	0.68723	64	0.027621	880.4893
PSO	0.8149	0.99131	0.91976	0.046397	0.91896	82	8.857e-05	714.8638
PSOGSA	0.66683	0.77779	0.68517	0.028702	0.6727	87.33	NA	787.4764
CGSAPSO1	0.75449	0.86095	0.81269	0.030656	0.81894	80	NN	766.8898
CGSAPSO2	0.75226	0.86704	0.8193	0.026588	0.81839	47.33	NN	884.4712
CGSAPSO3	0.71615	0.8604	0.78684	0.033626	0.78772	38.66	8.857e-05	733.4416
CGSAPSO4	0.76889	0.86526	0.81059	0.029111	0.80108	56.66	NN	732.8728
CGSAPSO5	0.77905	0.85509	0.81341	0.023704	0.81649	29.33	NN	729.6862
CGSAPSO6	0.74326	0.87807	0.81461	0.035167	0.81203	52	NN	726.9769
CGSAPSO7	0.70781	0.85935	0.80096	0.043322	0.80785	44	NN	723.7148
CGSAPSO8	0.72282	0.85756	0.80797	0.037465	0.82456	53.33	NN	886.8468
CGSAPSO9	0.76138	0.86939	0.81863	0.026323	0.81336	44.66	NN	731.8447
CGSAPSO10	0.76664	0.84981	0.81022	0.022434	0.81373	65.33	NN	736.4968

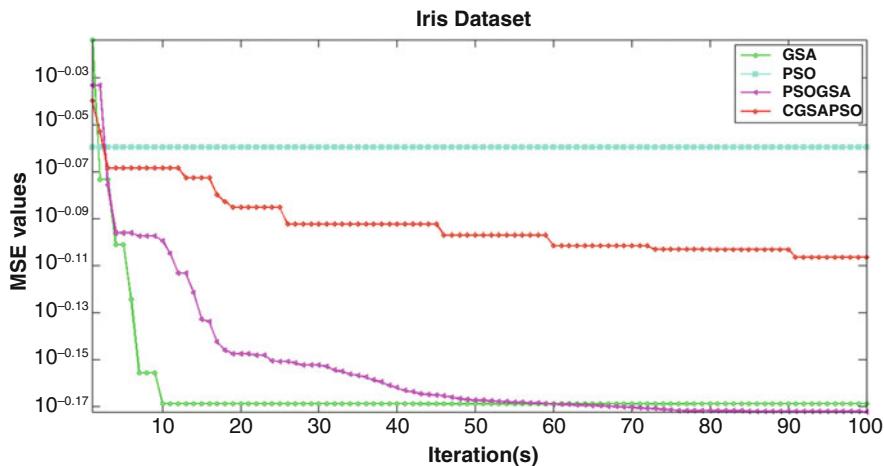


Fig. 13.10 Convergence curve of Iris dataset

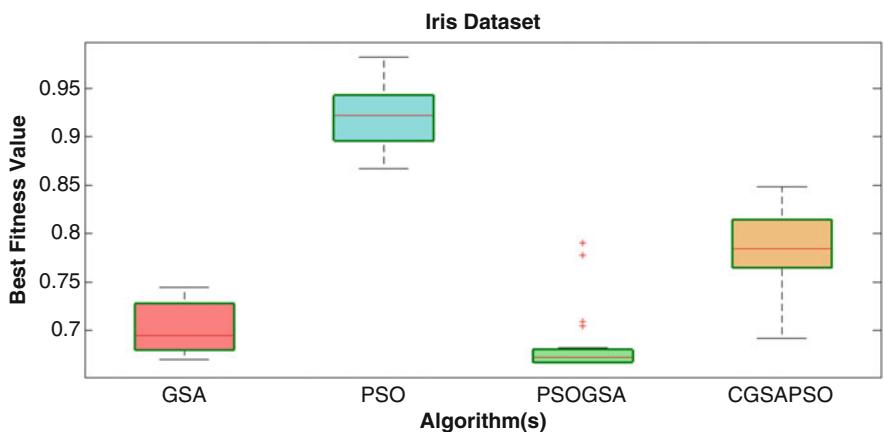


Fig. 13.11 Box plots of Iris dataset

CGSAPSO versions particularly CGSAPSO3 and CGSAPSO6. Besides, CGSAPSO takes less execution time to find global optimum and potential features in classification benchmarks. In addition, it can also be interpreted from the experimental results that PSOGSA has statistically more significant MSE outcomes than competing algorithms particularly PSO and GSA. Quantitatively speaking, the results validate the high intensification and diversification capability of CGSAPSO. On the theoretical side, CGSAPSO has been quite successful in overcoming neighborhood search and exploitation drawbacks of GSA-MLP by providing a high recognition rate and optimal MSE values. However, it is also obvious that CGSAPSO has issues with the convergence speed as it remains behind PSOGSA in Balloon and Iris

datasets. So, proper parameter tuning and enhanced stopping criteria are required to improve CGSAPSO convergence speed.

As far as portfolio optimization (PO) (Markowitz, 1952) is concerned, CGSAPSO can be used to maximize the return and minimize the risk. Actually, PO is a constrained optimization problem in which constraints are related to the financial assets and risk. Therefore, CGSAPSO can be employed to find the optimal portfolio asset(s) for the investor by utilizing the penalty function method to handle the constraints. Moreover, CGSAPSO can also be used in PO for the analysis of financial data in order to predict the future market trend(s). Simply, in place of XOR, Iris, and Balloon datasets (used in the study), financial datasets can be used, and then, CPSOGSA-MLP will be employed for accuracy check and regression analysis.

13.11 Conclusion and Subsequent Prospects

In this chapter, a newly proposed CGSAPSO optimization method has been used for MLP learning. Three classification datasets, namely, XOR, Balloon, and Iris, are utilized for checking the recognition capability of CGSAPSO. In the proposed CGSAPSO algorithm, diversification power is provided by CGSA, while exploitation is carried out through PSO. The simulation results on three classification datasets indicate that CGSAPSO provides minimum values for MSE and high classification accuracy as compared to standard GSA and PSO. Moreover, PSOGSA also displayed efficient convergence power and simulation accuracy as far as Balloon and Iris datasets are concerned.

In the future, CGSAPSO will be hybridized with other versatile neural networks including deep neural networks, LSTM, and radial basis function networks. Besides, a binary version of CGSAPSO can be applied to feature selection problems. In addition, CGSAPSO can be employed for the analysis of medical datasets related to the cancer diagnosis and the brain tumor detection.

Key Terms and Definitions

Chaos theory: Chaos theory is the branch of mathematics dealing with the systems that are in some kind of motion. These systems are highly sensitive to the changes in the initial conditions.

Hybridization: It is a mathematical technique in which two or more algorithms are combined to solve an optimization problem.

MLP (multi-layer perceptron): MLP is a feedforward neural network consisting of an input layer, one or more hidden layer(s), and an output layer. It uses sigmoid function as a fitness function to find a suitable candidate solutions. Moreover, it is utilized for the classification of data in nonlinear systems.

Optimization: It is an iterative technique in which the most suitable candidate solution is selected at the end of the process.

References

- Altay, E. V., & Alatas, B. (2020). Bird swarm algorithms with chaotic mapping. *Artificial Intelligence Review*, 53(2), 1373–1414. <https://doi.org/10.1007/s10462-019-09704-9>.
- Bebis, G., & Georgopoulos, M. (1994). Feed-forward neural networks. *IEEE Potentials*, 13(4), 27–31.
- Blake, C., & Merz, C. J. (1998). *UCI: Repository of machine learning databases*. Retrieved from <http://www.ics.uci.edu/~mlearn/MLRepository.html>.
- Derrac, J., García, S., Molina, D., & Herrera, F. (2011). A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation*, 1(1), 3–18.
- Dorffner, G. (1996). Neural networks for time series processing. *Neural Network World*, 6, 447–468.
- Dorigo, M., Birattari, M., & Stutzle, T. (2006). Ant colony optimization. *IEEE Computational Intelligence Magazine*, 1(4), 28–39. <https://doi.org/10.1109/MCI.2006.329691>.
- Gandomi, A. H., & Yang, X. S. (2014). Chaotic bat algorithm. *Journal of Computational Science*, 5 (2), 224–232. <https://doi.org/10.1016/j.jocs.2013.10.002>.
- Green, R. C., II, Wang, L., & Alam, M. (2012). Training neural networks using central force optimization and particle swarm optimization: Insights and comparisons. *Expert Systems with Applications*, 39(1), 555–563.
- Gudise, V. G., & Venayagamoorthy, G. K. (2003). Comparison of particle swarm optimization and backpropagation as training algorithms for neural networks. In *Proceedings of the 2003 IEEE Swarm Intelligence Symposium*, pp. 110–117.
- Halliday, D., Resnick, R., & Walker, J. (2000). *Fundamentals of physics (6th Edition)*. Delhi: Wiley.
- Huang, M. L., & Chou, Y. C. (2019). Combining a gravitational search algorithm, particle swarm optimization, and fuzzy rules to improve the classification performance of a feed-forward neural network. *Computer Methods and Programs in Biomedicine*, 180, 105–116. <https://doi.org/10.1016/j.cmpb.2019>.
- Hussain, K., Salleh, M. N. M., Cheng, S., & Shi, Y. (2019). On the exploration and exploitation in popular swarm-based metaheuristic algorithms. *Neural Computing and Applications*, 31(11), 7665–7683. <https://doi.org/10.1007/s00521-018-3592-0>.
- Itano, F., DeSousa, M. A. D. A., & Hernandez, E. D. M. (2018). Extending MLP ANN hyperparameters optimization by using genetic algorithm. In *Proceedings of the IEEE 2018 International Joint Conference on Neural Networks (IJCNN)*, pp. 1-8.
- Jacobs, R. A. (1988). Increased rates of convergence through learning rate adaptation. *Neural Networks*, 1(4), 295–307.
- James, J. Q., Lam, A. Y., & Li, V. O. (2011, June). Evolutionary artificial neural network based on chemical reaction optimization. In *2011 IEEE congress of evolutionary computation (CEC)* (pp. 2083–2090). New York: IEEE.
- Karaboga, D., Akay, B., & Ozturk, C. (2007). Artificial bee colony (ABC) optimization algorithm for training feed-forward neural networks. In *Springer international conference on modeling decisions for artificial intelligence* (pp. 318–329). Berlin: Springer.
- Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. In *Proceedings of ICNN '95-IEEE International conference on neural networks* (pp. 1942–1948). New York: IEEE.
- Khishe, M., & Mosavi, M. R. (2020). Classification of underwater acoustical dataset using neural network trained by chimp optimization algorithm. *Applied Acoustics*, 157, 107005. <https://doi.org/10.1016/j.apacoust.2019.107005>.
- Llonen, J., Kamarainen, J. K., & Lampinen, J. (2003). Differential evolution training algorithm for feed-forward neural networks. *Neural Processing Letters*, 17(1), 93–105.
- Markowitz, H. (1952). Portfolio selection. *Journal of Finance*, 79(1), 77–91.
- McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5(4), 115–133.

- Mejía-de-Dios, J. A., & Mezura-Montes, E. (2019). A new evolutionary optimization method based on center of mass. *Decision Science in Action, 2019*, 65–74. https://doi.org/10.1007/978-981-13-0860-4_6.
- Mendes, R., Cortez, P., Rocha, M., & Neves, J. (2002). Particle swarms for feed-forward neural network training. *Proceedings of the IEEE International Joint Conference on Neural Networks, 6*, 1895–1899.
- Mirjalili, S. (2015). How effective is the Grey Wolf optimizer in training multi-layer perceptrons. *Applied Intelligence, 43*(1), 150–161. <https://doi.org/10.1007/s10489-014-0645-7>.
- Mirjalili, S., & Gandomi, A. H. (2017). Chaotic gravitational constants for the gravitational search algorithm. *Applied Soft Computing, 53*, 407–419. <https://doi.org/10.1016/j.asoc.2017.01.008>.
- Mirjalili, S., & Hashim, S. Z. M. (2010). A new hybrid PSOGSA algorithm for function optimization. In *2010 IEEE International Conference on Computer and Information Application, 2010*, 374–377.
- Mirjalili, S., & Lewis, A. (2016). The whale optimization algorithm. *Advances in Engineering Software, 95*, 51–67. <https://doi.org/10.1016/j.advengsoft.2016.01.008>.
- Mirjalili, S., Mirjalili, S. M., & Lewis, A. (2014). Let a biogeography-based optimizer train your multi-layer perceptron. *Information Sciences, 269*, 188–209. <https://doi.org/10.1016/j.ins.2014.01.038>.
- Mirjalili, S., Mirjalili, S. M., & Yang, X. S. (2014). Binary bat algorithm. *Neural Computing and Applications, 25*(3-4), 663–681. <https://doi.org/10.1007/s00521-013-1525-5>.
- Mirjalili, S., Mohd Hashim, S. Z., & Moradian Sardroodi, H. (2012). Training Feed-forward neural networks using hybrid particle swarm optimization and gravitational search algorithm. *Applied Mathematics and Computation, 218*(22), 11125–11137.
- Mirjalili, S. M., Abedi, K., & Mirjalili, S. (2013). Optical buffer performance enhancement using particle swarm optimization in ring-shape-hole photonic crystal waveguide. *The Optician, 124* (23), 5989–5993.
- Nawi, N. M., Ransing, R. S., Salleh, M. N. M., Ghazali, R., & Hamid, N. A. (2010). An improved back propagation neural network algorithm on classification problems. In *Database theory and application, bio-science and bio-technology* (pp. 177–188). https://doi.org/10.1007/978-3-642-17622-7_18.
- Ooyen, A., & Nienhuis, B. (1992). Improving the convergence of the backpropagation algorithm. *Neural Networks, 5*(3), 465–471.
- Pereira, L. A., Afonso, L. C., Papa, J. P., Vale, Z. A., Ramos, C. C., Gastaldello, D. S., & Souza, A. N. (2013, April). Multilayer perceptron neural networks training through charged system search and its application for non-technical losses detection. In *2013 IEEE PES Conference on Innovative Smart Grid Technologies (ISGT Latin America)* (pp. 1–6). New York: IEEE.
- Pereira, L. A., Rodrigues, D., Ribeiro, P. B., Papa, J. P., & Weber, S. A. (2014, May). Social-spider optimization-based artificial neural networks training and its applications for Parkinson's disease identification. In *2014 IEEE 27th international symposium on computer-based medical systems* (pp. 14–17). New York: IEEE.
- Rashedi, E., Nezamabadi-pour, H., & Saryazdi, S. (2009). GSA: A gravitational search algorithm. *Information Sciences, 179*(13), 2232–2248.
- Rather, S. A., & Bala, P. S. (2019a). A holistic review on gravitational search algorithm and its hybridization with other algorithms. In *2019 IEEE International conference on electrical, computer and communication technologies (ICECCT)* (pp. 1–6). New York: IEEE. <https://doi.org/10.1109/ICECCT.2019.8869279>.
- Rather, S. A., & Bala, P. S. (2019b). Analysis of gravitation based optimization algorithms for clustering and classification. In *Handbook of research on big data clustering and machine learning* (pp. 77–99). Hershey: IGI Global. <https://doi.org/10.4018/978-1-7998-0106-1.ch005>.
- Rather, S. A., & Bala, P. S. (2019c). Hybridization of constriction coefficient based particle swarm optimization and gravitational search algorithm for function optimization. In *2019 Elsevier International Conference on Advances in Electronics, Electrical, and Computational Intelligence (ICAEEC-2019)*. Amsterdam: Elsevier. <https://doi.org/10.2139/ssrn.3576489>.

- Rather, S. A., & Bala, P. S. (2020a). A hybrid constriction coefficient based particle swarm optimization and gravitational search algorithm for training multi-layer perceptron (MLP). *International Journal of Intelligent Computing and Cybernetics*, 13(2), 129–165. <https://doi.org/10.1108/JICC-09-2019-0105>.
- Rather, S. A., & Bala, P. S. (2020b). Swarm-based chaotic gravitational search algorithm for solving mechanical engineering design problems. *World Journal of Engineering*, 17(1), 97–114. <https://doi.org/10.1108/WJE-09-2019-0254>.
- Rather, S. A., & Sharma, N. (2017). GSA-BBO hybridization algorithm. *International Journal of Advance Research in Science and Engineering*, 6, 596–608.
- Saremi, S., Mirjalili, S., & Lewis, A. (2014). Biogeography-based optimization with chaos. *Neural Computing and Applications*, 25(5), 1077–1097.
- Simon, D. (2008). Biogeography-based optimization. *IEEE Transactions on Evolutionary Computation*, 12, 702–713.
- Weir, M. K. (1991). A method doe self-determination of adaptive learning rates in backpropagation. *Neural Networks*, 4(3), 371–379.
- Wilcoxon, F. (1945). Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6), 80–83.
- Yang, X., & Gandomi, A. H. (2012). Bat algorithm: A novel approach for global engineering optimization. *Engineering Computations*, 29(5), 464–483. <https://doi.org/10.1108/02644401211235834>.

Additional Reading

- Clerc, M., & Kennedy, J. (2002). The particle swarm-explosion, stability, and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation*, 6(1), 58–73.
- Kalinlia, A., & Karabogab, N. (2005). Artificial immune algorithm for IIR filter design Engineering. *Applications of Artificial Intelligence*, 18, 919–929.
- Kirkpatrick, S., Gelatto, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 220, 671–680.
- Lavika, S., Sharthak, & Satyajit. (2016). Hybridization of gravitational search algorithm and biogeography based optimization and its application on grid scheduling problem. *Ninth International Conference on Contemporary Computing (IC3), 2016*, 1–6.
- Li, C., & Zhou, J. (2011). Parameters identification of hydraulic turbine governing system using improved gravitational search algorithm. *Energy Conversion and Management*, 52, 374–381.
- Svozil, D., Kvasnicka, V., & Pospichal, J. (1997). Introduction to multi-layer feed-forward neural networks. *Chemometrics and Intelligent Laboratory Systems*, 39(1), 43–62. [https://doi.org/10.1016/S0169-7439\(97\)00061-0](https://doi.org/10.1016/S0169-7439(97)00061-0).
- Tang, K. S., Man, K. F., Kwong, S., & He, Q. (1996). Genetic algorithms and their applications. *IEEE Signal Processing Magazine*, 13, 22–37.
- Wolpert, D. H., & Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1), 67–82.
- Yang, L. J., & Chen, T. L. (2002). Application of chaos in genetic algorithms. *Communications on Theoretical Physics*, 38(2), 168–172.
- Walczak, S., & Cerpa, N. (1999). Heuristic principles for the design of artificial neural networks. *Information and Software Technology*, 41(2), 107–117. [https://doi.org/10.1016/S0950-5849\(98\)00116-5](https://doi.org/10.1016/S0950-5849(98)00116-5).

Chapter 14

Solving Optimization Problem with Particle Swarm Optimization: Solving Hybrid Flow Shop Scheduling Problem with Particle Swarm Optimization Algorithm



Fatma Selen Madenoglu

Abstract The flow shop scheduling problem is widely discussed in the literature since it is frequently applied in real industry. This paper presents a variant of flow shop scheduling problem with parallel machines. The proposed problem includes multistage and identical parallel machines at each stage, and the sequence-dependent setup time and transportation time are considered. The objective function is minimization of makespan. The particle swarm optimization algorithm (PSO) is addressed to solve the problem and compared with genetic algorithm and heuristics. The benchmark instances are generated to demonstrate the performance of the PSO. The numerical results show that the PSO significantly outperforms the comparison set.

Keywords Hybrid flow shop · Combinatorial optimization · Makespan · Particle swarm optimization

14.1 Introduction

Due to the increasing use of automation in various sectors, the importance and function of the production scheduling problem has gradually increased. Organizing machinery and equipment with the use of automation reveals the need to optimize the data in the production environment. With increasing problem complexity, it is dealt with by using metaheuristics.

Production scheduling is an important tool for firms to gain the competitive advantage in the global market. The different scheduling problem has been discussed in the literature. One of them is the flow shop scheduling problem (FSP). There is a

F. S. Madenoglu (✉)
Abdullah Gül University, Kayseri, Turkey
e-mail: selen.madenoglu@agu.edu.tr

set of jobs $N = \{1, 2, \dots, n\}$ that are processed in the same order at a set of machines $M = \{1, 2, \dots, m\}$, i.e., one job goes to machine 1, then goes to machine 2, . . . up to machine m . The job is processed on at most one machine, and one machine processes at most only one machine. The most common objective function for this problem is minimizing total completion time (makespan). The problem includes the determination of job sequence according to the performance measure(s). Johnson (1954) introduces the academic research about solving the FSP firstly. Johnson addresses an exact algorithm for two-machine FSP with makespan. With the increase in the number of machines and job, the FSP has been included in the combinatorial optimization problem.

Nowadays, to provide a competitive advantage, to increase current production capacities, to meet customer orders, and to follow existing technologies, parallel machines are used at some of the stages in the flow shop scheduling problem. The problem turns into a hybrid flow shop scheduling problem (HFSP). There are alternative machines where operations can be processed in at least one of the stages. The handled problem is more complex than the FSP. Both assignment and sequencing of jobs in each stage are addressed in the HFSP.

The HFSP is widely used in various sectors such as steel (Pan et al., 2012), label sticker manufacturing company (Lin & Liao, 2003), semiconductor industry (Quadt & Kuhn, 2005), ceramic tiles (Ruiz & Maroto, 2006), textile (Grabowski & Pempera, 2000), and electronics (Jin, Ohno, Ito, & Elmaghraby, 2002). The high throughput number and resource utilization are important for real-life application. For that reason, the objective is to obtain a feasible schedule in which the completion time of all jobs will be minimal in this paper.

One of the special situations that we encounter on the shop floor where machines can operate more than one job is setup time. The setup time is not included in the processing time and is sometimes ignored in literature. In this paper, sequence-dependent setup time is dealt with separately in process times. A HFSP with sequence-dependent setup time is a more challenging and difficult class of scheduling problems (Pinedo, 2012).

In addition to the setup times, the transportation time is another important factor to be considered. As with the setup times, the transportation times are either included in the processing times by taking an average transportation time or are not evaluated at all. It is aimed to make the application more realistic by considering the transportation time between stages.

In this study, a PSO and a genetic algorithm (GA) is proposed to solve the HFSP considering setup times and transportation times with makespan minimization. PSO is a population-based algorithm. Structural simplicity, implementation ease, and rapid speed of acquiring solution are some advantages of the PSO (El-Ghazali, 2009). The generated benchmark instances are used to validate the performance of the PSO.

14.2 Background

The simple FSP has two stages ($g = 2$) and one machine at each stage and is a nondeterministic polynomial time-hard (NP-hard) problem. The extended HFSP with setup times and transportation times to be addressed in the study is in the NP-hard problem class. The solution approaches in which a feasible solution is obtained at an acceptable time are preferred to tackle the complexity of the problem. Ribas, Leisten, and Framiñan (2010) and Ruiz and Vázquez-Rodríguez (2010) present the literature reviews on HFSP. They point out that the makespan is the most used performance measurement for the HFSP.

Portmann, Vignier, Dardilhac, and Dezelay (1998), Rajendran and Chaudhuri (1992), and Morita and Shio (2005) propose branch and bound (B&B) algorithms as an exact solution method for the HFSP with makespan minimization. Vignier, Dardilhac, Dezelay, and Proust (1996) present another B&B algorithm to solve the HFSP with total completion time. In the literature, the ordinary HFSP is discussed. An approximate solution is presented by Gupta (1988) for one machine and two-stage HFSP. Metaheuristics are developed for the HFSP, such as artificial immune system (Engin & Döyen, 2004), an improved ant colony algorithm (Alaykyran, Engin, & Döyen, 2007), IG algorithm (Kizilay, Tasgetiren, Pan, & Wang, 2014; Öztop, Tasgetiren, Eliyi, & Pan, 2019), genetic algorithm (Kahraman, Engin, Kaya, & Kerim Yilmaz, 2008), tabu search (Negenman, 2001), simulated annealing (Jin et al., 2002, 2006), PSO (Liao, Tjandradjaja, & Chung, 2012), improved particle swarm optimization (Marichelvam, Geetha, & Tosun, 2020), and discrete artificial bee colony algorithm (Pan et al., 2014). Genetic algorithm (Gholami, Zandieh, & Alem-Tabriz, 2009; Mousavi, Mahdavi, Rezaeian, & Zandieh, 2018), iterated local search (Naderi, Ruiz, & Zandieh, 2010; Naderi, Zandieh, Balagh, & Roshanaei, 2009), improved simulated annealing algorithm (Mirsanei, Zandieh, Moayed, & Khabbazi, 2011), water flow-like algorithm (Pargar & Zandieh, 2012), and hybrid metaheuristics (Behnamian & Zandieh, 2013; Khare & Agrawal, 2019; Pargar, Zandieh, Kauppila, & Kujala, 2018; Shahvari & Logendran, 2018) are proposed for HSP with setup times.

Lin and Liao (2003) present a production environment which is a two-stage HFSP with setup time and dedicated machines. They consider the weighted maximal tardiness as an objective function. Babayan and He (2004) develop an agent-based solution method to solve the three-stage HFSP with identical machines considering makespan minimization. Engin and Döyen (2004) develop an artificial immune system algorithm (AIS) for solving the HFSP with minimizing makespan. They compose a procedure to determine the optimum parameter set of AIS. Jin et al. (2006) present two metaheuristics for solving the HFSP considering parallel identical machines. They construct the optimization method using simulated annealing and variable depth search. Janiak, Kozan, Lichtenstein, and Oğuz (2007) propose some approximation metaheuristics for solving the HFSP with multi-objective (the total weighted earliness, the total weighted tardiness, the weighted waiting time). Zandieh and Gholami (2009) introduce the HFSP with setup times and machine

breakdowns, and they tackle this problem by using an immune algorithm. Kahraman et al. (2010) apply parallel greedy algorithm for the HFSP with multistages. Khalouli, Ghedjati, and Hamzaoui (2010) study the HFSP with the minimization of the weighted earliness tardiness. They propose an ant colony optimization to cope with this problem. Ruiz, Şerifoğlu, and Urlings (2008) investigate complex and realistic HFSP. They solve the HFSP with release date, unrelated machines, machine eligibility, skipped stages, setup times, and precedence relationship by using a mixed-integer programming mathematical model and heuristics. İşler, Toklu, and Çelik (2012) address two-machine FSP under learning effects to minimize total earliness and tardiness. They proposed three solution approaches based on genetic algorithm, based on tabu search, and based on random search. The computational results show that genetic algorithm is better than tabu search and random search. Marichelvam et al. (2013) propose a bat algorithm to deal with the HFSP. Su, Yu, Wu, and Tian (2014) introduce a distributed coevolutionary algorithm for solving multi-objective HFSP considering minimizing completion time and total tardiness. Wang, Wang, Liu, and Xu (2013) and Wang, Wang, and Yu (2020) solve the HFSP with identical parallel machines under minimization makespan by using an enhanced estimation of distribution algorithm. Chung and Liao (2013) propose an immunoglobulin-based artificial immune system for the HFSP. Bożejko, Pempera, and Smutnicki (2013) present a parallel tabu search algorithm to solve the HFSP. Pan and Dong (2014) develop a novel migrating bird optimization for the HFSP to minimize total flowtime. The large-size production problems reflect the real-life production environment. Li and Pan (2015) solve the HFP with limited buffer by combining the tabu search and the artificial bee colony. They generate large-scale benchmark instances to conduct the experimental study. Wang et al. (2020) develop a branch and bound approach, a tabu search, and three heuristics for two-stage no-wait HFSP with setup times. Pan et al. (2017) introduce iterated search methods for solving the HFSP with due windows. Dios, Fernandez-Viagas, and Framinan (2018) propose efficient heuristics for the HFSP with missing operations. Fernandez-Viagas, Molina-Pariente, and Framinan (2018) develop two constructive heuristics to solve the HFSP with minimization makespan.

Naderi et al. (2009) develop a simulated annealing algorithm to handle the HFSP with sequence-dependent setup times and transportation times under total completion time and total tardiness criterion. Moccellin, Nagano, Neto, and de Athayde Prata (2018) investigate heuristic algorithms with priority rules for solving the HFS problem with machine blocking and setup times. Hidri, Elkasantini, and Mabkhot (2018) study exact procedures and a two phase's heuristic to solve the two-center HFS problem with transportation times. Madenoglu (2019) presents heuristics for the hybrid flow shop problem considering missing operations, transportation times, and sequence-dependent setup times.

The particle swarm optimization algorithm was introduced in 1995 by Kennedy and Eberhart for solving the nonlinear programming and constrained optimization. PSO has been adapted to solve various optimization problems. Sha and Hsu (2006) present the hybrid PSO for the HFSP. Pan et al. (2008) use a discrete PSO to solve no-wait FSP. Tseng and Liao (2008a, 2008b) propose the HFSP with multiprocessor

tasks by using PSO and propose the improved discrete PSO for lot streaming FSP. Zhang, Shao, Li, and Gao (2009) combine the PSO and tabu search algorithm for multi-objective HFSP. Zhang, Ning, and Ouyang (2010) combine the PSO with genetic operators and annealing strategy for the FSP. In the proposed algorithm, each particle includes two states to improve the performance of the proposed solution approach. The computational results show that the proposed algorithm outperforms the others. Liao et al. (2012) combine PSO with bottleneck heuristic and simulated annealing to solve the HFSP in an effective manner. The performance of the proposed algorithm is tested by using the well-known benchmark instance. The computational results show that the performance of the proposed algorithm is better than the other existing algorithms in the literature. Liu, Gao, and Pan (2011) address permutation FSP and use a hybrid PSO to solve this problem. Liao et al. (2012) hybridize PSO with bottleneck heuristic for exploiting the bottleneck and simulated annealing for escaping from local optima. Shao, Liu, Liu, and Zhang (2013) address the HFSP with minimization of makespan, maximal machine workload, and total workload. They proposed the hybrid discrete PSO. Chou (2013) address the HFSP with multiprocessor tasks by using PSO. Akhshabi, Tavakkoli-Moghaddam, and Rahnamay-Roodposhti (2014) propose a hybrid PSO based on memetic algorithm to solve no-wait FSP. Marichelvam et al. (2020) present an improved PSO for the HFSP with human factors. They integrate the variable neighborhood search algorithm into the PSO to improve the convergence speed of the proposed algorithm.

The related literature review provides that the HFSP considering sequence-dependent setup times and transportation times under makespan criterion has not been discussed widely. This paper purposes to fill this research gap by presenting PSO and results for the discussed problem.

14.3 Problem Definition

The problem presented in this paper is defined as follows: There is n jobs ($N = \{1, 2, \dots, n\}$) that are processed on a series of k stages ($K = \{2, \dots, k\}$) in a multistage flow shop environment where stage k contains m_k parallel machines ($M_k = \{1, 2, \dots, m_k\}$). The number of stages k must be at least two. The number of parallel machines m_k must be greater than and equal to one in at least one of the k stages. Every job completes the production process by going through the same multiple stages (k): stage 1, stage 2, ..., stage k . The parallel machines in a stage are identical. A job is processed by any machine m_k at stage k . The processing time of m_k machines for a job in stage k is the same. Each operation of the job is assigned to one machine in each stage. The storages between consecutive stages are unlimited. No interruption is allowed. The sequence-dependent setup time is handled. While one operation is processed on the machine, the setup operation is carried out according to the sequence-dependent setup time. After an operation processed on the machine, this job is moved from one stage to another stage to process. Transportation time between consecutive stages is considered. All machines and jobs are available at

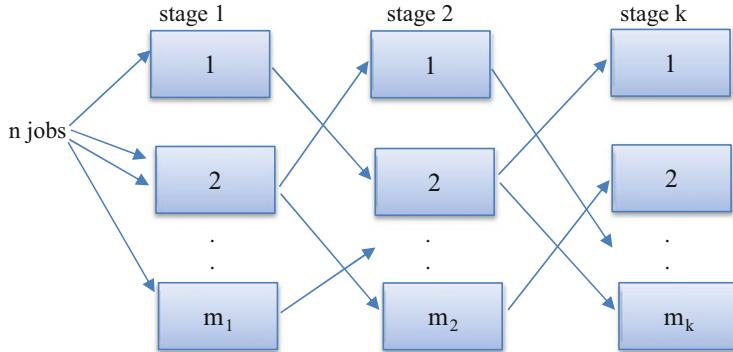


Fig. 14.1 HFSP shop environment. Source: Author's own creation

zero time. All machines are available if they are not busy. Each machine operates one operation at a time. There are no machine breakdowns and maintenance operations. In the HFSP, the aim is to minimize the maximum completion time. The problem is finding a schedule which optimizes a stated performance measure(s). Figure 14.1 illustrates the HFSP shop environment.

14.4 Particle Swarm Optimization Algorithm

The particle swarm optimization algorithm is a population-based algorithm inspired by the social behavior of animals living in swarms, such as flocks of birds. It is widely used in many areas because of fast computing speed and the parallel processing. It is a population-based algorithm which is preferred to solve complex scheduling problems. A population of particles initializes the PSO. A candidate solution is represented by each of the particles, and each particle has its own position $x_i(t)$ and velocity $v_i(t)$ in the search space. A population is generated randomly or based on some rules. Then, every particle updates itself based on the updating rule in each iteration until the stopping criteria is met. The output is the final best solution. In a PSO, the velocity and position of particle in $(t + 1)$ h iteration are updated by:

$$\begin{aligned} v_i(t + 1) &= w * v_i(t) + c_1 * [x_i^*(t) - x_i(t)] \\ &\quad + c_2 * [x_g^*(t) - x_i(t)] \end{aligned} \quad (14.1)$$

$$x_i(t + 1) = x_i(t) + v_i(t + 1) \quad (14.2)$$

where w is the inertia weight and c_1 and c_2 are random numbers between 0 and 1. $x_i^*(t)$ is the local best value. $x_g^*(t)$ is the global best value. The general steps in the simple PSO algorithm are shown below:

Step 1: Set the swarm size, termination criteria, inertia weight, c_1 , and c_2 .

PSO ALGORITHM

```

generate initial population randomly
for (t=1 to stopping criteria)
    for each particle jεJ
        update velocity ( $v_j(t)$ ) and position ( $x_j(t)$ )
        update local best ( $x_j^*(t)$ ) and global best ( $x_g^*(t)$ )
    end for
end for

```

Fig. 14.2 PSO procedure

Step 2: Determine the fitness function.

Step 3: Generate the initial population.

Step 4: Find the local best position of the particles.

Step 5: Initialize the velocity of the particles.

Step 6: Evaluate the objection function value for each particle.

Step 7: Specify the global best position of the particles.

Step 5: Update the position and velocity of the particles.

Step 6: Check the termination criteria is met, stop; otherwise return to step 4.

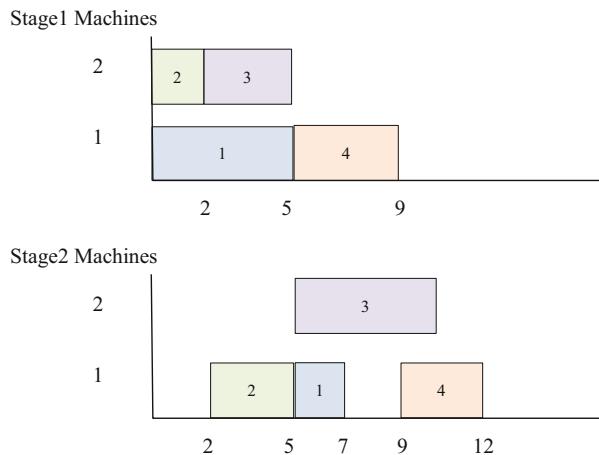
In this PSO, the initial population is generated randomly. Then, the position and velocity of the particle are modified by Eqs. (14.1) and (14.2) in each iteration. The algorithm runs until the iteration number has reached the stopping criteria. PSO procedure is defined in Fig. 14.2.

In the proposed algorithm, the forward scheduling approach, which is common in the relevant literature, is used to evaluate the objective function. In this approach, the jobs are assigned to the machines starting from the most available machine in the first production stage. In the following production stages, the jobs are assigned to the most available machine, depending on the release times from the previous production stage. The order of the jobs may be different at each production stage. In this way, the initial solution (s) is obtained by assigning the jobs to the machines step by step.

To illustrate, the approach applied with an example has four jobs and two production stages. The first stage processing times of jobs as 5, 2, 3, 4 and the second stage processing times of jobs as 2, 3, 5, 3 and initial solution is $s = \{1, 2, 3, 4\}$ are used to schedule the jobs. The most available machine is selected to process the jobs at the first production stage. At the second production stages, the jobs are assigned to the most available machines according to the releasing times. The job order turns into $\{2, 1, 3, 4\}$ at the second stage. Figure 14.3 demonstrates the Gantt chart for initial schedule with a makespan of 12.

The constructive heuristic for solving the flow shop scheduling problem is the heuristic of Nawaz, Enscore, and Ham (NEH). Nawaz, Enscore Jr, and Ham (1983) suggest that a job with a longer total processing time should have a higher priority in the series to make the job easier. NEH is very effective in minimizing time to complete (Taillard, 1990). In the first step, the jobs are sorted in decreasing order

Fig. 14.3 Gantt chart for the initial schedule. Source: Author's own creation



of total processing times. In the second step, two possible partial schedules are constructed by using the first two jobs and evaluated to select the best sequence. In the third step, the following job which is stated in the first step is replaced in all possible positions in the partial schedule and evaluated. Until all jobs are added to the partial schedule, this step is repeated. Finally, the final job sequence is the obtained schedule that is employed to the shop floor.

One of the common dispatching rules to arrange the jobs is shortest processing time rule (SPT). The jobs are sorted in ascending order of the processing time. The other is longest processing time rule (LPT) which sequences the jobs in descending order of processing time. In this paper, the results of NEH algorithm, dispatching rules (SPT and LPT), and genetic algorithm are used as a comparative set.

14.5 Computational Experiments

In this paper, the computational experiments are carried out to the performance of the heuristics and metaheuristics stated above. All computational experiments have been run on Intel Core i7-4700 CPU with 2.40 GHz, 16 GB RAM, and with Microsoft Windows 10 64 bits. MATLAB has been used to code all heuristics and algorithms used in this paper.

The benchmark problems are generated to evaluate the performance of the PSO. The processing time of an operation for each job, sequence-dependent setup time between the consecutive operations, and transportation time from one stage to the other stage are randomly generated uniform distribution over [1, 100], [1, 20], and [1, 10], respectively. The different configurations for the number of jobs, number of stages, number of parallel machines in each stage, number of generations, and population size are considered as a factor to investigate the effects of these on the performance of the algorithm.

Table 14.1 Factor levels

Factors	Levels
Number of jobs	20, 50
Number of stages	5, 10
Number of machines in each stage	3, 5
Population size	100, 200
Number of generations	200, 500

Table 14.2 MRPD comparison of different solution approaches for the generated benchmark instances

Solution approaches	MRPD
GA	2.86
PSO	0.15
NEH	4.95
SPT	16.38
LPT	22.63

The generated problem data can be categorized by factors, and each factor can have two levels. These factors and levels are given in Table 14.1. Thirty-two experiments are conducted. The PSO algorithm is compared with genetic algorithm, SPT, LPT, and NEH heuristic. The parameters of GA are similar to the respective paper from literature. The performance measure is relative percentage deviation (RPD) over the best solutions found in the experiment:

$$RPD = \frac{C_{\max} - C_{\max}^{\text{best}}}{C_{\max}^{\text{best}}} \times 100 \quad (14.3)$$

Here RPD refers to the relative percent deviation of the algorithm for a problem, C_{\max} is the completion time of a problem, and C_{\max}^{best} is the best completion time of a problem. The average relative percent deviation value of each metaheuristic and heuristic is calculated, and a comparison is conducted. Each experiment is run 20 times. The mean RPD (MRPD) values are mean value of the RPD values for the solution approaches.

14.6 Solutions and Recommendations

The MRPD values are given in Table 14.2. According to the MRPD values, the result of the PSO algorithm is superior to the GA, NEH, SPT, and LPT. The performance of the proposed algorithm is generally expected to be better than the heuristic. The results show that the PSO algorithm is more preferred than heuristics. Likewise, it is seen that PSO is better in finding a good result when compared with the widely preferred genetic algorithm in the literature.

The statistical tests are performed to compare better the performance of the solution approaches. The nonparametric statistical tests are preferred to analyze the

Table 14.3 Wilcoxon sign test results

Solution approaches	<i>p</i> -Value
PSO vs. GA	0.00
PSO vs. NEH	0.00
PSO vs. SPT	0.00
PSO vs. LPT	0.00

multiple problem. The independence, normality, and heteroscedasticity properties are conducted to check whether to apply or not a parametric statistical test. Independence condition can be given that one occurring does not affect the probability distribution of the other one occurring. Independence condition is satisfied in the experiments. All the tests which are used in this experiment represent abnormality shape. All properties are controlled, and the *p*-value of the experiments are lower than the level of significance ($\alpha = 0.05$). This means that the normality condition is not satisfied. Statistical software package MINITAB is used to conduct all computations. Heteroscedasticity condition can be controlled by applying Levene's test, which checks homogeneity of variances of the different algorithms. Homogeneity condition is not satisfied, because we reject the null hypothesis at a level of significance, $\alpha = 0.05$. The parametric test conditions are not fulfilled perfectly. These results induce us to use nonparametric statistics for analyzing the results. The Friedman test is a nonparametric analog of the parametric two-way analysis of variance. The Friedman test is used to specify differences between two or more solution approaches. The null hypothesis states that all solutions are equivalent; the rejection of the null hypothesis means that the performance of the solution approaches are different. The *p*-value of the test implies the differences among all solution approaches. After that, the Wilcoxon signed-rank test is applied to importance level of two different solution approaches. The Wilcoxon signed-rank test results are given in Table 14.3. According to these comparisons, the PSO outperforms the other algorithms for all test instances.

The results show that PSO can be preferable for solving the HFSP. Compared with GA, PSO has few parameters to calibrate. During the evaluation process, PSO only uses mathematical operators; however GA uses complex genetic operators. GA is slower than PSO. PSO shares the single directional information way, but GA shares the mutual information way. Since the particles tend to follow the best in the PSO algorithm, after a certain period, we can see all the particles clustered in one place looking for the solution. This clustering really means an advantage where the solution is found, as the solution will be reached more quickly.

14.7 Conclusion

The hybrid flow shop scheduling problems have many real-life applications in industry. For that reason, many exact algorithms, heuristics, and metaheuristics have been studied in the literature to solve the HFSP effectively. In this paper, I

address a HFSP which reflects the real-life job floor environment. One of the situations encountered in dynamic production environments, discussed in the literature, is setup times required for the settings that need to be made in the transition between different jobs processed on the machines. In some studies, setup times are included in the operation times, while in others they are considered separately. As it is known, setup times affect the job to be assigned to the machine and affect the quality of the schedule. Another issue is the transportation times between production stages in production environments where there are more than one production stage and there is more than one machine that can operate at each production stage. In today's modern production environment, transportation operations are carried out with the help of automatic systems, automated guided vehicle, or different systems. The evaluation of the transport time contributes to the management of the production environment and the applicability of the obtained schedule to the production environment and to the detection of deviations from the obtained schedule. In the HFSP dealt with in this paper, setup times and transportation times are also evaluated separately. In addition, minimization of the total completion time is used as an objective function. The PSO is proposed to solve the HFSP with setup times and transports times. Experimental studies have been carried out to evaluate the performance of the proposed solution approach. Test problems to be used in experimental studies have been generated. To analyze the quality of the proposed solution approach, the results obtained from the proposed approach are compared with the results of the commonly used dispatching rules that provide fast and quick solutions and the genetic algorithm approaches. The results of the experimental study show that the PSO is highly effective for the proposed problem.

Key Terms and Definitions

Metaheuristic: A problem solution approach that can generate acceptable solutions within an acceptable time for large-scale optimization problems.

Optimization: The collection of transactions that find the best results among the alternative solutions.

Scheduling: Scheduling is the method by which operation is assigned to resources that complete the operation.

References

- Akhshabi, M., Tavakkoli-Moghaddam, R., & Rahnamay-Roodposhti, F. (2014). A hybrid particle swarm optimization algorithm for a no-wait flow shop scheduling problem with the total flow time. *The International Journal of Advanced Manufacturing Technology*, 70(5–8), 1181–1188.
- Alaykýran, K., Engin, O., & Döyen, A. (2007). Using ant colony optimization to solve hybrid flow shop scheduling problems. *The International Journal of Advanced Manufacturing Technology*, 35(5–6), 541550.
- Babayan, A., & He, D. (2004). Solving the n-job 3-stage flexible flowshop scheduling problem using an agent-based approach. *International Journal of Production Research*, 42(4), 777–799.

- Behnamian, J., & Zandieh, M. (2013). Earliness and tardiness minimizing on a realistic hybrid flowshop scheduling with learning effect by advanced metaheuristic. *Arabian Journal for Science and Engineering*, 38(5), 1229–1242.
- Bożejko, W., Pempera, J., & Smutnicki, C. (2013). Parallel tabu search algorithm for the hybrid flow shop problem. *Computers & Industrial Engineering*, 65(3), 466–474.
- Chou, F. D. (2013). Particle swarm optimization with cocktail decoding method for hybrid flow shop scheduling problems with multiprocessor tasks. *International Journal of Production Economics*, 141(1), 137–145.
- Chung, T. P., & Liao, C. J. (2013). An immunoglobulin-based artificial immune system for solving the hybrid flow shop problem. *Applied Soft Computing*, 13(8), 3729–3736.
- Dios, M., Fernandez-Viagas, V., & Framinan, J. M. (2018). Efficient heuristics for the hybrid flow shop scheduling problem with missing operations. *Computers & Industrial Engineering*, 115, 88–99.
- El-Ghazali, T. (2009). *Metaheuristics: From design to implementation* (Vol. 9, pp. 10–11). Chichester: Jonh Wiley and Sons Inc..
- Engin, O., & Döyen, A. (2004). A new approach to solve hybrid flow shop scheduling problems by artificial immune system. *Future Generation Computer Systems*, 20(6), 1083–1095.
- Fernandez-Viagas, V., Molina-Pariente, J. M., & Framinan, J. M. (2018). New efficient constructive heuristics for the hybrid flowshop to minimise makespan: A computational evaluation of heuristics. *Expert Systems with Applications*, 114, 345–356.
- Gholami, M., Zandieh, M., & Alem-Tabriz, A. (2009). Scheduling hybrid flow shop with sequence-dependent setup times and machines with random breakdowns. *The International Journal of Advanced Manufacturing Technology*, 42(1–2), 189–201.
- Grabowski, J., & Pempera, J. (2000). Sequencing of jobs in some production system. *European Journal of Operational Research*, 125(3), 535–550.
- Gupta, J. N. (1988). Two-stage, hybrid flowshop scheduling problem. *Journal of the Operational Research Society*, 39(4), 359–364.
- Hidri, L., Elkossantini, S., & Mabkhot, M. M. (2018). Exact and heuristic procedures for the two-center hybrid flow shop scheduling problem with transportation times. *IEEE Access*, 6, 21788–21801.
- İşler, M. C., Toklu, B., & Çelik, V. (2012). Scheduling in a two-machine flow-shop for earliness/tardiness under learning effect. *The International Journal of Advanced Manufacturing Technology*, 61(9–12), 1129–1137.
- Janiak, A., Kozan, E., Lichtenstein, M., & Oğuz, C. (2007). Metaheuristic approaches to the hybrid flow shop scheduling problem with a cost-related criterion. *International Journal of Production Economics*, 105(2), 407–424.
- Jin, Z., Yang, Z., & Ito, T. (2006). Metaheuristic algorithms for the multistage hybrid flowshop scheduling problem. *International Journal of Production Economics*, 100(2), 322–334.
- Jin, Z. H., Ohno, K., Ito, T., & Elmaghraby, S. E. (2002). Scheduling hybrid flowshops in printed circuit board assembly lines. *Production and Operations Management*, 11(2), 216–230.
- Johnson, S. M. (1954). Optimal two-and three-stage production schedules with setup times included. *Naval Research Logistics Quarterly*, 1(1), 61–68.
- Kahraman, C., Engin, O., Kaya, İ., & Kerim Yilmaz, M. (2008). An application of effective genetic algorithms for solving hybrid flow shop scheduling problems. *International Journal of Computational Intelligence Systems*, 1(2), 134–147.
- Kahraman, C., Engin, O., Kaya, İ., & Öztürk, R. E. (2010). Multiprocessor task scheduling in multistage hybrid flow-shops: A parallel greedy algorithm approach. *Applied Soft Computing*, 10(4), 1293–1300.
- Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. In *Proceedings of IEEE international conference on neural networks* (pp. 1942–1948). New York: IEEE.
- Khalouli, S., Ghedjati, F., & Hamzaoui, A. (2010). A meta-heuristic approach to solve a JIT scheduling problem in hybrid flow shop. *Engineering Applications of Artificial Intelligence*, 23(5), 765–771.

- Khare, A., & Agrawal, S. (2019). Scheduling hybrid flowshop with sequence-dependent setup times and due windows to minimize total weighted earliness and tardiness. *Computers & Industrial Engineering*, 135, 780–792.
- Kizilay, D., Tasgetiren, M. F., Pan, Q. K., & Wang, L. (2014). An iterated greedy algorithm for the hybrid flowshop problem with makespan criterion. In *2014 IEEE symposium on computational intelligence in production and logistics systems* (pp. 16–23). New York: IEEE.
- Li, J. Q., & Pan, Q. K. (2015). Solving the large-scale hybrid flow shop scheduling problem with limited buffers by a hybrid artificial bee colony algorithm. *Information Sciences*, 316, 487–502.
- Liao, C. J., Tjandradjaja, E., & Chung, T. P. (2012). An approach using particle swarm optimization and bottleneck heuristic to solve hybrid flow shop scheduling problem. *Applied Soft Computing*, 12(6), 1755–1764.
- Lin, H. T., & Liao, C. J. (2003). A case study in a two-stage hybrid flow shop with setup time and dedicated machines. *International Journal of Production Economics*, 86(2), 133–143.
- Liu, H., Gao, L., & Pan, Q. (2011). A hybrid particle swarm optimization with estimation of distribution algorithm for solving permutation flowshop scheduling problem. *Expert Systems with Applications*, 38(4), 4348–4360.
- Madenoglu, F. S. (2019). Solving the hybrid flow shop scheduling problem using heuristic algorithms. *Business & Management Studies: An International Journal*, 7(3), 14–25.
- Marichelvam, M. K., Geetha, M., & Tosun, Ö. (2020). An improved particle swarm optimization algorithm to solve hybrid flowshop scheduling problems with the effect of human factors—A case study. *Computers & Operations Research*, 114, 104812.
- Marichelvam, M. K., Prabaharan, T., Yang, X. S., & Geetha, M. (2013). Solving hybrid flow shop scheduling problems using bat algorithm. *International Journal of Logistics Economics and Globalisation*, 5(1), 15–29.
- Mirsanei, H. S., Zandieh, M., Moayed, M. J., & Khabbazi, M. R. (2011). A Simulated Annealing Algorithm Approach to Hybrid Flow Shop Scheduling with Sequence-Dependent Setup Times. *Journal of Intelligent Manufacturing*, 22(6), 965–978.
- Moccellin, J. V., Nagano, M. S., Neto, A. R. P., & de Athayde Prata, B. (2018). Heuristic algorithms for scheduling hybrid flow shops with machine blocking and setup times. *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, 40(2), 40.
- Morita, H., & Shio, N. (2005). Hybrid branch and bound method with genetic algorithm for flexible flowshop scheduling problem. *JSME International Journal. Series C, Mechanical Systems, Machine Elements and Manufacturing*, 48(1), 46–52.
- Mousavi, S. M., Mahdavi, I., Rezaeian, J., & Zandieh, M. (2018). An efficient bi-objective algorithm to solve re-entrant hybrid flow shop scheduling with learning effect and setup times. *Operational Research*, 18(1), 123–158.
- Naderi, B., Ruiz, R. A., & Zandieh, M. (2010). Algorithms for a realistic variant of flowshop scheduling. *Computers & Operations Research*, 37(2), 236–246.
- Naderi, B., Zandieh, M., Balagh, A. K. G., & Roshanaei, V. (2009). An improved simulated annealing for hybrid flowshops with sequence-dependent setup and transportation times to minimize total completion time and total tardiness. *Expert Systems with Applications*, 36(6), 9625–9633.
- Nawaz, M., Enscore, E. E., Jr., & Ham, I. (1983). A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. *Omega*, 11(1), 91–95.
- Negenman, E. G. (2001). Local search algorithms for the multiprocessor flow shop scheduling problem. *European Journal of Operational Research*, 128(1), 147–158.
- Öztop, H., Tasgetiren, M. F., Eliiyi, D. T., & Pan, Q. K. (2019). Metaheuristic algorithms for the hybrid flowshop scheduling problem. *Computers & Operations Research*, 111, 177–196.
- Pan, Q. K., & Dong, Y. (2014). An improved migrating birds optimisation for a hybrid flowshop scheduling with total flowtime minimisation. *Information Sciences*, 277, 643–655.
- Pan, Q. K., Ruiz, R., & Alfaro-Fernández, P. (2017). Iterated search methods for earliness and tardiness minimization in hybrid flowshops with due windows. *Computers & Operations Research*, 80, 50–60.

- Pan, Q. K., Tasgetiren, M. F., & Liang, Y. C. (2008). A discrete particle swarm optimization algorithm for the no-wait flowshop scheduling problem. *Computers & Operations Research*, 35(9), 2807–2839.
- Pan, Q. K., Wang, L., Li, J. Q., & Duan, J. H. (2014). A novel discrete artificial bee colony algorithm for the hybrid flowshop scheduling problem with makespan minimisation. *Omega*, 45, 42–56.
- Pan, Q. K., Wang, L., Mao, K., Zhao, J. H., & Zhang, M. (2012). An effective artificial bee colony algorithm for a real-world hybrid flowshop problem in steelmaking process. *IEEE Transactions on Automation Science and Engineering*, 10(2), 307–322.
- Pargar, F., & Zandieh, M. (2012). Bi-criteria SDST hybrid flow shop scheduling with learning effect of setup times: Water flow-like algorithm approach. *International Journal of Production Research*, 50(10), 2609–2623.
- Pargar, F., Zandieh, M., Kauppila, O., & Kujala, J. (2018). The effect of worker learning on scheduling jobs in a hybrid flow shop: A bi-objective approach. *Journal of Systems Science and Systems Engineering*, 27(3), 265–291.
- Pinedo, M. L. (2012). *Scheduling theory, algorithms, and systems* (4th ed.). New York: Springer.
- Portmann, M. C., Vignier, A., Dardilhac, D., & Dezalay, D. (1998). Branch and bound crossed with GA to solve hybrid flowshops. *European Journal of Operational Research*, 107(2), 389–400.
- Quadt, D., & Kuhn, H. (2005). Conceptual framework for lot-sizing and scheduling of flexible flow lines. *International Journal of Production Research*, 43(11), 2291–2308.
- Rajendran, C., & Chaudhuri, D. (1992). Scheduling in n-job, m-stage flowshop with parallel processors to minimize makespan. *International Journal of Production Economics*, 27(2), 137–143.
- Ribas, I., Leisten, R., & Framiñan, J. M. (2010). Review and classification of hybrid flow shop scheduling problems from a production system and a solutions procedure perspective. *Computers & Operations Research*, 37(8), 1439–1454.
- Ruiz, R., & Maroto, C. (2006). A genetic algorithm for hybrid flowshops with sequence dependent setup times and machine eligibility. *European Journal of Operational Research*, 169(3), 781–800.
- Ruiz, R., Şerifoğlu, F. S., & Urlings, T. (2008). Modeling realistic hybrid flexible flowshop scheduling problems. *Computers & Operations Research*, 35(4), 1151–1175.
- Ruiz, R., & Vázquez-Rodríguez, J. A. (2010). The hybrid flow shop scheduling problem. *European Journal of Operational Research*, 205(1), 1–18.
- Sha, D. Y., & Hsu, C. Y. (2006). A hybrid particle swarm optimization for job shop scheduling problem. *Computers & Industrial Engineering*, 51(4), 791–808.
- Shahvari, O., & Logendran, R. (2018). A comparison of two stage-based hybrid algorithms for a batch scheduling problem in hybrid flow shop with learning effect. *International Journal of Production Economics*, 195, 227–248.
- Shao, X., Liu, W., Liu, Q., & Zhang, C. (2013). Hybrid discrete particle swarm optimization for multi-objective flexible job-shop scheduling problem. *The International Journal of Advanced Manufacturing Technology*, 67(9–12), 2885–2901.
- Su, S., Yu, H., Wu, Z., & Tian, W. (2014). A distributed coevolutionary algorithm for multiobjective hybrid flowshop scheduling problems. *The International Journal of Advanced Manufacturing Technology*, 70(1–4), 477–494.
- Taillard, E. (1990). Some efficient heuristic methods for the flow shop sequencing problem. *European Journal of Operational Research*, 47(1), 65–74.
- Tseng, C. T., & Liao, C. J. (2008a). A particle swarm optimization algorithm for hybrid flow-shop scheduling with multiprocessor tasks. *International Journal of Production Research*, 46(17), 4655–4670.
- Tseng, C. T., & Liao, C. J. (2008b). A discrete particle swarm optimization for lot-streaming flowshop scheduling problem. *European Journal of Operational Research*, 191(2), 360–373.
- Vignier, A., Dardilhac, D., Dezalay, D., & Proust, C. (1996, November). A branch and bound approach to minimize the total completion time in a k-stage hybrid flowshop. In *Proceedings*

- 1996 IEEE conference on emerging technologies and factory automation. ETFA '96 (Vol. 1, pp. 215–220). New York: IEEE.
- Wang, S., Wang, X., & Yu, L. (2020). Two-stage no-wait hybrid flow-shop scheduling with sequence-dependent setup times. *International Journal of Systems Science: Operations & Logistics*, 7(3), 291–307.
- Wang, S. Y., Wang, L., Liu, M., & Xu, Y. (2013). An enhanced estimation of distribution algorithm for solving hybrid flow-shop scheduling problem with identical parallel machines. *The International Journal of Advanced Manufacturing Technology*, 68(9-12), 2043–2056.
- Zandieh, M., & Gholami, M. (2009). An immune algorithm for scheduling a hybrid flow shop with sequence-dependent setup times and machines with random breakdowns. *International Journal of Production Research*, 47(24), 6999–7027.
- Zhang, C., Ning, J., & Ouyang, D. (2010). A hybrid alternate two phases particle swarm optimization algorithm for flow shop scheduling problem. *Computers & Industrial Engineering*, 58(1), 1–11.
- Zhang, G., Shao, X., Li, P., & Gao, L. (2009). An effective hybrid particle swarm optimization algorithm for multi-objective flexible job-shop scheduling problem. *Computers & Industrial Engineering*, 56(4), 1309–1318.

Chapter 15

Constriction Coefficient-Based Particle Swarm Optimization and Gravitational Search Algorithm for Image Segmentation



Sajad Ahmad Rather and P. Shanthi Bala

Abstract Image segmentation is one of the pivotal steps in image processing. Actually, it deals with the partitioning of the image into different classes based on pixel intensities. In this work, a new image segmentation method has been introduced based on the constriction coefficient-based particle swarm optimization and gravitational search algorithm (CPSOGSA). The random samples of the image histogram act as searcher agents of the CPSOGSA. Besides, the optimal number of thresholds is determined using Kapur's entropy method. The effectiveness and applicability of CPSOGSA have been accomplished by applying it to four standard images from the USC-SIPI image database including airplane, cameraman, clock, and truck. Various performance metrics have been employed to investigate the simulation outcomes including optimal thresholds, standard deviation, mean, run-time analysis, PSNR (peak signal-to-noise ratio), best fitness value calculation, convergence maps, and box plot analysis. In addition, the experimental results of CPSOGSA are compared with standard PSO and GSA. The simulation results clearly indicate that hybrid CPSOGSA takes less computational time in finding the best threshold values of the benchmark images.

Keywords CPSOGSA · Image segmentation · Multilevel thresholding · Optimization · Kapur's entropy method · Particle swarm optimization (PSO) · Hybridization · Constriction coefficient · Meta-heuristics · Optimization · Gravitational search algorithm (GSA)

Nomenclature

Symbol	Extension
IS	Image segmentation
MT	Multilevel thresholding
HA	Heuristic algorithm

S. A. Rather (✉) · P. S. Bala
Pondicherry University, Kalapet, India

PSNR	Peak signal-to-noise ratio
SD	Standard deviation
SSIM	Structural similarity index measure
FSIM	Feature similarity index measure
WOA	Whale optimization algorithm
MFO	Moth flame optimizer
PSO	Particle swarm optimization
GA	Genetic algorithm
CPSOGSA	Constriction coefficient-based GSA and PSO
MSE	Mean square error
DE	Differential evolution
GSA	Gravitational search algorithm
BSA	Bat swarm algorithm
CS	Cuckoo search
FA	Firefly algorithm
ACO	Ant colony optimization
GWO	Grey wolf optimizer
KHO	Krill herd optimizer
HBO	Honey bee optimization

15.1 Introduction

Image segmentation (IS) is an important process in image processing and computer vision. It divides the image into many regions and pixels. In other words, IS simplifies the features of the image. Over the years, a number of IS methods have been proposed including edge detection (ED) (Papari & Petkov, 2011), thresholding (Otsu, 1979), etc. However, thresholding is mostly used IS technique due to its simple design and robustness (Oliva, Cuevas, Pajares, Zaldivar, & Osuna, 2014).

Basically, thresholding deals with the partitioning of the image into smaller segments based on grayscale intensity values. In fact, thresholding is classified into bi-level and multilevel thresholding (MT). The former one divides the image into two classes by considering only one threshold (th) value. On the other hand, MT requires more than two threshold values and splits pixels of the image into multiple classes.

In the research literature, bi-level thresholding has been solved by two famous IS methods, namely, Otsu's method (Otsu, 1979) and Kapur's method (Kapur, Sahoo, & Wong, 1985). The first method maximizes the class variance of the pixels, while Kapur's method deals with maximizing the histogram entropy of the image. However, when the number of threshold values is increased, the computational cost and complexity also increase (Horng, 2010). Consequently, aforementioned methods become incapable of solving real-time practical applications.

It has been proved that heuristic algorithms (HAs) have simplicity in design and high convergence speed (Maitra & Chatterjee, 2008). Therefore, nowadays HAs are readily employed for solving image segmentation problems by considering it as a constrained optimization task. The HAs which have been used to solve MT problem includes GA (Lai & Tseng, 2004), PSO (Gao, Xu, Sun, & Tang, 2009; Maitra & Chatterjee, 2008), HBO (Hornig, 2010), BFA (bacterial foraging algorithm) (Sathya & Kayalvizhi, 2011), equilibrium optimizer (EO) (Abdel-Basset, Chang, & Mohamed, 2020), and so on.

In this chapter, a novel hybrid optimization technique, namely, constriction coefficient-based particle swarm optimization and gravitational search algorithm (CPSOGSA) (Rather & Bala, 2019a, 2020a), has been employed for image segmentation. In CPSOGSA, global exploration of the solution space is performed by GSA, while exploitation is done by CPSO. The CPSOGSA will maximize Kapur's entropy objective function in order to find the high-intensity pixel values of the images. Besides, standard benchmark images have been used to evaluate the accuracy and effectiveness of the proposed method. Moreover, for comparative analysis, PSNR, SD, convergence maps, and box plots have also been considered.

The other parts of the chapter are structured as follows: First, the related works dealing with image segmentation by HAs are covered. Next, Kapur's method is discussed. Subsequently, CPSO and classical GSA are briefly explained. Moreover, the CPSOGSA and its application for IS task are introduced. Afterward, the experimental outcomes and simulation analysis of the results is carried out. Finally, the conclusion and future direction of the work are provided.

15.2 Literature Survey

It is obvious that high convergence speed and less time complexity are necessary for finding the optimal number of thresholds in image segmentation. The aforementioned features are present in HAs which makes them suitable for multilevel thresholding (MT) task. Moreover, in a hybrid approach, PSO has been used for IS in order to provide stochastic initialization. Besides, fuzzy clustering and Mahalanobis distance were utilized to collect image features and reduce the geometric complexity of classes, respectively. The performance evaluation was done by using test images from the brain web database. Also, the PSO-based HA indicated better performance (Benaichouche, Oulhadj, & Siarry, 2013).

In another work, GSA and GA algorithms were combined to provide high diversification power and fast convergence speed for making IS computationally inexpensive. Besides, the GSA-GA hybrid algorithm used Otsu's and entropy methods as fitness functions. Moreover, six standard images were employed for simulation analysis (Sun, Zhang, Yao, & Wang, 2016).

Krill herd optimization (KHO) algorithm is one of the recently introduced optimization technique that is inspired by the group dynamics of krills. The KHO has been employed for MT in order to find the intense pixels of the test images.

Moreover, the experiments have shown that KHO reduces run time and premature convergence issues associated with MT (Resma & Nair, 2018). Similarly, He and Huang (2020) have introduced a modified version of the KHO algorithm for color IS. They used three fitness functions including Otsu's between-class variance, Kapur's entropy, and Tsallis entropy for finding the best pixel values of the images. The efficient KHO algorithm was compared with six other HAs. Moreover, different performance metrics such as PSNR, SSIM (structural similarity index measure), standard deviation (SD), etc. were used to benchmark the performance. The simulation results indicated the robustness and efficiency of the modified KHO algorithm.

WOA and MFO are prominent HAs developed to solve complex optimization problems. They have been employed to solve IS problem. The optimal values of Otsu's objective function were used to get the best threshold outcomes. Several metrics were used for checking the efficiency of the WOA-MT and MFO-MT methods such as SSIM, PSNR, time complexity, and so on. The simulation results indicated the efficient performance of MFO as compared to WOA in terms of computational cost and exploitation rate (Abd El Aziz, Ewees, & Hassanien, 2017).

Statistical region merging (SRM) has also been used for MT in which centroid of the image histogram provides the best pixels. Moreover, the concept of cross entropy was utilized to eliminate noisy segmented regions of the output image. The experimental results of SRM were compared with other HAs like PSO, DE, CS, and so on (Li, Tang, Wang, & Zhang, 2019).

DE is one of the classical evolutionary optimization techniques having good exploitation power. It has been embedded with maximum Renyi entropy (MRE) for MT of hyper-spectral satellite images. Besides, SVM acts as a classifier to increase the recognition accuracy. Moreover, different HAs were employed for comparative analysis (Sarkar, Das, & Chaudhuri, 2016).

In another hybrid approach, dragonfly algorithm (DA) and DE have been combined for the color IS problem. The DA helps in global searching, whereas DE finds the optimal candidate solutions in the search space. Besides, standard images from the Berkeley database were used for performance evaluation. Different efficiency metrics and statistical tests were employed to check the applicability and potential of the DA-DE approach for color MT (Xu, Jia, Lang, Peng, & Sun, 2019).

GWO is another famous HA known for its mathematical simplicity and application potential. It was applied to the MT problem for finding the optimal features in the images. Both Kapur's and Otsu's methods were used as fitness functions. For comparative analysis, PSO and BFO algorithms were utilized, while the quality of the simulation results was measured through the structural similarity index (Khairuzzaman & Chaudhury, 2017).

HSA is an interesting HA based on the musician's instrument playing technique. It has the capability of powerful global exploration potential which is essential for the resolution of entrapment in local minima problem. Besides, HSA has been considered for MT to reduce the computational time and intensification issues of conventional IS methods. Standard test images were selected for performance benchmarking, and experimental results demonstrated the potential of HSA (Oliva, Cuevas, Pajares, Zaldivar, & Perez-Cisneros, 2013).

Quite often it has been reported that computational overhead and exploitation rate are two main issues that are faced by IS methods. To resolve these problems, crow search algorithm (CSA) has been employed for pixel optimization. Besides, the advantages of CSA like high intensification power and appreciable global exploration capability help to find optimal thresholds in less iterations. For comparative analysis, the simulation results were compared with PSO, DE, GWO, MFO, and so on (Upadhyay & Chhabra, 2019).

Equilibrium optimizer (EO) is yet another HA inspired by physical science. It is based on the concept of control V-M balance models. It has been applied to the grayscale IS problem while using Kapur's method as an objective function. Besides, Berkeley segmentation database of test images was employed for performance evaluation (Abdel-Basset et al., 2020). In addition, electromagnetism optimizer (EMO) is also used for MT of grayscale images to find high-intensity regions in test images. Both Otsu's and Kapur's methods were utilized to determine the precise number of pixels. The simulation results showed the efficient performance of EMO (Oliva et al., 2014).

Elephant herding optimizer (EHO) is another fascinating HA based on the clan and leadership behavior of elephants. It has been utilized for MT to select the best thresholds. Six standard test images, namely, barber, living room, boats, Gold Hill, lake, and aerial, were considered for testing the optimization capability. Besides, SD and mean values of EMO and other participating algorithms were calculated to compare simulation results (Tuba, Alihodzic, & Tuba, 2017).

It has been seen that the classical firefly algorithm (FA) has issues with global exploration which results in getting trapped at local minima. The aforementioned problem was resolved by combining Cauchy mutation and neighborhood schemes into FA. Besides, the applicability of improved FA was tested by applying it to IS problem. It showed efficient results as compared to PSO, DE, and classical FA (Chen et al., 2016).

Similarly, fireworks algorithm (FA) has been exercised for IS job to reduce the CPU time and accelerate the convergence speed of the traditional MT methods. Six benchmark images including lake, Barbara, Gold Hill, aerial, boats, and living room were used for performance evaluation. Moreover, statistical measures like average and SD were also calculated, while other HAs like PSO, DE, CS, and so on were employed for comparative analysis (Tuba, Bacanin, & Alihodzic, 2015).

The researchers have introduced another HA for MT, namely, flower pollination algorithm (FPA). In fact, some modifications have been introduced in classical FPA to accelerate its intensification power. The improved FPS equipped with high convergence capability had been applied to different tests and remote sensing images to measure its applicability and efficiency (Shen, Fan, & Huang, 2018).

Likewise, the multi-objective knee evolutionary algorithm (KEA) has been employed for IS problem to calculate the Pareto optimal solutions for different fitness functions. Various performance metrics were used to check the efficiency like PSNR, objective values, and convergence maps. The simulation outcomes depicted the productiveness and accuracy of KEA (Abd El Aziz et al., 2017). In addition, water cycle algorithm (WCA) is a recent HA inspired from the downward

moment of rainwater towards the sea. It has been considered for MT by employing two prominent objective functions, namely, Tsallis and Masi entropy. Different HAs like BA, PSO, grasshopper optimizer (GO), and so on were harnessed for comparative analysis. Moreover, different quality metrics and statistical measures were used for performance testing (Kandhway & Bhandari, 2019).

Researchers have applied modified grasshopper optimizer (MGO) for color segmentation task in order to reduce the time complexity and exploitation of the candidate solutions. In fact, Tsallis cross entropy was used for determining the optimal number of image pixels. Besides, simulation results were compared with the other two MT methods, namely, Otsu's and Renyi entropy. The experimental outcomes showed the potential and efficacy of MGO for IS (Liang et al., 2019).

The outline of the related works dealing with IS and MT by HAs are shown in Table 15.1. It is clear that traditional MT methods like Kapur and Otsu have the drawbacks of computational overhead and slow intensification rate which makes them unsuitable for image segmentation task. On the other hand, HAs like GSA, GA, PSO, etc. have high exploration power and accelerated exploitation potential necessary for determining the optimal thresholds present in the image histograms. Besides, PSNR, SSIM, FSIM, SD, and objective values are common performance metrics exploited by researchers for performance evaluation of the recent MT methods. In the present work, CPSOGSA has been utilized for image segmentation to find high-intensity pixels in the test images. Besides, different efficiency measures have been utilized for performance testing of the proposed CPSOGSA with other HAs.

15.3 Kapur's Segmentation Scheme

Kapur's entropy criterion (Kapur et al., 1985) is one of the traditional MT methods in image analysis. It deals with the maximization of the histogram entropy of the output image. It is based on the concept of the Shannon capacity (Shannon, 2001) which states that the occurrence of an event is directly related to the information content. Besides, Kapur's criterion is used for multilevel thresholding of grayscale images only.

Mathematically speaking, if an image consists of “ m ” thresholds, that is, $t_1, t_2, t_3, \dots, t_m$, having “ m ” classes, namely, $C_1, C_2, C_3, \dots, C_m$, then Kapur's fitness function is represented in Eq. (15.1).

$$f(t_1, t_2, t_3, \dots, t_m) = F_0, F_1, F_2, \dots, F_m \quad (15.1)$$

such that

Table 15.1 HA-based image segmentation techniques

References	Algorithm	Thresholding scheme	Performance metrics
Abdel-Basset et al. (2020)	EO	Kapur's entropy	PSNR, SSIM, and fitness values
He and Huang (2020)	Improved KHO	Kapur's and Tsallis entropy	SSIM, PSNR, and standard deviation
Resma and Nair (2018)	KHO	Kapur's and Otsu's methods	Objective values and convergence curves
Li et al. (2019)	DE	Minimum cross entropy	PSNR, mean shift, NCuts, and convergence graphs
Xu et al. (2019)	DA and DE	Otsu's and minimum cross entropy	PSNR, SSIM, FSIM, SD, and statistical test
Upadhyay and Chhabra (2019)	CSA	Kapur's entropy	Run time, p-values, SSIM, FSIM, and fitness values
Abd El Aziz et al. (2017)	KEA	Kapur, Tsallis, Otsu, and so on	PSNR, SSIM, and run time
Kandhway and Bhandari (2019)	WCA	Masi entropy	Objective values and PSNR
Liang, Jia, Xing, Ma, and Peng (2019)	MGO	Tsallis entropy	P-value and optimal threshold values
Shen et al. (2018)	FPA	Otsu's method	Fitness values, quality measures, and exploitation curves
Abd El Aziz et al. (2017)	WOA + MFO	Otsu's method	PSNR, SSIM, fitness values, and ANOVA test
Khairuzzaman and Chaudhury (2017)	GWO	Kapur's and Otsu's methods	SSIM and threshold values
Tuba et al. (2017)	EHO	Kapur's and Otsu's schemes	Mean and SD values
Sun et al. (2016)	GSA + GA	Kapur's and Otsu's methods	Threshold values and statistical test
Sarkar et al. (2016)	DE	Renyi entropy	ANOVA test, the Wilcoxon test, and run time
Chen et al. (2016)	Improved FA	Otsu's between-class variance	SD, mean, and computational time
Tuba et al. (2015)	FWO	Kapur's entropy	Threshold values, mean, and SD
Oliva et al. (2014)	EMO	Kapur's and Otsu's schemes	Thresholds, SD, PSNR, and iterations
Benaichouche et al. (2013)	PSO	Fuzzy c-means	Segmentation accuracy, run time, and thresholds
Oliva et al. (2013)	HSA	Kapur's and Otsu's methods	Objective values, SD, PSNR, and thresholds

$$F_0 = - \sum_{i=0}^{t_1-1} \frac{l_i}{\varphi_0} \ln \frac{l_i}{\varphi_0}, \varphi_0 = \sum_{i=0}^{t_1-1} l_i \quad (15.2)$$

$$F_1 = - \sum_{i=t_1}^{t_2-1} \frac{l_i}{\varphi_1} \ln \frac{l_i}{\varphi_1}, \varphi_1 = \sum_{i=t_1}^{t_2-1} l_i \quad (15.3)$$

$$F_2 = - \sum_{i=t_2}^{t_3-1} \frac{l_i}{\varphi_2} \ln \frac{l_i}{\varphi_2}, \varphi_2 = \sum_{i=t_2}^{t_3-1} l_i \quad (15.4)$$

$$F_m = - \sum_{i=t_m}^{L-1} \frac{l_i}{\varphi_m} \ln \frac{l_i}{\varphi_m}, \varphi_m = \sum_{i=t_m}^{L-1} l_i \quad (15.5)$$

Moreover, $\langle F_0, F_1, F_2, \dots, F_m \rangle$ are histogram entropies. Also, class probabilities are depicted by $\langle \varphi_0, \varphi_1, \varphi_2, \dots, \varphi_m \rangle$. Besides, l_i is the number of image pixels, and L is the upper limit of the grayscale range.

15.4 Constriction Coefficient-Based PSO

Particle swarm optimization is one of the highly popular and widely utilized optimization techniques in the field of swarm intelligence. It is inspired by the group behavior of birds and fishes. There are three main operators in PSO which are important for its optimization process, namely, inertia factor, pbest, and gbest. It is important to note that pbest and gbest parameters help in finding feasible regions of the solution space, whereas particle inertia aids in the global search. As particles change their values continuously in the successive iterations, therefore, updated particle velocity and position are calculated using Eqs. (15.6) and (15.7).

$$v_i^d(t+1) = w(t)v_i^d(t) + c_1 r_{i1}(\text{pbest}_i - x_i^d(t)) + c_2 r_{i2}(\text{gbest} - x_i^d(t)) \quad (15.6)$$

$$x_i^d(t+1) = x_i^d(t) + v_i^d(t+1) \quad (15.7)$$

Where $\langle c_1, c_2 \rangle$ are learning constants, while $\langle r_{i1}, r_{i2} \rangle$ are the numbers in the range of $[0, 1]$.

It has been seen that during the optimization process, the PSO particles move outside the solution space which results in the slow convergence of the candidate solutions towards feasible regions (Rather & Bala, 2019b, 2020b). To resolve the issue, constriction coefficients were introduced in PSO (Clerc et al., 2002) to accelerate the exploitation of the particles and, therefore, increase the performance of the PSO. The various CPSO parameters are as under:

$$\begin{aligned}\varphi_1 &= 2.05, \varphi_2 = 2.05, \\ \varphi &= \varphi_1 + \varphi_2\end{aligned}\quad (15.8)$$

$$K = 2 / (\varphi - 2 + \sqrt{\varphi^2 - 4}) \quad (15.9)$$

The path of the particles is controlled by the parameters represented by $\langle \varphi_1, \varphi_2 \rangle$, while K is the constriction coefficient. Additionally, inertia factor, $w(t) = K$, individual learning factor, $c_1 = K \varphi_1$, and social learning factor, $c_2 = K \varphi_2$, then the velocity in Eq. (15.6) is written in improved form as shown in Eq. (15.10).

$$\begin{aligned}v_i^d(t+1) &= \left(\frac{2}{\varphi - 2 + \sqrt{\varphi^2 - 4} v_i^d(t)} \right) + K \varphi_1 r_{i1} (\text{pbest}_i(t) - x_i^d(t)) \\ &\quad + K \varphi_2 r_{i2} (\text{gbest} - x_{id}(t))\end{aligned}\quad (15.10)$$

In order to sustain the balance in the PSO solution space, it is important that the value of φ should be greater than four. Besides, Eq. (15.10) shows the convergence of the particles towards global optima which is directly related to the social and personal learning factors of the particle system.

15.5 Standard Gravitational Search Algorithm (GSA)

GSA is one of the highly regarded physics-based HA. It is inspired by the law of gravitation and motion. In fact, gravity is one of the four basic forces in nature (Halliday, Resnick, & Walker, 2000; Rather & Bala, 2019c, 2019d; Rather & Sharma, 2017). The other three forces are weak nuclear force, electromagnetic force, and the strong nuclear force. Moreover, the law of gravitation is basically an inverse square law which states that “the attractive force between two masses is directly proportional to the product of their masses and inversely proportional to the square of the distance between them” (Rashedi, Nezamabadi-pour, & Saryazdi, 2009).

The GSA is first initialized with a random distribution of searcher agents in the form of masses. The force between the point masses is calculated in Eq. (15.11).

$$F_{ij} = G(t) \frac{m_{pi}(t)m_{aj}(t)}{R_{ij}(t) + \epsilon} \left(x_j^d(t) + x_i^d(t) \right) \quad (15.11)$$

where $m_{pi}(t)$ and $m_{aj}(t)$ are passive and active gravitational masses, respectively. The Euclidian distance is represented as $R_{ij}(t)$, while ϵ is a small constant.

To get a proper balance between exploration and exploitation, the GSA utilizes an important parameter called gravitational constant represented by “ G .” Besides, it helps in the accuracy of the search. It is given by Eq. (15.12).

$$G(t) = G(t_0) e^{(-\alpha_{\text{MI}}^{\text{CI}})} \quad (15.12)$$

where $G(t)$ and $G(t_0)$ are the values of the gravitational constant at time interval t and t_0 , respectively. Also, α is an exponentially decreasing coefficient, whereas CI and MI correspond to the current iteration and the maximum number of iteration(s), respectively.

As the masses are moving in the search space, and each of them is exerting a force. Therefore, the total force is given by Eq. (15.13).

$$F_i^d(t) = \sum_{j=1, j \neq i}^m \gamma_j F_{ij} \quad (15.13)$$

where γ_j has values between 0 and 1.

Furthermore, after a number of iterations, the heavy masses will be scattered throughout the search space which represents feasible solutions. So, it is important to preserve the quality of the best solutions. Therefore, cardinality constraint, i.e., k -best strategy, is used in GSA. It means that only optimal and efficient heavy mass will execute force in all directions after the fulfillment of stopping criterion. It is shown in Eq. (15.14).

$$F_i^d(t) = \sum_{j=k\text{best}, j \neq i}^m \gamma_j F_{ij}^d(t) \quad (15.14)$$

Moreover, the acceleration of the masses is calculated according to the second law of motion as given in Eq. (15.15).

$$a_i^d(t) = \frac{F_i^d(t)}{m_i(t)} \quad (15.15)$$

In GSA, the point masses get attracted to heavy masses because they have the highest intensity and strong force of attraction. Hence, the position and velocity of the heavy mass are pivotal for finding the global optimum which is provided in Eqs. (15.16) and (15.17), respectively.

$$v_i^d(t+1) = \gamma_j v_i^d(t) + a_i^d(t) \quad (15.16)$$

$$x_i^d(t+1) = x_i^d(t) + v_i^d(t+1) \quad (15.17)$$

15.6 CPSOGSA Algorithm

The reason behind designing hybrid CPSOGSA is to use the diversification capability of the GSA and the convergence power of CPSO. Besides, CPSOGSA also shields the candidate solutions to go outside the solution space which result in the faster exploitation of the particles towards global optimum. Moreover, the presence of gravitational constant, $G(t)$, in CPSOGSA takes searcher agents from low fitness regions of local minima and, hence, resolves entrapment in local minima problem. The unification equation that integrates the heuristic approaches is shown in Eq. (15.18).

$$\begin{aligned} V_i^d(t+1) \\ = \left(\frac{2}{\varphi - 2 + \sqrt{\varphi^2 - 4}} V_i^d(t) + K\varphi_1 r_{i1}(a_i^d(t) - x_i^d(t)) + K\varphi_2 r_{i2}(g\text{best} - x_i^d(t)) \right) \end{aligned} \quad (15.18)$$

where V_i^d is the velocity of the swarm particles, a_i^d is the acceleration of the particles, and g best represents the social capability component of the particle system.

The position of the particles X_i^d is given by Eq. (15.19).

$$X_i^d(t+1) = X_i^d(t) + V_i^d(t+1) \quad (15.19)$$

The pseudo-code of CPSOGSA is presented in Algorithm 15.1.

Algorithm 15.1: CPSOGSA Algorithm

- 1: Randomized initialization of the search space.
 - 2: Get the objective function values of candidate solutions.
 - 3: Initialize the parameters including the maximum number of iterations (T), initial value of the gravitational constant $G(t_0)$ and coefficient α .
 - 4: Start the iteration counter at $t = 0$.
 - 5: **while** $t < T$ **do**,
 - 6: **for** each candidate solution **do**,
 - 7: Update the gravitational constant, $G(t)$.
 - 8: Using Eq. (15.11) find the gravitational force, $F_i^d(t)$.
 - 9: Calculate the mass acceleration, $a_i^d(t)$ by using Eq. (15.15)
 - 10: Update the mass velocity, $V_i^d(t+1)$ with the help of Eq. (15.18)
 - 11: Update the mass position, $X_i^d(t+1)$ using Eq. (15.19)
 - 12: **end for**
 - 13: $t = t + 1$
 - 14: **end while**
 - 15: Return the optimal candidate solution
-

15.7 Image Segmentation Using CPSOGSA

In this work, CPSOGSA has been utilized for IS of grayscale images in order to locate the high-intensity regions of the image histogram. For determining the best threshold values, Kapur's entropy method is being exploited. As far as CPSOGSA is concerned, due to its hybrid nature, it has high diversification capability which relieves it from entrapment in local minima problem, whereas high intensification capability makes it immune from slow convergence and high computational overhead. The aforementioned advantages declare CPSOGSA ideal for IS task and capable enough to compete with other state-of-the-art HAs for multilevel thresholding.

The stepwise CPSOGSA algorithm for image segmentation is briefly formulated as follows:

- Step 1:** Read the grayscale image
- Step 2:** Acquire the histograms of the test images
- Step 3:** Randomized initialization of the CPSOGSA parameters
- Step 4:** Get the updated value of $G(t)$
- Step 5:** Using Eq. (15.11), find the value of $F_i^d(t)$
- Step 6:** Acceleration of the particles is calculated using Eq. (15.15)
- Step 7:** Find the velocity ($V_i^d(t + 1)$) of the optimal particles
- Step 8:** Particle positions are calculated by updating the value of $X_i^d(t + 1)$

It clearly shows that the optimization process starts with the reading of the test image. Then, Kapur's entropy acts as an objective function for getting the best threshold values. Consequently, CPSOGSA parameters change their values with consecutive iterations. Therefore, at last, it provides a segmented image as output having high-intensity pixels. In the next section, the simulation analysis of the results is carried out.

15.8 Experimental Results and Discussion

The applicability of the CPSOGSA has been tested by applying it for multilevel thresholding of the test images from the USC-SIPI image database. Four standard grayscale images, namely, aeroplane, cameraman, clock, and truck which have the same pixel size were utilized for performance evaluation. Figs. 15.1, 15.2, 15.3, and 15.4 depict standard images and their respective histograms.

It can be clearly seen that the histograms have chaotic behavior and complex solution space(s). The traditional MT methods are unable to handle complex search spaces, and therefore, HAs like CPSOGSA are employed to find feasible regions of the solution space and, hence, provide optimal solutions in less computational time.

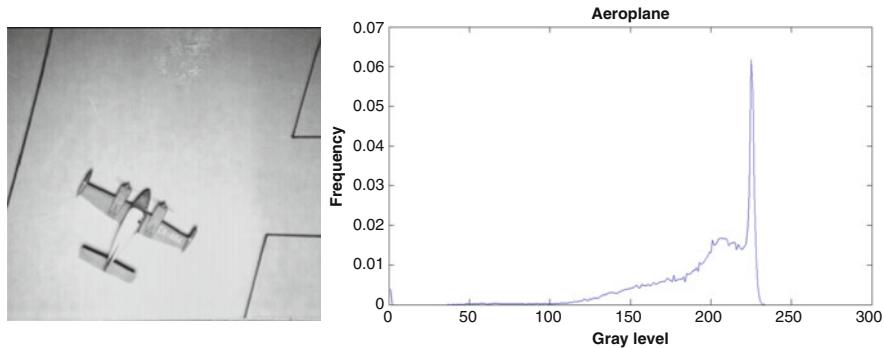


Fig. 15.1 Aeroplane image and its histogram

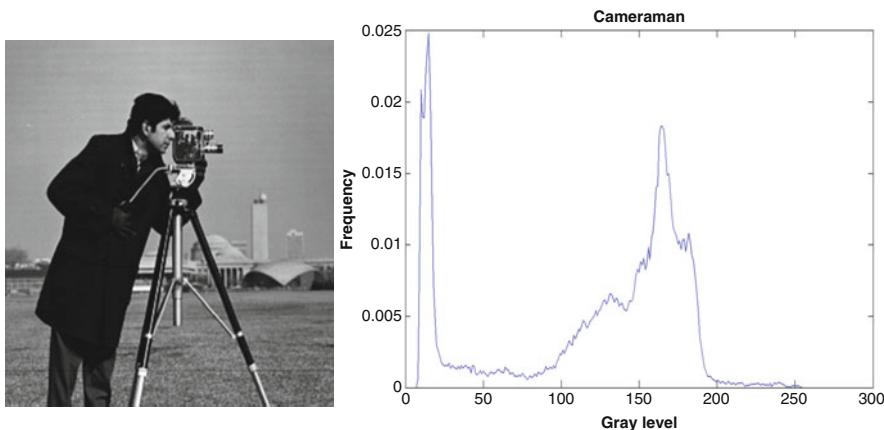


Fig. 15.2 Cameraman image and its histogram

15.8.1 Simulation Setup and Parameter Setting

The simulation results of the CPSOGSA have been compared with classical PSO (Kennedy & Eberhart, 1995) and standard GSA (Rashedi et al., 2009). Besides, the CPSOGSA will be judged for computational overhead, convergence speed, and capability of finding the optimal number of threshold values.

The experiments were performed on a computer system having Windows 10 OS, 2.2 GHz Intel Core i5 processor, 4GB RAM, and 500GB hard disk, and implementation was performed on the R2013a version of MATLAB. Moreover, MATLAB code is publicly available on the GitHub platform (<https://github.com/SAJADAHMAD1>) and the authors' MathWorks personal web page (<https://in.mathworks.com/matlabcentral/profile/authors/6240015-sajad-ahmad-rather>).

The search space of HAs will be the pixel range of images, that is, 0–255. Besides, the population size of the algorithms is the same (50) throughout the

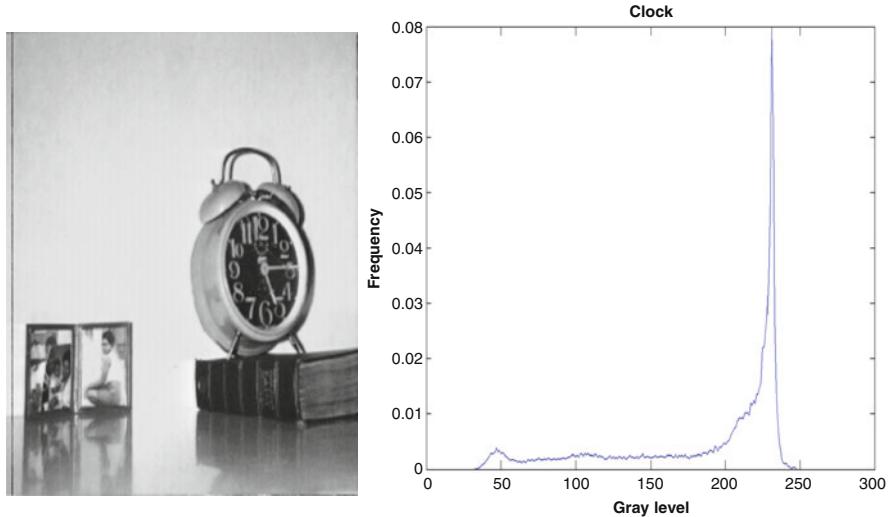


Fig. 15.3 Clock image and its histogram

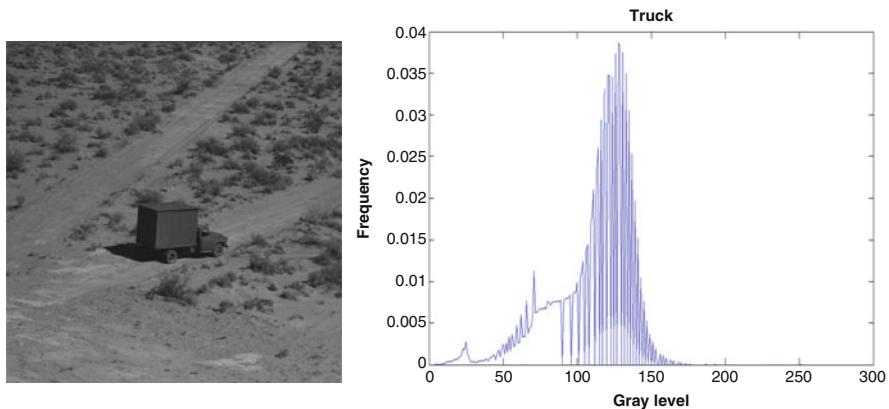


Fig. 15.4 Truck image and its histogram

optimization process. Moreover, when the objective value remains same for 10% of the maximum number of iterations (100), then CPSOGSA, PSO, and GSA will be stopped.

It is important to measure the accuracy and quality of the simulation results. Therefore, PSNR (peak signal-to-noise ratio) is selected as a potential performance metric for comparative analysis. In PSNR, the values of the optimal thresholds are considered. Besides, PSNR deals with the symmetry and accuracy of the output image. It is mathematically calculated as shown in Eq. (15.20).

Table 15.2 Initialization of HA parameters

Optimization technique	Initialization parameter	Value
PSO	c_1, c_2	2
	W_{\max}	0.9
	W_{\min}	0.2
GSA	Elitist check	1
	Rpower	1
	Min_flag (1 : minimum; 0 : maximum)	0
CPSOGSA	φ_1, φ_2	2.05
	Coefficient (α)	20
	$G(t_0)$	100

$$\text{PSNR} = 10 \log_{10} \frac{255^2}{\text{MSE}} \quad (15.20)$$

$$\text{MSE} = \frac{1}{RC} \sum_{i=1}^R \sum_{j=1}^C (I(i,j) - O(i,j))^2 \quad (15.21)$$

In Eq. (15.20), MSE is the mean square error, while R and C are the rows and columns of the image matrix, respectively. Besides, I is the input test image and O is the output image.

For all optimization algorithms, there are some parameters that have to be initialized for getting suitable simulation results. The initialization parameters of HAs have been presented in Table 15.2.

In the next section, the simulation results are presented for CPSOGSA, GSA, and PSO on four standard benchmark images to evaluate their performance. Besides, the capability of handling complex search spaces will also be checked.

15.8.2 Simulation Analysis of Aeroplane Benchmark

The experimental results of CPSOGSA, GSA, and PSO for aeroplane standard image are reported in Table 15.3. The fitness values are taken at 2, 3, 4, and 5 thresholds (k) of the sample image. Moreover, the best fitness outcomes of GSA are better than PSO and CPSOGSA. As far as CPSOGSA is concerned, its mean and SD values are in proximity with PSO. The optimal performance of CPSOGSA is validated by its low computational overhead at different threshold values $<16.90, 21.47, 27.41, 32.03>$ as compared to PSO $<17.74, 22.42, 28.45, 34.54>$ and GSA $<21.45, 25.96, 15.91, 37.61>$. The PSNR values of CPSOGSA are also appreciable.

The convergence graph and segmented image histogram of CPSOGSA at 100 iterations are shown in Fig. 15.5. It is evident from the convergence curve that fitness values are close to each other at successive iterations showing efficient performance. Besides, box plots of PSO, GSA, and CPSOGSA are represented in Fig. 15.6. It

Table 15.3 Simulation results for aeroplane benchmark

Image	Algorithm	k	Optimal thresholds	SD	Mean	PSNR	Best value	Run time
Aeroplane	PSO	2	7.50, 10.50	0.71	6.12	2.76	6.39	17.74
		3	1.57, 4.53, 6.48	1.12	6.46	2.54	6.17	22.42
		4	2.43, 6.36, 10.58, 12.48	1.22	8.39	2.81	7.71	28.45
		5	6.03, 12.81, 18.57, 21.80, 24.01	0.93	13.67	3.38	13.33	34.54
	GSA	2	194.91, 222.07	0.05	11.25	7.91	11.17	21.45
		3	148.90, 162.38, 201.99	0.10	13.65	15.34	13.78	25.96
		4	177.85, 195.22, 214.77, 228.76	0.16	16.96	10.54	17.15	15.91
		5	195.55, 195.84, 208.05, 210.28, 225.80	0.16	17.76	7.77	17.79	37.61
	CPSOGSA	2	2.91, 20.88	0.95	6	3.24	5.75	16.90
		3	1, 3.04, 4.01	1.82	4	2.45	3.58	21.47
		4	1.92, 2.80, 7.07, 6.67	1.93	5.69	2.58	5.32	27.41
		5	1.03, 4.79, 5.89, 9.82, 12.90	2.52	7.30	2.86	6.75	32.03

depicts that optimal objective function values of GSA are large, while CPSOGSA and PSO have proximal outcomes indicating symmetrical performance.

15.8.3 *Simulation Analysis of Cameraman Benchmark*

Table 15.4 shows the experimental outcomes for the cameraman standard image. It indicates that CPSOGSA has efficient performance as compared to PSO and GSA. It is because CPSOGSA has the best values for mean and standard deviation. Besides, PSNR values also depict the potential of CPSOGSA in finding the optimal regions of the image histogram. Moreover, the computational overhead of CPSOGSA is also minimum.

Figure 15.7 presents the convergence curve and histogram structure of the cameraman test image at $k = 5$. The fitness values of CPSOGSA show few deflections during the course of optimization as clearly evident from the exploitation map. Besides, box plots are presented in Fig. 15.8. It indicates high fitness values for GSA, while objectives values of CPSOGSA and PSO are in the close neighborhood.

15.8.4 *Simulation Analysis of Clock Benchmark*

The clock benchmark image is another famous test benchmark utilized for investigating the performance of HAs for MT task. The simulation outcomes are recorded

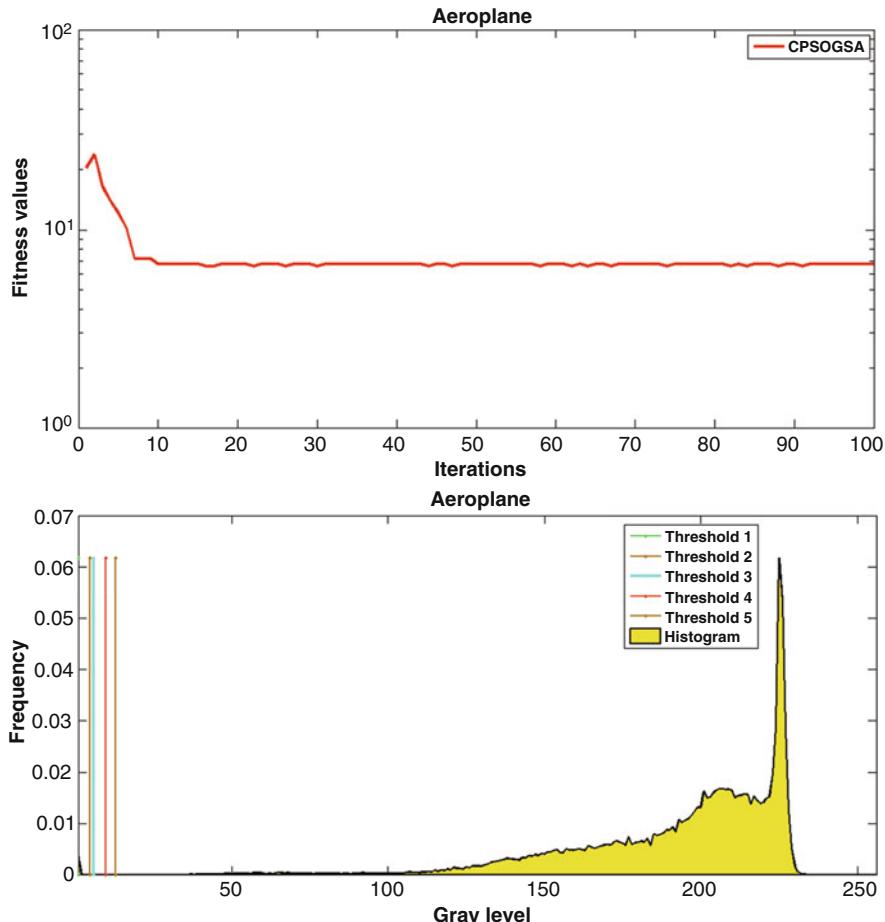


Fig. 15.5 CPSOGSA convergence curve and optimal thresholds at $k = 5$ for aeroplane image

in Table 15.5. It is evident that the PSNR values of CPSOGSA are better than PSO. Besides, mean values are also greater than PSO and close to GSA. However, GSA has optimal values for standard deviation. As far as CPU time is concerned, CPSOGSA takes less time to convergence towards global optimum. It shows that CPSOGSA has potential in handling complex search spaces.

The exploitation capability of the participating algorithms, namely, CPSOGSA, GSA, and PSO, is tested by analyzing convergence curves and, consequently, checking their ability of finding optimal number of thresholds as shown in Fig. 15.9. In addition, Fig. 15.10 shows the box plots of the clock image. It clearly indicates that PSO has sub-optimal performance as it has low fitness values. In contrast, GSA and CPSOGSA have appreciable performance as they have better values for the median and upper and lower quartiles.

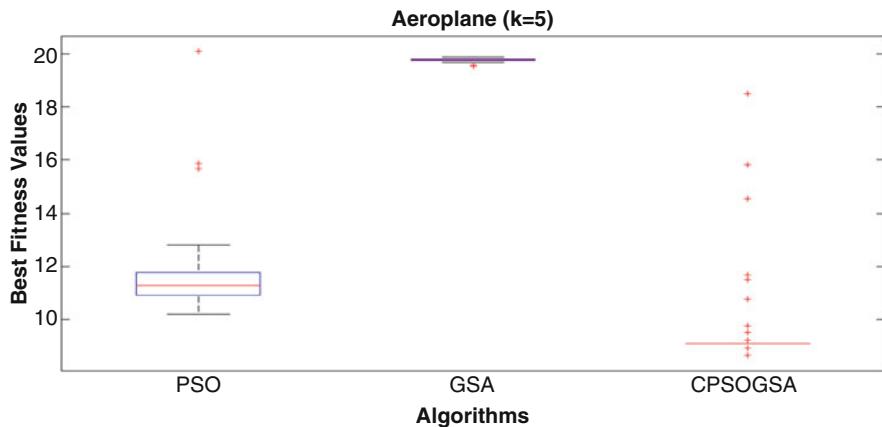


Fig. 15.6 Box plots of aeroplane benchmark

Table 15.4 Simulation results for cameraman benchmark

Image	Algorithm	<i>k</i>	Optimal thresholds	SD	Mean	PSNR	Best value	Run time
Cameraman	PSO	2	7.20, 9.40	1.13	4.80	6.11	4.37	16.98
		3	1.66, 2.76, 2.88	2.50	3.70	5.76	2.58	22.47
		4	12.33, 14.23, 14.70, 15.16	1.29	9.78	6.47	9.01	28.69
		5	0.49, 1.20, 1.31, 1.32, 1.49	3.55	1.35	5.64	0	32.19
	GSA	2	215.69, 239.38	0.04	11.42	5.67	11.46	8.18
		3	47.81, 62.21, 75.13	0.09	12.89	11.08	12.97	26.59
		4	202.04, 204.31, 213.26, 235.82	0.10	18.86	5.72	18.98	31.38
		5	102.49, 119.40, 124.01, 154.30, 173.32	0.07	18.69	19.95	18.75	36.76
	CPSOGSA	2	2.92, 20.31	0.85	8.68	6.78	8.47	20.45
		3	8.09, 12.95, 14.11	1.23	8.93	6.41	8.64	22
		4	5.81, 10.11, 13.99, 15.11	1.77	8.99	6.47	8.62	26.89
		5	2.05, 6.08, 9.05, 13.25, 31.07	1.66	10.52	7.52	9.91	28.94

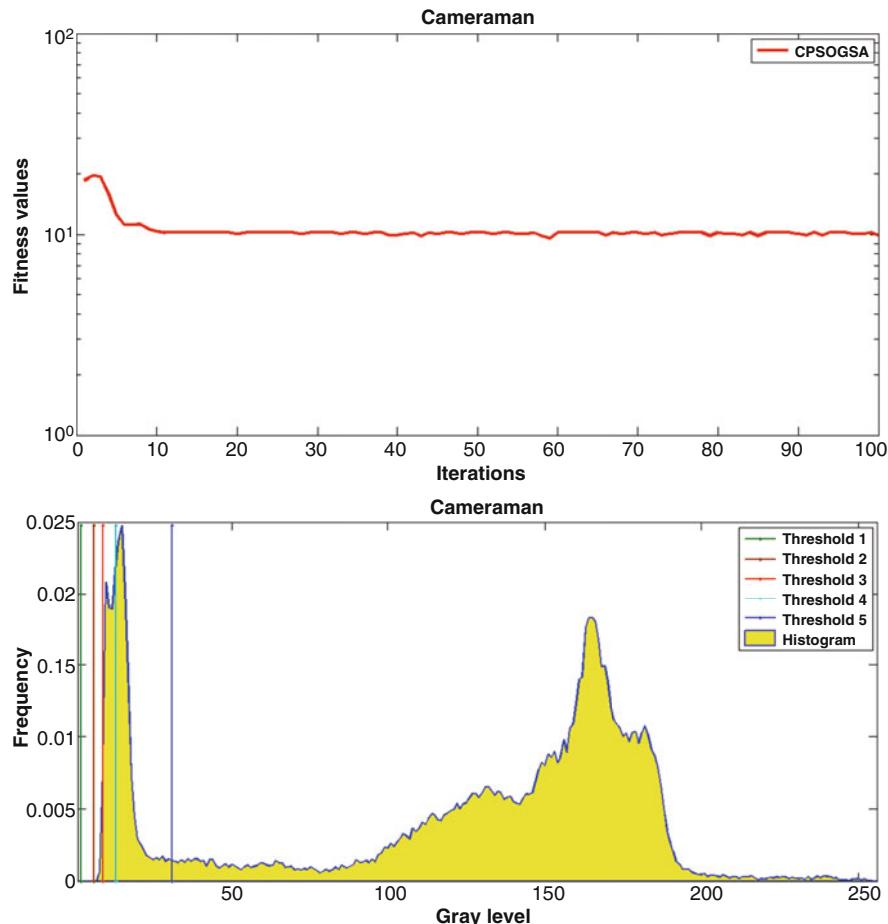


Fig. 15.7 CPSOGSA convergence curve and optimal thresholds at $k = 5$ for cameraman image

15.8.5 Simulation Analysis of Truck Benchmark

The truck test image is another benchmark image employed for comparative analysis of the HAs. Table 15.6 clearly shows that PSNR values of CPSOGSA and GSA are close to each other that indicates symmetrical performance in handling unknown solution spaces. Besides, the mean outcomes of CPSOGSA are much better than PSO. Moreover, yet again CPSOGSA has less computational overhead as compared to GSA and PSO.

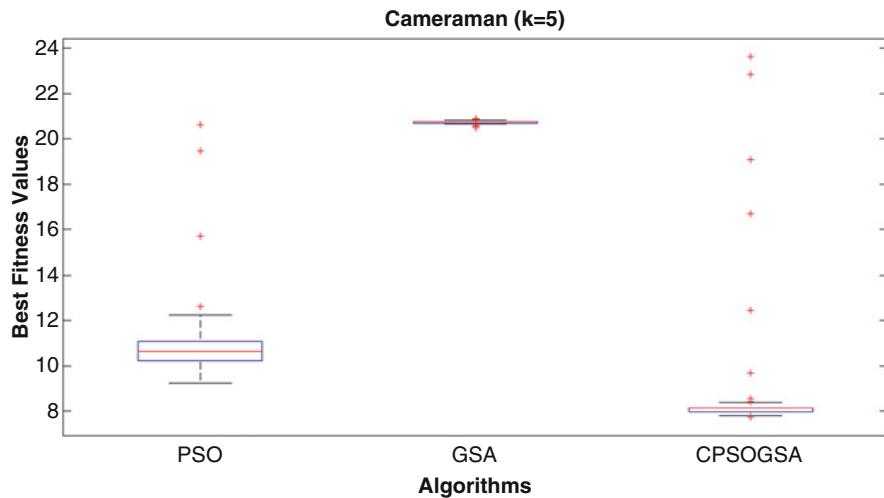


Fig. 15.8 Box plots of cameraman benchmark

The intensification of the truck image solution space by CPSOGSA is shown in Fig. 15.11. It indicates that the fitness values undergo huge variation in fitness values during the optimization process. The box plots are presented in Fig. 15.12. It is obvious that GSA has large values for objective function, while PSO and GSA have small values for median and interquartile ranges indicating appreciable performance.

Table 15.5 Simulation results for clock benchmark

Image	Algorithm	k	Optimal thresholds	SD	Mean	PSNR	Best value	Run time
Clock	PSO	2	5.03, 5.89	1.45	3.54	2.61	3.32	17.28
		3	9.33, 9.54, 9.76	1.07	8.07	2.78	7.78	22.69
		4	1.63, 1.91, 2.97, 3.86	2.97	3.78	2.52	3	26.74
		5	0.64, 0.97, 1.08, 1.49, 2.40	3.09	1.72	2.43	1	31.87
	GSA	2	155.10, 197.79	0.04	12.60	13.05	12.66	14.86
		3	170.75, 178.90, 188.72	0.13	16.86	11.68	17.04	26.11
		4	154.61, 162.29, 172.98, 183.29	0.08	16.57	12.44	16.63	31.40
		5	163.36, 168.14, 177.77, 190.80, 193.39	0.09	18.80	12.32	18.86	37.80
	CPSOGSA	2	2.94, 7.93	1.32	6.19	2.70	5.91	15.23
		3	7.08, 14.14, 30	1.17	10.89	3.72	10.61	19.65
		4	7.97, 15.98, 19.96, 29.90	1.07	12.53	3.72	12.30	24.53
		5	7.80, 13.24, 24.20, 46.93, 50.24	0.98	16.51	4.75	16.22	28.94

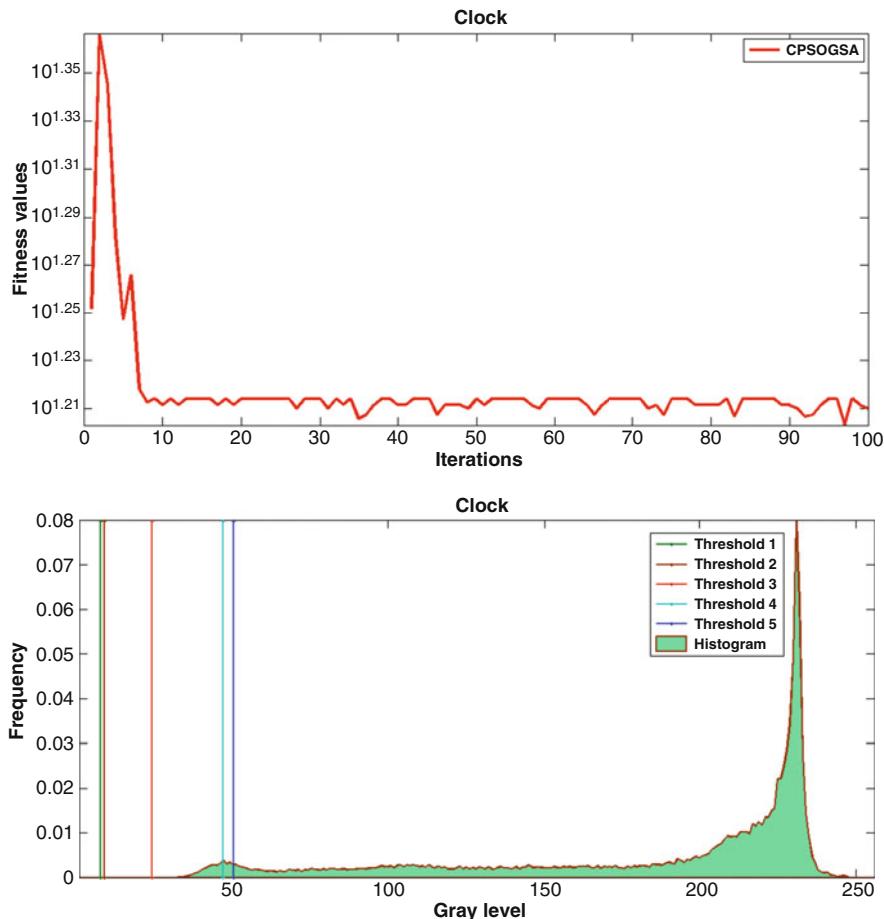


Fig. 15.9 CPSOGSA convergence curve and optimal thresholds at $k = 5$ for clock image

15.8.6 Summary of Experimental Outcomes

The simulation results confirm that CPSOGSA has better performance than PSO and GSA as far as convergence speed and computational time are concerned. Moreover, the PSNR values of CPSOGSA were better than PSO and close to GSA. However, the optimal threshold values for GSA were efficient as compared to CPSOGSA and

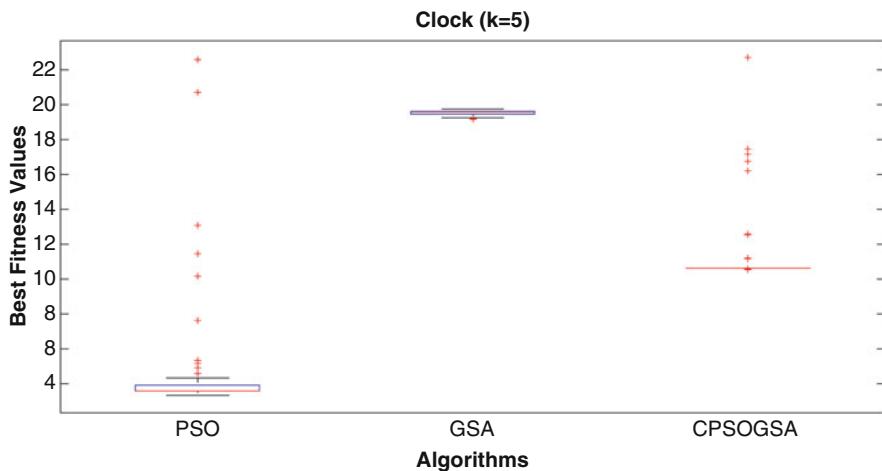


Fig. 15.10 Box plots of clock benchmark

Table 15.6 Simulation results for truck benchmark

Image	Algorithm	k	Optimal thresholds	SD	Mean	PSNR	Best value	Run time
Truck	PSO	2	2.79, 2.87	1.83	1.79	7.50	1	16.63
		3	0.92, 0.95, 1.32	2.68	0.92	7.34	0	20.88
		4	4.25, 25.67, 33.41, 33.99	0.91	12.51	10.29	11.37	25.09
		5	3.47, 7.58, 22.65, 23.11, 35.38	0.85	14.24	10.40	14.65	29.72
	GSA	2	202.39, 216.28	0.12	9.74	7.27	9.83	19.73
		3	80.10, 87.66, 115.98	0.05	14.98	18.34	15.01	26.12
		4	113.79, 116.90, 153.13, 165.19	0.11	16.67	12.22	16.61	30.96
		5	196.82, 198.19, 212.08, 221.29, 221.80	0.15	18.44	7.27	18.22	37.67
	CPSOGSA	2	10.99, 18.74	0.67	7.76	8.84	7.60	15.41
		3	6.09, 8.15, 16.95	1.61	7.19	8.66	6.78	19.88
		4	2.17, 6.94, 17.97, 23.10	1.17	10.57	9.21	10.07	24.35
		5	12.98, 22.09, 30.89, 37.98, 38.83	0.81	15.26	10.83	15.12	29.98

GSA. Another observation that can be deduced from the results is that as the number of threshold values increases, the computational overhead of PSO and GSA also increases. However, the run time of CPSOGSA is minimum in all four test image samples, which was quite surprising to see regardless of its hybrid nature. Moreover,

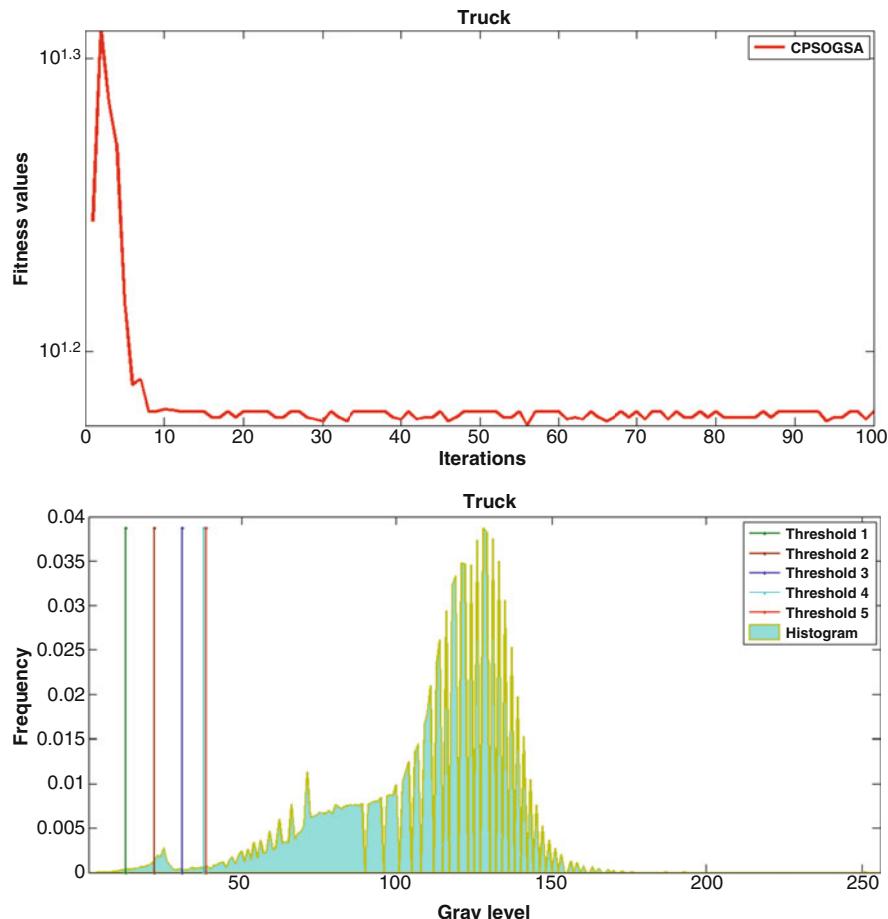


Fig. 15.11 CPSOGSA convergence curve and optimal thresholds at $k = 5$ for truck image

the SD and mean values of PSO and GSA remained close to each other in three images except the truck image where there was a significant difference in the values. To sum up, the CPSOGSA has been quite successful in overcoming exploitation and run time drawbacks of Kapur's entropy method.

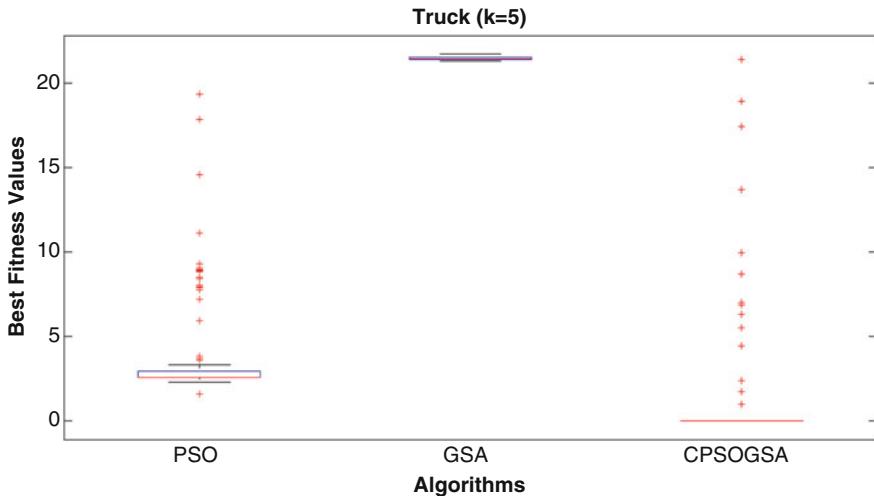


Fig. 15.12 Box plots of truck benchmark

15.9 Conclusion and Future Directions

In this chapter, a robust multilevel thresholding algorithm has been proposed, namely, a constriction coefficient-based on particle swarm optimization and gravitational search algorithm (CPSOGSA), for finding the optimal number of thresholds in the digital images. Moreover, four standard benchmark images, namely, aeroplane, cameraman, clock, and truck, from the USC-SIPI database have been utilized to evaluate the performance of the CPSOGSA for image segmentation. Besides, Kapur's method acts as a fitness function for the proposed hybrid approach. The quality of the segmentation is benchmarked through PSNR value which calculates the symmetry between sample image and segmented image.

The simulation results clearly indicate that CPSOGSA takes less computational overhead while finding the best thresholds of the benchmark images. Also, CPSOGSA has a high speed of convergence as compared to classical PSO. In addition, standard GSA also shows efficient results for the standard benchmark images.

As far as the future scope of the CPSOGSA is concerned, firstly, it can be utilized to find the optimal thresholds in the RGB color images. Secondly, Otsu's variance method can be used as an objective function in place of Kapur's method. Nowadays, it can be seen that medical data analysis is a very active and fertile application area of machine learning. Therefore, CPSOGSA can be utilized for thresholding of the complex medical dataset images. Moreover, the chaotic version of CPSOGSA can also be explored for the segmentation task.

Key Terms and Definitions

Heuristic algorithm (HA): A HA is basically an optimization algorithm that finds the global optimum solution after going through successive changes in the variables. In other words, HA finds the feasible regions of the solution space where probability of finding the optimal candidate solutions is very high.

Image segmentation (IS): It is the process of dividing the test image into various regions to remove the complexity and ambiguity from the image. In other words, IS helps in comprehending the features and information contained in an image.

Multilevel thresholding (MT): It is the process of finding the most suitable thresholds in the test image histogram. There are two main methods in MT, namely, Kapur's entropy and Otsu's between-class schemes.

Peak signal-to-noise ratio (PSNR): It is a popular performance metric used in the image analysis. It indicates the similarity between the input sample image and output optimized image.

References

- Abd El Aziz, M., Ewees, A. A., & Hassanien, A. E. (2017). Whale optimization algorithm and moth-flame optimization for multilevel thresholding image segmentation. *Expert Systems with Applications*, 83, 242–256.
- Abdel-Basset, M., Chang, V., & Mohamed, R. (2020). A novel equilibrium optimization algorithm for multi-thresholding image segmentation problems. *Neural Computing and Applications*, 2020, 1–34.
- Benaichouche, A. N., Oulhadj, H., & Siarry, P. (2013). Improved spatial fuzzy c-means clustering for image segmentation using PSO initialization, Mahalanobis distance and post-segmentation correction. *Digital Signal Processing*, 23(5), 1390–1400.
- Chen, K., Zhou, Y., Zhang, Z., Dai, M., Chao, Y., & Shi, J. (2016). Multilevel image segmentation based on an improved firefly algorithm. *Mathematical Problems in Engineering*, 2016, 1578056.
- Clerc, M., & Kennedy, J. (2002). The particle swarm-explosion, stability, and convergence in a multidimensional complex space. *IEEE transactions on Evolutionary Computation*, 6(1), 58–73.
- Gao, H., Xu, W., Sun, J., & Tang, Y. (2009). Multilevel thresholding for image segmentation through an improved quantum-behaved particle swarm algorithm. *IEEE Transactions on Instrumentation and Measurement*, 59(4), 934–946.
- Halliday, D., Resnick, R., & Walker, J. (2000). *Fundamentals of physics (6th Edition)*. Delhi: Wiley.
- He, L., & Huang, S. (2020). An efficient krill herd algorithm for color image multilevel thresholding segmentation problem. *Applied Soft Computing*, 89, 106063.
- Horng, M. H. (2010). Multilevel minimum cross entropy threshold selection based on the honey bee mating optimization. *Expert Systems with Applications*, 37(6), 4580–4592.
- Kandhway, P., & Bhandari, A. K. (2019). A water cycle algorithm-based multilevel thresholding system for color image segmentation using Masi entropy. *Circuits, Systems, and Signal Processing*, 38(7), 3058–3106.
- Kapur, J. N., Sahoo, P. K., & Wong, A. K. (1985). A new method for gray-level picture thresholding using the entropy of the histogram. *Computer Vision, Graphics, and Image Processing*, 29(3), 273–285.

- Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. *Proceedings of ICNN '95-IEEE International Conference on Neural Networks, 1995*, 1942–1948.
- Khairuzzaman, A. K. M., & Chaudhury, S. (2017). Multilevel thresholding using grey wolf optimizer for image segmentation. *Expert Systems with Applications*, 86, 64–76.
- Lai, C. C., & Tseng, D. C. (2004). A hybrid approach using Gaussian smoothing and genetic algorithm for multilevel thresholding. *International Journal of Hybrid Intelligent Systems*, 1 (3–4), 143–152.
- Li, J., Tang, W., Wang, J., & Zhang, X. (2019). A multilevel color image thresholding scheme based on minimum cross entropy and alternating direction method of multipliers. *Optik*, 183, 30–37.
- Liang, H., Jia, H., Xing, Z., Ma, J., & Peng, X. (2019). Modified grasshopper algorithm-based multilevel thresholding for color image segmentation. *IEEE Access*, 7, 11258–11295.
- Maitra, M., & Chatterjee, A. (2008). A hybrid cooperative–comprehensive learning based PSO algorithm for image segmentation using multilevel thresholding. *Expert Systems with Applications*, 34(2), 1341–1350.
- Oliva, D., Cuevas, E., Pajares, G., Zaldivar, D., & Osuna, V. (2014). A multilevel thresholding algorithm using electromagnetism optimization. *Neurocomputing*, 139, 357–381.
- Oliva, D., Cuevas, E., Pajares, G., Zaldivar, D., & Perez-Cisneros, M. (2013). Multilevel thresholding segmentation based on harmony search optimization. *Journal of Applied Mathematics*, 2013, 575414.
- Otsu, N. (1979). A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1), 62–66.
- Papari, G., & Petkov, N. (2011). Edge and line oriented contour detection: State of the art. *Image and Vision Computing*, 29(2–3), 79–103.
- Rashedi, E., Nezamabadi-pour, H., & Saryazdi, S. (2009). GSA: A gravitational search algorithm. *Information Sciences*, 179(13), 2232–2248.
- Rather, S. A., & Bala, P. S. (2019a). A holistic review on gravitational search algorithm and its hybridization with other algorithms. In *2019 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT)* (pp. 1–6). <https://doi.org/10.1109/ICECCT.2019.8869279>.
- Rather, S. A., & Bala, P. S. (2019b). Analysis of gravitation based optimization algorithms for clustering and classification. In *Handbook of research on big data clustering and machine learning* (pp. 77–99). Hershey: IGI Global. <https://doi.org/10.4018/978-1-7998-0106-1.ch005>.
- Rather, S. A., & Bala, P. S. (2019c). Hybridization of constriction coefficient based particle swarm optimization and gravitational search algorithm for function optimization. In *2019 Elsevier international conference on advances in electronics, electrical, and computational intelligence (ICAEEC-2019)*. Amsterdam: Elsevier. <https://doi.org/10.2139/ssrn.3576489>.
- Rather, S. A., & Bala, P. S. (2019d). Hybridization of constriction coefficient-based particle swarm optimization and chaotic gravitational search algorithm for solving engineering design problems. In *International conference on advanced communication and networking* (pp. 95–115). Singapore: Springer.
- Rather, S. A., & Bala, P. S. (2020a). A hybrid constriction coefficient based particle swarm optimization and gravitational search algorithm for training multi-layer perceptron (MLP). *International Journal of Intelligent Computing and Cybernetics*, 13(2), 129–165. <https://doi.org/10.1108/JICC-09-2019-0105>.
- Rather, S. A., & Bala, P. S. (2020b). Swarm-based chaotic gravitational search algorithm for solving mechanical engineering design problems. *World Journal of Engineering*, 17(1), 97–114. <https://doi.org/10.1108/WJE-09-2019-0254>.
- Rather, S. A., & Sharma, N. (2017). GSA-BBO hybridization algorithm. *International Journal of Advance Research in Science and Engineering*, 6, 596–608.
- Resma, K. B., & Nair, M. S. (2018). Multilevel thresholding for image segmentation using Krill Herd optimization algorithm. *Journal of King Saud University-Computer and Information Sciences*. <https://doi.org/10.1016/j.jksuci.2018.04.007>.

- Sarkar, S., Das, S., & Chaudhuri, S. S. (2016). Hyper-spectral image segmentation using Rényi entropy based multi-level thresholding aided with differential evolution. *Expert Systems with Applications*, 50, 120–129.
- Sathy, P. D., & Kayalvizhi, R. (2011). Optimal multilevel thresholding using bacterial foraging algorithm. *Expert Systems with Applications*, 38(12), 15549–15564.
- Shannon, C. E. (2001). A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(1), 3–55.
- Shen, L., Fan, C., & Huang, X. (2018). Multi-level image thresholding using modified flower pollination algorithm. *IEEE Access*, 6, 30508–30519.
- Sun, G., Zhang, A., Yao, Y., & Wang, Z. (2016). A novel hybrid algorithm of gravitational search algorithm with genetic algorithm for multi-level thresholding. *Applied Soft Computing*, 46, 703–730.
- Tuba, E., Alihodzic, A., & Tuba, M. (2017). Multilevel image thresholding using elephant herding optimization algorithm. In *2017 14th International conference on engineering of modern electric systems (EMES)* (pp. 240–243). New York: IEEE.
- Tuba, M., Bacanin, N., & Alihodzic, A. (2015). Multilevel image thresholding by fireworks algorithm. In *2015 25th International Conference Radioelektronika (RADIOELEKTRONIKA)* (pp. 326–330). New York: IEEE.
- Upadhyay, P., & Chhabra, J. K. (2019). Kapur's entropy based optimal multilevel image segmentation using crow search algorithm. *Applied Soft Computing*, 2019, 105522.
- Xu, L., Jia, H., Lang, C., Peng, X., & Sun, K. (2019). A novel method for multilevel color image segmentation based on dragonfly algorithm and differential evolution. *IEEE Access*, 7, 19502–19538.

Additional Readings

- Abd Elaziz, M., Ewees, A. A., & Oliva, D. (2020). Hyper-heuristic method for multilevel thresholding image segmentation. *Expert Systems with Applications*, 146, 113201.
- Chakraborty, F., Nandi, D., & Roy, P. K. (2019). Oppositional symbiotic organisms search optimization for multilevel thresholding of color image. *Applied Soft Computing*, 82, 105577.
- Kuruvilla, J., Sukumaran, D., Sankar, A., & Joy, S. P. (2016). A review on image processing and image segmentation. In *2016 International conference on data mining and advanced computing (SAPIENCE)* (pp. 198–203). New York: IEEE.
- Manikandan, S., Ramar, K., Iruthayarajan, M. W., & Srinivasagan, K. G. (2014). Multilevel thresholding for segmentation of medical brain images using real coded genetic algorithm. *Measurement*, 47, 558–568.
- Sayed, G. I., Hassanien, A. E., & Azar, A. T. (2019). Feature selection via a novel chaotic crow search algorithm. *Neural Computing and Applications*, 31(1), 171–188.

Chapter 16

An Overview of the Performance of PSO Algorithm in Renewable Energy Systems



Omar Hazem Mohammed and Mohammed Kharrich

Abstract An increase in the penetration of renewable energy sources in the electrical production has been matched by the emergence of many and varied challenges and problems. Among the most important challenges is finding the smart technologies and algorithms which are capable of achieving efficient solutions. This chapter provides an expanded view of the uses of the particle swarm optimization (PSO) algorithm in the renewable energy systems field. Additionally, it describes how the algorithm can be developed to cope with problems related to renewable energies to achieve desired goals. The PSO algorithm was used to solve many problems in the renewable energy systems, such as in optimal hybrid power systems, optimal sizing, and optimal net present cost, among others, where the PSO algorithm showed its high adaptability in problem-solving. Further, many researchers proceeded with the study and development of the PSO algorithm. In contrast, other researchers tried to hybridize it with different algorithms to be more efficient and convenient to overcome some of the problems and challenges that they encountered. The renewable energy systems have several issues to discuss, such as the cost of investment, the feasible technical criteria, optimal control, and the ecological problems as well as the social effect. Overall, studies and research have proven that the PSO algorithm is one of the best algorithms used in the field of renewable energy. This is attributed to the algorithm's simplicity, high efficiency, and effectiveness compared to other algorithms and optimization methods.

Keywords Particle swarm optimization · Multi-objective particle swarm optimization · Renewable energy systems · Hybrid system · Hybrid algorithm

O. H. Mohammed (✉)

Northern Technical University, Technical College of Mosul, Mosul, Iraq
e-mail: omar.hazem@ntu.edu.iq

M. Kharrich

Mohammed V University, Mohammadia School of Engineers, Rabat, Morocco

16.1 Introduction

The twentieth century has witnessed a tremendous amount of human progress in various fields, including energy. This follows the accumulation of knowledge and scientific progress made at the beginning of the twentieth century to the present. Among the most important of these advances is technological progress and the need to develop technologies to solve complex issues in all scientific fields. As a result, this led to an urgent need in the search for new and smart algorithms to keep up with the times. The field of renewable energy, like other important areas, has witnessed unprecedented development in the past decades and is now a recent topic of action and has more interest in the world. Many researchers used various techniques, methods, and algorithms to solve and mitigate different problems in the renewable energy domain such as meteorological prediction, system configurations, design and sizing problems, and dispatch problems, among others. While the PSO algorithm is always an efficient and useful metaheuristic algorithm, it's easy to be adapted in order to resolve these problems. The PSO algorithm is one of the most important metaheuristic algorithms and is used to resolve complex problems. Many renewable energy problems are treated using the PSO algorithm; herein, an overview of the PSO algorithm uses in different renewable energy fields are presented.

Optimization methods were limited and constrained because of a series of developments in this area in the mid-twentieth century. Dantzig developed the simplex method in 1947 to solve linear programming problems. Moreover, Bellman in 1957 employed the principle of optimality to solve dynamic programming problems. Subsequently, several contributions were presented, such as Kuhn and Tucker in 1951 on the necessary and sufficient conditions for the optimal solution for nonlinear programming. Zoutendijk and Rosen also made important contributions.

However, Dantzig, Charnes, and Cooper developed stochastic programming techniques and solved problems by assuming that the design parameters were independent and normally distributed. Charnes and Cooper in 1961 proposed the programming of the objectives for linear problems. Recently, modern optimization methods have emerged as powerful and popular methods of solving complex problems, especially in the engineering field. These methods include genetic algorithms, simulated annealing, particle swarm optimization, and ant colony optimization, among others (Yang, Chen, & Zhao, 2007).

The particle swarm optimization (Kennedy & Eberhart, 1995) is a metaheuristic algorithm inspired by the social behavior of bird flocking or fish schooling. The PSO algorithm was and is still very popular and has been extended to the MOPSO which is able to deal with multi-objective problems using an external memory and a geographically based approach for maintaining diversity (Coello Coello & Lechuga, 2002). The PSO popularity serves it to be the principal algorithm used in hybridization with multiple algorithms such as PSO-GWO (Şenel, Gökçe, Yüksel, & Yiğit, 2019), HEA which is a hybrid algorithm of the PSO and GA (Yang et al., 2007), PSO-SA (Idoumghar, Melkemi, Schott, & Aouad, 2011), and PSO-BFO (Liu XiaoLong, Li, & Ping, 2010).

The PSO has been improved in order to obtain more efficiency for the mono-objective functions. In (Bansal et al., 2011), the authors proposed multiple strategies of the inertia weight parameter of the PSO, which significantly affects the convergence and exploration-exploitation trade-off in the PSO process besides the multi-objectives. In (Mahfouf, Chen, & Linkens, 2004), the authors proposed a modified PSO to improve its performance using an adaptive inertia weight and acceleration factor to enhance the search capabilities. In addition, a weighted aggregation function was introduced to guide the selection of personal and global best solutions.

The authors in (Sedighizadeh & Masehian, 2009) provided an overview of the pre- and post-conditions of the PSO algorithm as well as the opportunities and challenges. The history of this algorithm, its various methods, and its classification are discussed, and its various applications are evaluated along with an analysis of these applications.

In the last decade, the topic of renewable energy has been a trending topic, which is characterized by huge parameters and technological input. Additionally, lots of software and smart algorithms have been developed and used to solve complex issues in renewable energy domains. The PSO algorithm has been adopted in a lot of research and studies to solve renewable energy problems, and it has proven its effectiveness compared to the new metaheuristic and smart algorithms used to solve most complex problems, especially when it is applied to renewable energy systems (Alshammari & Asumadu, 2020; HassanzadehFard & Jalilian, 2018; Keles, Alagoz, & Kaygusuz, 2017; Mozafar, Moradi, & Amini, 2017). The authors in (Kharrich, Sayouti, & Akherraz, 2018a) used the PSO to find the optimal sizing of a hybrid microgrid based on photovoltaic, wind, diesel, and battery energy storage systems. This hybrid microgrid was designed using the MOSPO, respecting environmental and economic factors (Kharrich, Sayouti, & Akherraz, 2018b).

In (Khaled, Eltamaly, & Beroual, 2017), the authors applied the PSO algorithm in power flow to avoid the voltage collapse problems by adding renewable energy in a 30-bus IEEE system. In (Yousif et al., 2018) the authors handled the PSO algorithm for a scheduling strategy of a microgrid system, while authors in (Zhao et al., 2019) adopted the PSO to simulate and calculate the pricing strategy model in order to find the optimal price of the microgrid market transactions and the optimal benefits of sellers. The authors in (Kharrich, Hazem Mohammed, Suliman, & Akherraz, 2019) introduced a recent review of the optimal sizing methodologies for the hybrid renewable systems that entails a decisive comparison among single and hybrid algorithms and software tools applied for optimal sizing of hybrid microgrid systems. Furthermore, an assessment was achieved for all the possible technical, economic, environmental, and social indices.

The authors in (Mansouri Kouhestani et al., 2020) tried to achieve the optimal economic and sizing of hybrid renewable system configuration by developing a strategy based on the PSO algorithm. The system consisted of PV, wind turbines, and a battery, taking into consideration environmental, reliability, and economic factors. In (Singh, Chauhan, & Singh, 2020), the authors proposed applying colony algorithm, particle swarm optimization, and a hybrid of both to design a hybrid

energy techno-economic system based on grid-connected solar photovoltaic/fuel cell to supply an electrical load demand in India.

The rest of this chapter presents an overview of the different enhancements and improvements as well as the importance still accorded to the PSO algorithm and its application in renewable energy systems. This chapter is arranged as follows: Sect. 16.2 presents optimization by using the particle swarm algorithm. Section 16.3 presents the optimization of renewable energy systems. Section 16.4 presents the PSO algorithm and its application in renewable energy systems. Finally, this chapter ends in the Conclusions section extracted from this work.

16.2 Optimization by using Particle Swarm Algorithm

The particle swarm optimization (PSO) is a stochastic and metaheuristic algorithm, developed by Kennedy and Eberhart in 1995. The PSO is a nature-inspired algorithm, motivated by swarm behaviors such as birds or fishes. The PSO algorithm is based on the displacement of the particles; each displacement is characterized by defining the actual position and the velocity of each particle, and the position and velocity are expressed in Eqs. (16.1) and (16.2). Herein, Fig. 16.1 presents the flowchart of the PSO, which contains the different steps of optimization. We can generally summarize it in the following steps (Lazinica, 2009; Olsson, 2010):

Step 1: Input the parameters of the system, constraints, and limitations.

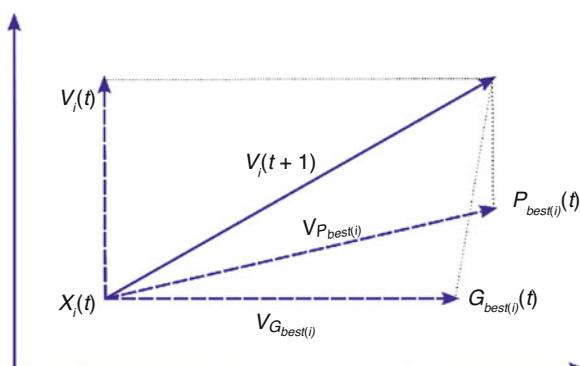
Step 2: Initialize the PSO algorithm settings.

Step 3: Set the iteration number equal to zero, and initialize the population of particles with random values (positions and velocities on dimensions).

Step 4: The objective functions of each particle are calculated and compared with the individual best value, the best P_b values are rearranged based on higher values, and the recent position of the particle is always recorded.

Step 5: Elect the particles associated with the best individual fitness value is known P_{best} for all particles, and define the value of P_{best} as the overall G_{best}

Fig. 16.1 The search technique of the PSO algorithm. Source: Authors' own creation



indicates to the global best position which reached by all the individuals of the population.

Step 6: Modernize the position and velocity of each particle.

Step 7: If the number of iterations reaches the final limit, go to Step 8; contrarily, begin the iteration index as equal to zero, and go to Step 4.

Step 8: The best particle expressed by G_{best} provides the optimal solution to the problem.

$$V_i(t+1) = w V_i(t) + c_1 r_1(t) (P_{\text{best}(i)}(t) - X_i(t)) + c_2 r_2(t) \times (G_{\text{best}(i)}(t) - X_i(t)) \quad (16.1)$$

$$X_i(t+1) = X_i(t) + \chi V_i(t+1) \quad (16.2)$$

where:

c_1 and c_2 represent the acceleration constants that attract each particle towards P_{BEST} and G_{BEST} positions.

r_1 and r_2 represent random real numbers dragged from [0, 1].

X_i and V_i represent the position and speed of the particle (i) in the study space.

w is the inertia weight which can be calculated randomly by Eq. (16.3) or proposed by a constant value such as (0.7). Alternatively, there are many techniques to estimate the inertia weight such as adaptive inertia weight, sigmoid increasing inertia weight, sigmoid decreasing inertia weight, and chaotic inertia weight, among others (Bansal et al., 2011).

χ is the constriction factor and results in the fast convergence of the particles over time and can be calculated by Eq. (16.4).

Figure 16.1 presents the search technique of the PSO algorithm in a multidimensional search space (Alam, 2016).

$$w = 0.5 + \frac{\text{rand}}{2} \quad (16.3)$$

$$\chi = \frac{2}{|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}|}, \varphi = c_1 + c_2 \quad (16.4)$$

16.3 The Optimization of Renewable Energy systems

The field of renewable energies is one of the modern fields that is developing rapidly. Consequently, it is difficult to limit it to a chapter; hence, this section focuses on some of the issues related to the optimization of renewable energy systems. The step of choosing the method to achieve optimization is one of the most important steps

that must be chosen carefully in order to reach an optimal solution to any problem in renewable energy systems.

The objectives of renewable energy systems vary to reach the optimal solution and achieve the goal function, whether by increasing or decreasing while respecting the different restrictions and conditions. Various criteria are considered to optimize renewable energy systems. These criteria are generally be classified as technical, economic, and controlling criteria.

16.3.1 Technical Criteria

Technical criteria include optimizing the technical objectives when designing renewable energy systems. The criteria include factors such as improving the efficiency of the renewable energy system, increasing their reliability, and maximizing the power availability. It also comprises environmental goals such as reducing harmful emissions that cause global warming and increasing the participation of renewable energies in the electric power generation system.

16.3.2 Economic Criteria

Economic criteria are one of the most important criteria for establishing optimal and cost-effective design of renewable energy systems. The criteria focus on reducing the total cost of renewable energy systems, minimizing the net present cost (NTPC), optimizing average generation cost, and reducing the cost of energy production (levelized cost of energy (LCoE)). Other objectives are minimizing the annual cost, capital cost, and maintenance and replacement cost, reducing the total cost of the system life cycle, and all cost-related optimizations such as buying and selling clean energy.

16.3.3 Control Criteria

The control criteria are conducted through optimal control of renewable energy systems. It includes various goals such as optimal energy management, optimal hybrid system allocation and placement, optimal control of the smart grid, and optimal maintaining of stability of power factor. Other objectives are improving the level of current and voltage of the system, reducing harmful harmonics in the supplied power, and optimal controlling of active and reactive power, optimal power flow, optimal VAR control, optimal dispatching model, and optimal CEED model, among others.

16.4 PSO Algorithm and its application in Renewable Energy systems

The unique abilities of the PSO algorithm have been exploited in various researches to resolve some of the problems in the field of renewable energy. In addition, optimal solutions have been achieved for many different goals of renewable energy systems, which have been mentioned in the previous section.

The results proved that the PSO algorithm has high efficiency in reaching the optimum solution in a short time compared to other algorithms. It is also useful in solving complex issues where optimization of renewable energy systems has been achieved, taking into account different goals and criteria such as technical, economic, and control criteria. In general, we can divide the exploitation of the PSO algorithm to reach the optimization in renewable energy systems into two categories:

1. The first category of researches is to use the classical PSO algorithm, where the algorithm is used alone without developing or improving its properties. As an example, in (Bouakkaz, Haddad, Martin-Garcia, Mena, & Castañeda, 2019), the standard PSO algorithm has been employed to schedule the household appliances in the stand-alone hybrid energy system to reduce and save the energy consumption cost. Many of scientific studies and researches with different aims and various subjects in renewable energy systems are tabulated in Table 16.1.
2. The second category is to take advantage of the PSO algorithm after developing and improving its properties or hybridizing it with other algorithms. These methods help to improve its capabilities in solving more complex problems that may also include multiple objects. As an example, in (Gholami & Dehnavi, 2019), the particle swarm optimization algorithm is modified for the optimal power-sharing among numerous renewable energy systems as wind/PV/combined heat and power plants within a microgrid system with the aim of cost minimization with and without load uncertainty. Many of researches with different aims and various subjects in renewable energy systems are tabulated in Table 16.2.

In general, it is possible to summarize the employment of the PSO algorithm to solve the problems of renewable energy systems in the following steps:

1. Initialization of parameters and site data under study such as weather and location data, wind speed, tidal speed, temperature, solar radiation, humidity, and other weather parameters.
2. Modeling and configuring parameters for renewable energy system elements such as the characteristics of solar panels, wind turbines, tidal turbines, storage system, and battery characteristics and modeling of all renewable energy elements used in renewable energy generation.
3. Determining the objectives of the project and the feasibility of its establishment and including all conditions and restrictions required.
4. Initializing the parameters and particles of the swarm algorithm for birds to represent all variables, goals, and constraints required in the algorithm.

Table 16.1 Application of PSO in renewable energy systems

Algorithm	Power system	Subject	Objective function	Refs.
PSO	Wind/tidal/PV/battery	Optimal design	– TNPC	Mohammed, Amirat, and Benbouzid (2019)
	PV/battery on grid	Optimal control Operations of a BESS	– Cost to charge the battery from RE	Hossain, Pota, and Moreno (2019)
	Wind/PV/battery	Optimal eco-nomic dispatch	– Social welfare	Eladl and ElDesouky (2019)
	Wind farms	Optimal active power dispatch	– Captured power – Fatigue distribution	Liao et al. (2020)
	Wind system	Optimal control	– Pitch angle	Ben Smida and Sakly (2019)
	Wind system	Optimal design	– Wind turbine's blade	Ma, Zhang, Yang, Hu, and Bai (2019)
	PV/wind turbine/batteries/diesel generator	Optimal scheduling of household appliances	– Energy saving – Reducing consumption – Optimal scheduling	Bouakkaz et al. (2019)
	Wind farms	Optimal placement	– LPC	Hou, Hu, Soltani, and Chen (2015)
	– PV/wind/diesel/battery – PV/tidal/bio-mass/battery – PV/biomass	Economic and technical assessment	– NPC	Mohammed Kharrich, Mohammed, and Akherraz (2019)
	Battery/microgrid system	Optimal real-time energy management	– Optimum energy control – Energy management – Cost of charging and discharging of energy	Hossain, Pota, Squartini, and Abdou (2019)
	PV/wind/MHP/biomass	Optimal sizing	– Optimal total annual cost – Power reliability	Chauhan and Dwivedi (2017)
	PV/wind/biomass	Optimal sizing	– Techno-economic feasibility – Cost of energy (COE) – LPSP	Sawle, Gupta, and Bohre (2017)
	PV/wind/diesel/battery	Optimal design	– NPC	Mohammed Kharrich, Mohammed, and Akherraz (2020)

Table 16.2 Improved version of PSO and its application

Algorithm	Power system	Subject	Objective function	Refs.
DNN-RODDPSO	Solar system	Accurate predictive model	– MSE	Ali Jallal, Chabaa, and Zeroual (2020)
WIPSO	Wind/solar/wave energies	Optimal allocation	– Eco-statistic	Masoumi, Ghassemzadeh, Hosseini, and Ghavidel (2020)
MPSO	– PV/wind/battery – PV/wind/on grid	Optimal design	– TIC	Hassan, Saadawi, Kandil, and Saeed (2015)
PSO-based	PV/wind/battery/diesel	Optimum size	– System cost	Mohamed, Eltamaly, and Alolah (2016)
PSO-based	PV/wind/battery/grid	Optimal sizing	– System cost – Environmental emission – Purchase electricity – The reliability	Mansouri Kouhestani et al. (2020)
MPBPSO	Wind/PV/CHP	Optimal schedule	– System cost	Gholami and Dehnavi (2019)
GPSOA	Wind/thermal	Optimal CEED model	– Fuel cost – Emission	Jiang, Zhang, Wu, and Chen (2019)
SIP-CO-PSO-ERS	FC/PV/wind/MT/diesel/battery	Optimal dispatching model	– OMC – PDE – LCC	Wang, Zhou, Ren, and Liu (2017)
CPSO	Wind/diesel	Optimal VAR control	– Real power loss	Hong, Lin, Lin, and Hsu (2013)
CPSO	Solar/wind/battery	Optimal hybrid system	– System cost – Environmental emission	Khare, Nema, and Baredar (2017)
MOPSO	Solar/wind/on grid	Renewable penetration	– Payback year – Power loss – Voltage stability	Kayal and Chanda (2015)
MOPSO	PV/CSP/ORC/TES/battery/LPG	Optimal design	– Minimum tariff – Propane consumption	Sigarchian, Orosz, Hemond, and Malmquist (2016)
PSO-GWO	PV/wind/BSS/diesel	Optimal design Optimal size	– Cost-effective – CO ₂ emissions	Abdelshafy, Hassan, and Jurasz (2018)
MOPSO	Wind/diesel/battery	Optimum size	– Net present cost – Energy not served	Abdoos and Ghazvini (2018)
MOPSO	Hydraulic/gas/thermal	Load shedding	– Lowest swing frequency – Amount of dropped load	Hafez, Hatata, and Abdelaziz (2019)

(continued)

Table 16.2 (continued)

Algorithm	Power system	Subject	Objective function	Refs.
MOPSO	PV/wind/energy storages (ESS)	Optimal design	– TPC – LPSP	Kiptoo et al. (2019)
Nested MOPSO	Wind/PV	Optimal sizing Optimal power flow	– System cost – Transmission line – Losses – Environmental emission	Eltamaly and Al-Saud (2018)
Advanced MOPSO	Wind/solar/bio-mass/capacitor banks	Optimal placement and sizing	– Power loss – Voltage stability – Voltage deviation	Kumar, Nallagownden, and Elamvazuthi (2017)
DMOPSO	PV/wind/diesel/battery/FC/electrolyzer/H2 tank	Optimal design	– NPC – CO2 emissions – Loss of load probability	Sharafi and ElMekkawy (2014)
IBBMOPSO	Solar system	Optimal design	– Output power – Thermal efficiency – Entropy yield	Niu, Wang, Sun, and Yang (2019)
PSO-SVR	Wind Power	Optimal wind power prediction	– Wind speed prediction – Wind power prediction – Combination model	Zhang, Sun, and Guo (2019)
Fuzzy-PSO	Photovoltaic system	Maximum power point tracking in PV	– Optimal parameter control in PV – Optimal design and intelligent controller	Soufi, Bechouat, and Kahla (2017)

5. Exploiting the algorithm to reach optimization by applying the rules of the PSO algorithm to all variables and characteristics of the renewable energy system represented by flying particles and updating their speed and distance covered to determine the objective function to reach the optimal solution.
6. Repeating the previous step and evaluating the results after each repetition after calculating the objective function until the program reaches the proposed convergence value or the number of proposed iterations and the optimum particles that represent the optimal solution that will be stored at the end of the program.

Finally, the characteristics of the PSO algorithm can be improved in solving complex multi-objective problems by developing its capabilities programmatically

or hybridizing it with other algorithms, which studies have proven to increase their efficiency and their ability to reach the optimal solution.

16.5 Conclusion

The rapid development taking place in the field of renewable energy in general and the field of renewable energy systems in particular has led to the emergence of many complex problems that need the application of smart algorithms to provide the optimal solutions. Many smart technologies and algorithms have been proposed to solve problems related to renewable energy systems. One of the most important of those algorithms is the PSO algorithm. Various researchers and scientists have focused on solving problems associated with renewable energy systems, especially in hybrid energy systems. PSO uses are summarized around three main axes: economic, technical, and control, and it has proven its best capabilities and high efficiency by reaching the optimum solutions with great convergence rate.

Among the most important advantages of the PSO algorithm are as follows:

- Has a lot of ease and flexibility.
- Has high competence and credibility in finding the optimal solutions.
- Able to perform a parallel calculation.
- Has a few parameters to set.
- Can be smart and powerful.
- Has the capability to reach into the optimal solution in a short time.
- Can be developed and studied with the possibility of improving and hybridizing it with new algorithms in light of the increasing challenges in renewable energy systems to solve the most complex problems.

However, among the most disadvantages of the PSO algorithm are the following:

- Has difficulty solving scattering problems.
- Can be fooled by the local minimum, particularly with multiple problems.
- It can be difficult to define basic design parameters.

With all of the above capabilities, the field of the renewable energy system is still open for many research and studies in the future as well as with regard to integrating this algorithm with many applications and smart algorithms and technologies to develop and improve its performance in reaching the optimal solution.

References

- Abdelshafy, A. M., Hassan, H., & Jurasz, J. (2018). Optimal design of a grid-connected desalination plant powered by renewable energy resources using a hybrid PSO–GWO approach. *Energy Conversion and Management*, 173, 331–347. <https://doi.org/10.1016/j.enconman.2018.07.083>.

- Abdoos, M., & Ghazvini, M. (2018). Multi-objective particle swarm optimization of component size and long-term operation of hybrid energy systems under multiple uncertainties. *Journal of Renewable and Sustainable Energy*, 10(1), 15902.
- Alam, M. N. (2016). Particle swarm optimization: Algorithm and its codes in matlab. *ResearchGate*, 8, 1–10.
- Ali Jallal, M., Chabaa, S., & Zeroual, A. (2020). A novel deep neural network based on randomly occurring distributed delayed PSO algorithm for monitoring the energy produced by four dual-axis solar trackers. *Renewable Energy*, 149, 1182–1196. <https://doi.org/10.1016/j.renene.2019.10.117>.
- Alshammari, N., & Asumadu, J. (2020). Optimum unit sizing of hybrid renewable energy system utilizing harmony search, Jaya and particle swarm optimization algorithms. *Sustainable Cities and Society*, 60, 102255. <https://doi.org/10.1016/j.scs.2020.102255>.
- Bansal, J. C., Singh, P. K., Saraswat, M., Verma, A., Jadon, S. S., & Abraham, A. (2011). Inertia weight strategies in particle swarm optimization. 2011 Third World congress on nature and biologically inspired computing, pp. 633–640.
- Ben Smida, M., & Sakly, A. (2019). Smoothing wind power fluctuations by particle swarm optimization-based pitch angle controller. *Transactions of the Institute of Measurement and Control*, 41(3), 647–656.
- Bouakkaz, A., Haddad, S., Martin-Garcia, J. A., Mena, A. J.-G., & Castañeda, R. J. (2019). Optimal scheduling of household appliances in off-grid hybrid energy system using PSO algorithm for energy saving. *International Journal of Renewable Energy Research (IJRER)*, 9(1), 427–436.
- Chauhan, A., & Dwivedi, V. K. (2017). Optimal sizing of a stand-alone PV/wind/MHP/biomass based hybrid energy system using PSO algorithm. *2017 6th International conference on computer applications in electrical engineering-recent advances (CERA)*, pp. 7–12.
- Coello Coello, C. A., & Lechuga, M. S. (2002). MOPSO: A proposal for multiple objective particle swarm optimization. *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No.02TH8600)*, 2, 1051–1056.
- Eladl, A. A., & ElDesouky, A. A. (2019). Optimal economic dispatch for multi heat-electric energy source power system. *International Journal of Electrical Power & Energy Systems*, 110, 21–35.
- Eltamaly, A. M., & Al-Saud, M. S. (2018). Nested multi-objective PSO for optimal allocation and sizing of renewable energy distributed generation. *Journal of Renewable and Sustainable Energy*, 10(3), 35302.
- Gholami, K., & Dehnavi, E. (2019). A modified particle swarm optimization algorithm for scheduling renewable generation in a micro-grid under load uncertainty. *Applied Soft Computing*, 78, 496–514.
- Hafez, A. A., Hatata, A. Y., & Abdelaziz, A. Y. (2019). Multi-objective particle swarm for optimal load shedding remedy strategies of power system. *Electric Power Components & Systems*, 47 (18), 1651–1666.
- Hassan, A., Saadawi, M., Kandil, M., & Saeed, M. (2015). Modified particle swarm optimisation technique for optimal design of small renewable energy system supplying a specific load at Mansoura University. *IET Renewable Power Generation*, 9(5), 474–483.
- HassanzadehFard, H., & Jalilian, A. (2018). Optimal sizing and siting of renewable energy resources in distribution systems considering time varying electrical/heating/cooling loads using PSO algorithm. *International Journal of Green Energy*, 15(2), 113–128.
- Hong, Y.-Y., Lin, F.-J., Lin, Y.-C., & Hsu, F.-Y. (2013). Chaotic PSO-based VAR control considering renewables using fast probabilistic power flow. *IEEE Transactions on Power Delivery*, 29(4), 1666–1674.
- Hossain, M. A., Pota, H. R., & Moreno, C. M. (2019). Real-time battery energy management for residential solar power system. *IFAC-PapersOnLine*, 52(4), 407–412.
- Hossain, M. A., Pota, H. R., Squartini, S., & Abdou, A. F. (2019). Modified PSO algorithm for real-time energy management in grid-connected microgrids. *Renewable Energy*, 136, 746–757.

- Hou, P., Hu, W., Soltani, M., & Chen, Z. (2015). Optimized placement of wind turbines in large-scale offshore wind farm using particle swarm optimization algorithm. *IEEE Transactions on Sustainable Energy*, 6(4), 1272–1282.
- Idoumghar, L., Melkemi, M., Schott, R., & Aouad, M. I. (2011). Hybrid PSO-SA type algorithms for multimodal function optimization and reducing energy consumption in embedded systems. *Applied Computational Intelligence and Soft Computing*, 2011, 138078. <https://doi.org/10.1155/2011/138078>.
- Jiang, S., Zhang, C., Wu, W., & Chen, S. (2019). Combined economic and emission dispatch problem of wind-thermal power system using gravitational particle swarm optimization algorithm. *Mathematical Problems in Engineering*, 2019, 5679361.
- Kayal, P., & Chanda, C. K. (2015). A multi-objective approach to integrate solar and wind energy sources with electrical distribution network. *Solar Energy*, 112, 397–410.
- Keles, C., Alagoz, B. B., & Kaygusuz, A. (2017). Multi-source energy mixing for renewable energy microgrids by particle swarm optimization. *2017 International Artificial Intelligence and Data Processing Symposium (IDAP)*, pp. 1–5.
- Kennedy, J., & Eberhart, R. (1995). *Particle swarm optimization*.
- Khaled, U., Eltamaly, A. M., & Beroual, A. (2017). Optimal power flow using particle swarm optimization of renewable hybrid distributed generation. *Energies*, 10(7), 1013.
- Khare, V., Nema, S., & Baredar, P. (2017). Optimisation of the hybrid renewable energy system by HOMER, PSO and CPSO for the study area. *International Journal of Sustainable Energy*, 36(4), 326–343.
- Kharrich, M., Hazem Mohammed, O. H. M., Suliman, M. Y., & Akherraz, M. (2019). A review on recent sizing methodologies for hybrid microgrid systems. *International Journal on Energy Conversion*, 7, 17813. <https://doi.org/10.15866/irecon.v7i6.17813>.
- Kharrich, M., Mohammed, O. H. M., & Akherraz, M. (2019). Assessment of renewable energy sources in Morocco using economical feasibility technique. *International Journal of Renewable Energy Research (IJRER)*, 9(4), 1856–1864.
- Kharrich, M., Sayouti, Y., & Akherraz, M. (2018a). Optimal microgrid sizing and daily capacity stored analysis in summer and winter season. *2018 4th International Conference on Optimization and Applications (ICOA)*, pp. 1–6.
- Kharrich, M., Sayouti, Y., & Akherraz, M. (2018b). Microgrid sizing with environmental and economic optimization. *2018 Renewable Energies, Power Systems & Green Inclusive Economy (REPS-GIE)*, pp. 1–6.
- Kharrich, M., Mohammed, O., & Akherraz, M. (2020). Design of hybrid microgrid PV/wind/diesel/battery system: Case study for Rabat and Baghdad. *EAI Endorsed Transactions on Energy Web*, 7(26), e7.
- Kiptoo, M. K., Adewuyi, O. B., Lotfy, M. E., Senjuu, T., Mandal, P., & Abdel-Akher, M. (2019). Multi-objective optimal capacity planning for 100% renewable energy-based microgrid incorporating cost of demand-side flexibility management. *Applied Sciences*, 9(18), 3855.
- Kumar, M., Nallagownden, P., & Elamvazuthi, I. (2017). Optimal placement and sizing of renewable distributed generations and capacitor banks into radial distribution systems. *Energies*, 10(6), 811.
- Lazinica, A. (2009). *Particle swarm optimization*. BoD: Books on Demand.
- Liao, H., Hu, W., Wu, X., Wang, N., Liu, Z., Huang, Q., Chen, C., & Chen, Z. (2020). Active power dispatch optimization for offshore wind farms considering fatigue distribution. *Renewable Energy*, 151, 1173–1185.
- Ma, Y., Zhang, A., Yang, L., Hu, C., & Bai, Y. (2019). Investigation on optimization design of offshore wind turbine blades based on particle swarm optimization. *Energies*, 12(10), 1972.
- Mahfouf, M., Chen, M.-Y., & Linkens, D. A. (2004). Adaptive weighted particle swarm optimisation for multi-objective optimal design of alloy steels. *International Conference on Parallel Problem Solving from Nature*, 3242, 762–771.
- Mansouri Kouhestani, F., Byrne, J., Johnson, D., Spencer, L., Brown, B., Hazendonk, P., & Scott, J. (2020). Multi-criteria PSO-based optimal design of grid-connected hybrid renewable energy systems. *International Journal of Green Energy*, 17, 617–631.

- Masoumi, A., Ghassem-zadeh, S., Hosseini, S. H., & Ghavidel, B. Z. (2020). Application of neural network and weighted improved PSO for uncertainty modeling and optimal allocating of renewable energies along with battery energy storage. *Applied Soft Computing*, 88, 105979.
- Mohamed, M. A., Eltamaly, A. M., & Alolah, A. I. (2016). PSO-based smart grid application for sizing and optimization of hybrid renewable energy systems. *PLoS One*, 11(8), e0159702.
- Mohammed, O. H., Amirat, Y., & Benbouzid, M. (2019). Particle swarm optimization of a hybrid wind/tidal/PV/battery energy system. Application to a remote area in Bretagne, France. *Energy Procedia*, 162, 87–96.
- Mozafar, M. R., Moradi, M. H., & Amini, M. H. (2017). A simultaneous approach for optimal allocation of renewable energy sources and electric vehicle charging stations in smart grids based on improved GA-PSO algorithm. *Sustainable Cities and Society*, 32, 627–637.
- Niu, Q., Wang, H., Sun, Z., & Yang, Z. (2019). An improved bare bone multi-objective particle swarm optimization algorithm for solar thermal power plants. *Energies*, 12(23), 4480.
- Olsson, A. E. (2010). *Particle swarm optimization: Theory, techniques and applications*. Hauppauge: Nova Science Publishers, Inc..
- Sawle, Y., Gupta, S. C., & Bohre, A. K. (2017). Optimal sizing of standalone PV/Wind/Biomass hybrid energy system using GA and PSO optimization technique. *Energy Procedia*, 117, 690–698.
- Sedighizadeh, D., & Masehian, E. (2009). Particle swarm optimization methods, taxonomy and applications. *International Journal of Computer Theory and Engineering*, 1(5), 486.
- Şenel, F. A., Gökçe, F., Yüksel, A. S., & Yiğit, T. (2019). A novel hybrid PSO–GWO algorithm for optimization problems. *Engineering with Computers*, 35(4), 1359–1373. <https://doi.org/10.1007/s00366-018-0668-5>.
- Sharafi, M., & ElMekkawy, T. Y. (2014). A dynamic MOPSO algorithm for multiobjective optimal design of hybrid renewable energy systems. *International Journal of Energy Research*, 38(15), 1949–1963.
- Sigarchian, S. G., Orosz, M. S., Hemond, H. F., & Malmquist, A. (2016). Optimum design of a hybrid PV—CSP—LPG microgrid with particle swarm optimization technique. *Applied Thermal Engineering*, 109, 1031–1036.
- Singh, S., Chauhan, P., & Singh, N. (2020). Capacity optimization of grid connected solar/fuel cell energy system using hybrid ABC-PSO algorithm. *International Journal of Hydrogen Energy*, 45(16), 10070–10088. <https://doi.org/10.1016/j.ijhydene.2020.02.018>.
- Soufi, Y., Bechouat, M., & Kahla, S. (2017). Fuzzy-PSO controller design for maximum power point tracking in photovoltaic system. *International Journal of Hydrogen Energy*, 42(13), 8680–8688.
- Wang, F., Zhou, L., Ren, H., & Liu, X. (2017). Search improvement process-chaotic optimization-particle swarm optimization-elite retention strategy and improved combined cooling-heating-power strategy based two-time scale multi-objective optimization model for stand-alone microgrid operation. *Energies*, 10(12), 1936.
- XiaoLong, L., Li, R. J., & Ping, Y. (2010). A bacterial foraging global optimization algorithm based on the particle swarm optimization. *2010 IEEE International Conference on Intelligent Computing and Intelligent Systems*, 2, 22–27.
- Yang, B., Chen, Y., & Zhao, Z. (2007). A hybrid evolutionary algorithm by combination of PSO and GA for unconstrained and constrained optimization problems. In *2007 IEEE International Conference on Control and Automation* (pp. 166–170). New York: IEEE.
- Yousif, M., Ai, Q., Gao, Y., Wattou, W. A., Jiang, Z., & Hao, R. (2018). Application of particle swarm optimization to a scheduling strategy for microgrids coupled with natural gas networks. *Energies*, 11(12), 3499.
- Zhang, Y., Sun, H., & Guo, Y. (2019). Wind power prediction based on PSO-SVR and grey combination model. *IEEE Access*, 7, 136254–136267.
- Zhao, W., Lv, J., Yao, X., Zhao, J., Jin, Z., Qiang, Y., Che, Z., & Wei, C. (2019). Consortium Blockchain-Based microgrid market transaction research. *Energies*, 12(20), 3812.

Chapter 17

Application of PSO in Distribution Power Systems: Operation and Planning Optimization



Paschalis A. Gkaidatzis, Aggelos S. Bouhouras, and Dimitris P. Labridis

Abstract Being an engineering field, power systems provide an extensive subject for optimization to be applied upon. Modern power systems have evolved in an increasingly highly complex system. The liberalization of the energy market and the introduction of distributed generation and, in particular, distributed renewable energy resources (DRES) have raised both opportunities and challenges that need to be tackled. Thus, complex issues related to the operation and planning of the distribution systems have emerged. Such issues involve many variables and refer to nonlinear objectives; thus their optimization is significantly based on heuristic techniques, such as particle swarm optimization (PSO). In this chapter, the implementation of PSO when contemplating various problems in power systems is presented. In particular, the utilization of PSO is demonstrated in the optimal distributed generation placement problem (ODGP), also known as optimal siting and sizing of distributed generation problem, and in the network reconfiguration problem. Finally, PSO is implemented in an optimal schedule of electric vehicles (EVs) charging, providing an apt example of the variety of problems for which PSO can be utilized and providing useful aid to important decisions, in the field of power systems.

Keywords Optimal · Distribution · Generation · Placement · Network · Reconfiguration · Electric · Vehicles · Heuristics · Optimization

P. A. Gkaidatzis (✉) · D. P. Labridis
Aristotle University of Thessaloniki, Thessaloniki, Greece
e-mail: pgkaidat@ece.auth.gr; labridis@auth.gr

A. S. Bouhouras
University of Western Macedonia, Kozani, Greece
e-mail: abouchouras@uowm.gr

17.1 Introduction

Power systems tend to be large, complex, and multivariate systems. Therefore, any optimization problem applied to any of the three major components of them, that is, generation, transmission, and distribution (Kothari, 2012), tends to be equally complex and very difficult to solve, suffering also from the dimensionality curse (Song, 2013). Optimization techniques are applied also to several aspects of power systems, such as operation, control, and planning.

Conventional optimization processes, such as linear programming, have been deployed for plain and optimal power flow (Albac, Bright, Prais, & Stott, 1990; Hyedt & Grady, 1983) or active and reactive power dispatch (Zhu & Irving, 1996). Nonlinear programming techniques have been used again for providing power flow solutions, being more accurate than linear but more time-consuming (Sun, Ashley, Brewer, Hughes, & Tinney, 1984), as well as for the problem of hydrothermal cooperation scheduling (Kothari, 1989). Additionally, there are problems that require integer and/or mixed-integer programming approaches, such as the unit commitment and economic dispatch problems (Chatzivasileiadis, 2018; Defourny & Terlaky, 2015; Dillon, Edwin, Kochs, & Taud, 1978). Dynamic programming has been also deemed necessary, when dealing with transmission planning (Partanen, 1990) or reactive power control (Lu & Hsu, 1995).

Apart from the conventional optimization methods, artificial neural network (ANN) (Dai, He, Fan, Li, & Chen, 1999) and fuzzy logic (Bansal, 2003) are also among the preferred techniques used to solve several power system problems.

On the one hand, notwithstanding the considerable achievements in conventional approaches, the latter still have not be applied to fast and reliable real-time implementations. Thus, significant labor is required to prevent mathematical entrapments, such as ill-conditioning and convergence arduousness. On the other hand, ANN or fuzzy techniques rely heavily on extensive expert domain knowledge. This means that they suffer from the expert user's knowledge in their design and utilization and moreover the lack of the formal model theory, being vulnerable in that way to the experts' depth of knowledge in problem definition (Bansal, 2005).

Metaheuristic techniques on the contrary can access deep knowledge via well-established mathematical models. There has been a variety of metaheuristic techniques utilized and in equally diverse variety of problems regarding power systems such as genetic algorithm (GA) in voltage control (Iba, 1994), simulated annealing (SA) for maintenance scheduling (Satoh & Nara, 1991), and tabu search for fault diagnosis/alarm processing (Wen & Chang, 1997).

ODGP is another power system problem that has proved quite a challenge itself (Jordehi, 2016). By strict conditions, ODGP contemplates the best positions, where the DG units should be connected to the distribution network (DN), and what should be their size in terms of rated capacity. Distributed generation includes technologies such as diesel generators (Paliwal, Patidar, & Nema, 2014) and microturbines (Ismail, Moghavvemi, & Mahlia, 2013), called collectively as conventional DG units, since they still use fossil fuels to produce electricity; then, there are also

renewable energy sources (RES), such as photovoltaics (PVs) (Palz, 2013), wind turbines (WTs) (World Wind Energy Association, 2014), and hydroelectric power plants (HPPs) (Chen, Chen, & Fath, 2015), that use natural resources, such as solar irradiance, wind, and water kinetic energy, respectively, in order to transform it to electric energy.

17.2 Literature Review

A few of the benefits that the integration of the DG in the DN offers are the reduction of greenhouse gas emissions, more liberated energy strategies, diversity of energy sources, peak operating cost decrease, network upgrades deferral, reduced power losses, decreased costs in transmission and distribution, and potential service quality augmentation towards the end customer (Georgilakis & Hatziargyriou, 2013). However, all of these, in order to be efficiently applied, require a proper strategic planning. Otherwise, critical consequences could impact the DN, such as loss increase (Atwa & El-Saadany, 2011; Gautam & Mithulanathan, 2007), voltage rise (Schwaegerl, Bollen, Karoui, & Yagmur, 2005; Tu, Yin, & Xu, 2018), reverse power flow (Delfanti, Falabretti, & Merlo, 2013), and reliability reduction (Abdmouleh, Gastli, Ben-Brahim, Haouari, & Al-Emadi, 2017; Esmaili, 2013). Therefore, the siting and sizing of the DGs could greatly affect all of those issues and thus become significantly important to solve.

The ODGP problem is by its nature mixed-integer nonlinear and therefore quite complex to solve. Several approaches have been utilized: empirical methods, such as the “2/3 rule” (Willis, 2000); analytical methods using the exact loss formula (Hung, Mithulanathan, & Bansal, 2010), loss sensitivity factors, or an improved analytical technique (Hung & Mithulanathan, 2011); numerical such as gradient search algorithm (Rau & Wan, 1994), dynamic programming (Khalesi, Rezaei, & Haghifam, 2011), linear programming (Keane & O’Malley, 2005), and exhaustive search method (Singh, Misra, & Singh, 2007); and finally metaheuristics techniques such as GA (Soroudi & Ehsan, 2011), differential evolution (DE) (Arya, Koshti, & Choube, 2012), artificial bee colony (ABC) (Seker & Hocaoglu, 2013), harmony search (Rao, Ravindra, Satish, & Narasimham, 2012), cuckoo search (CS) (Moravej & Akhlaghi, 2013), bacterial foraging optimization algorithm (BFOA) (Mohammadi, Rozbahani, & Montazeri, 2016), grey wolf pack optimization (Sultana, Khairuddin, Mokhtar, Zareen, & Sultana, 2016), ant-lion optimization (Ali, Abd Elazim, & Abdelaziz, 2016), and particle swarm optimization (PSO) (Prakash & Lakshminarayana, 2016).

The empirical methods, as previously stated, rely heavily on the experts’ knowledge depth. Moreover, they are restricted in uniformly distributed loads and radial DNs. As far as the analytical methods are concerned, they are perfectly suited when one DG is contemplated. However, when more than one is considered for installation, the problem becomes perplexed enough to solve via analytical methods. Numerical methods can tackle this issue. However, in order to do so, they either

require several assumptions, such as considering the problem as linear, instead of nonlinear, or searching exhaustively all the possible solutions in order to retrieve the optimal one. The former provides a deviation from reality; the latter proves to be rather time-consuming.

As for the metaheuristic techniques, they seem to provide several advantages when contemplating the ODGP: they are not restricted by type or size of the examined DN, load considered, or DG unit number. Moreover, they do not require any assumptions, thus solving the problem in its core. The only disadvantage they present is that being basically random search methods, they require more than one trial and are susceptible to local minima entrapment (Parsopoulos & Vrahatis, 2002). Therefore, the solution that they provide in most cases is near-optimal. Despite of their drawbacks, however, the benefits of using metaheuristics greatly outweigh their demerits.

This chapter focuses mainly on PSO and how it performs when applied to the ODGP problem. A comparison of various PSO versions is presented along with other metaheuristic techniques, in order to further enhance PSO performance. Additionally, the application of PSO in the combined problems of ODGP and NR is described. Finally, the application of PSO in the optimal schedule of EVs is demonstrated.

17.3 ODGP: Problem Determination

In this section the ODGP problem is determined by defining the objective function and the constraints that are required to be imposed.

17.3.1 Objective Function

As an objective function, the power losses is examined, that is:

$$F_{\text{loss}} = \min \sum_{k=1}^{l_k} g_{m,n} [V_m^2 + V_n^2 - 2V_m V_n \cos(\varphi_m - \varphi_n)] \quad (17.1)$$

where:

$g_{m,n}$:	Conductance between buses m and n , respectively
l_k :	Total network line number
V_m, V_n :	Buses m and n voltage magnitudes, respectively
φ_m, φ_n :	Buses m and n voltage angles, respectively

17.3.2 Constraints

The problem is bound to several constraints, such as technical constraints imposed by the DN and power flow constraints:

$$P_{G,m} - P_{D,m} - \sum_{n=1}^{l_n} |V_m| |V_n| |Y_{m,n}| \cos(\varphi_{m,n} - \varphi_m + \varphi_n) = 0 \quad (17.2)$$

$$Q_{G,m} - Q_{D,m} + \sum_{n=1}^{l_n} |V_m| |V_n| |Y_{m,n}| \sin(\varphi_{m,n} - \varphi_m + \varphi_n) = 0 \quad (17.3)$$

Voltage and line limits:

$$V_m^{\min} \leq V_m \leq V_m^{\max} \quad (17.4)$$

$$S_j \leq S_j^{\max} \quad (17.5)$$

where:

$P_{G,m}, Q_{G,m}$:	Bus m active and reactive power generation, respectively
$P_{D,m}, Q_{D,m}$:	Bus m active and reactive power demand, respectively
l_n :	Total network node number
$Y_{m,n}$:	Bus admittance element m,n
$\varphi_{m,n}$:	Bus admittance element m,n
V_m^{\min}, V_m^{\max} :	Bus m voltage lower and upper limits, respectively
S_j^{\max} :	Line j thermal limit, by terms of apparent power

There are also constraints with respect to the *DG* units themselves, such as the technical operation limits of the *DG* units considered for installation:

$$S_i^{\min} \leq S_i^{\max} \leq S_i^{\max} \quad (17.6)$$

$$pf_i^{\min} \leq pf_i^{\max} \leq pf_i^{\max} \quad (17.7)$$

And permitted *DG* penetration constraint, that is:

$$\sum_{i=1}^{m_{DG}} S_i^{\max} \leq \eta \cdot S_{\text{Total}}^{\text{Load}} \quad (17.8)$$

where:

$S_i^{\max} ::$	DG unit apparent power
$pf_i^{\max} ::$	DG unit power factor

(continued)

$S_{\min}^{DG}, S_{\max}^{DG} ::$	DG unit apparent power lower and upper limits, respectively
$pF_{\min}^{DG}, pF_{\max}^{DG} ::$	DG unit power factor lower and upper limits, respectively
$m_{DG} ::$	DG unit total number
$\eta ::$	Desired DG unit penetration level percentage
$S_{\text{Total}}^{\text{Load}} ::$	DN total load

17.4 Penalty Function Formulation

Generally, either deterministic or stochastic techniques have been utilized to solve optimization problems bound by constraints. Deterministic approaches, for example, feasible direction and generalized gradient descent, require strict mathematical properties for the objective function, such as continuity and differentiability. In addition, using analytical techniques to solve the ODGP problem could prove to be complex and time-consuming (Del Valle, Venayagamoorthy, Mohagheghi, Hernandez, & Harley, 2008) or be confined to solutions towards installing only one DG unit. However, these properties are not always present or easily employed. In those cases, evolutionary computation offers a reliable alternative option. Most evolutionary techniques, though, have been primarily designed to address unconstrained problems. Therefore, constrained handling techniques are usually accompanying the implementation of evolutionary techniques in order to detect and avoid any infeasible solutions. The most common practice dictates the use of a penalty function. Despite its disadvantages, if a proper calibration of the penalty parameters is undertaken, it performs rather efficiently (Parsopoulos & Vrahatis, 2007). To that end, constraints are expressed using penalty terms that in turn are incorporated into the objective function. This leads to the formulation of the penalty function. The latter penalizes any infeasible solutions as follows:

$$P(x) = f(x) + O(x) \quad (17.9)$$

$$O(x) = o \left\{ g^2(x) + [\max(0, h(x))]^2 \right\} \quad (17.10)$$

where:

$P(x) ::$	Penalty function
$f(x) ::$	Objective function
$O(x) ::$	Penalty term
$o ::$	Penalty factor
$g(x) ::$	Related equality constraints
$h(x) ::$	Related inequality constraints

Thus, in the case of the ODGP problem and using, for the sake of argument only, the F_{loss} as the objective as expressed in Eq. (17.1) and the DN technical constraints, that is, the equality constraints as defined in Eqs. (17.2) and (17.3) and inequality constraints as defined in Eqs. (17.4)–(17.8), the updated penalty function is expressed as:

$$P(x) = \min (F_{\text{loss}} + O_P + O_Q + O_V + O_L) \quad (17.11)$$

where O_P and O_Q refer to the equality constraints:

$$O_P = o_P \sum_{m=1}^{l_n} \left\{ P_{G,m} - P_{D,m} - \sum_{n=1}^{l_n} |V_m| |V_n| |Y_{m,n}| \cos(\varphi_{m,n} - \varphi_m + \varphi_n) \right\} \quad (17.12)$$

$$\begin{aligned} O_Q = o_Q \sum_{m=1}^{l_n} & \times \left\{ Q_{G,m} - Q_{D,m} + \sum_{n=1}^{l_n} |V_m| |V_n| |Y_{m,n}| \sin(\varphi_{m,n} - \varphi_m + \varphi_n) \right\} \end{aligned} \quad (17.13)$$

and O_V and O_L to inequality constraints:

$$O_V = o_V \sum_{m=1}^{l_n} \left\{ \max(0, V_m^{\min} - V_m) \right\}^2 + o_V \sum_{m=1}^{l_n} \left\{ \max(0, V_m - V_m^{\max}) \right\}^2 \quad (17.14)$$

$$O_L = o_L \sum_{j=1}^{l_k} \left\{ \max(0, S_j - S_j^{\max}) \right\}^2. \quad (17.15)$$

As it can be easily deduced, any other constraints such as Eqs. (17.6), (17.7), or (17.8) can be incorporated in Eq. (17.11) via the same process.

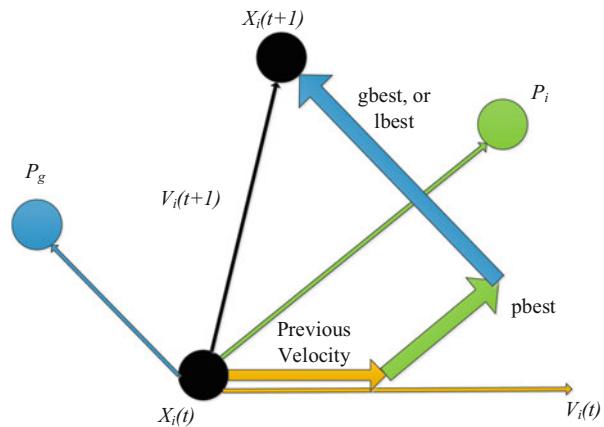
17.5 PSO Analysis

17.5.1 General

In this section the PSO algorithm is presented, as addressed and appropriately adjusted, in order to contemplate the ODGP problem.

PSO has been developed by Kennedy and Eberhart (Eberhart & Kennedy, 1995) and was inspired by the movement of fish schools or herds of animals that are trying to find some food source or avoid a potential enemy. Mathematically, the main idea is that, having defined the feasible solution space, a swarm of particles explores

Fig. 17.1 PSO velocity diagram. Source: Authors' own creation



it. Their position within the solution space changes with every iteration step, along with their velocity. This change in velocity consists of three terms:

1. The personal or cognitive knowledge of the solution space, as gathered by each individual particle.
2. The social knowledge of the solution space that each particle has gathered after communicating with other particles.
3. Its previous status.

This can be formulated in the following generalized equations:

$$v_h(t + 1) = v_h(t) + c_1 R_1 (P_h(t) - X_h(t)) + c_2 R_2 (P_g(t) - X_h(t)) \quad (17.16)$$

$$X_h(t + 1) = X_h(t) + v_h(t + 1) \quad (17.17)$$

where:

$h = 1, \dots, N:$	Particle number
$X_h(t):$	Current position of particle h
$X_h(t + 1):$	Next position of particle h
$v(t):$	Current velocity of particle h
$v_h(t + 1):$	Next velocity of particle h
$P_h(t):$	Personal best
$P_g(t):$	Social best
$c_{1, 2}:$	Weighting factors, i.e., cognitive and social parameters, respectively
$R_{1, 2}:$	Random variables uniformly distributed within [0,1]

This movement is also depicted in Fig. 17.1.

17.6 Velocity Limits

In order to address the swarm explosion issues, velocity thresholds have been imposed separately to each dimension of the particle, i.e.:

$$v_{\max,l} = \frac{b_l - a_l}{d} \quad (17.18)$$

where:

$v_{\max,l}$:	Maximum velocity threshold for dimension l
b_l :	Upper bound of dimension l
a_l :	Lower bound of dimension l
d :	A denominator factor, most commonly equal to 2

17.7 Particle Formulation

With respect to the dimensions of each particle, i.e., the solution space as demonstrated in Fig. 17.2, these consist of the bus number where the DG unit should be installed and of the unit size, expressed via its active and reactive rated power.

This particle formulation implies that the solution space is directly proportional to the number of DG units considered. The number of DG units is considered equal to the number of buses of the examined DN.

In order to ensure a more rapid convergence, the initially random values of particle dimensions at the beginning of the optimization process are considered within certain limits, such as the maximum number of buses in the examined DN and the operating technical limits of the examined DG units as described in Eqs. (17.5) and (17.6). This does not prevent the algorithm from retrieving the optimal solution. Furthermore it provides a more realistic approach, since it filters infeasible results especially at the first iteration steps.

Additionally, with respect to the algorithm's termination, two criteria are considered: a maximum number of iteration steps and a minimum convergence deviation

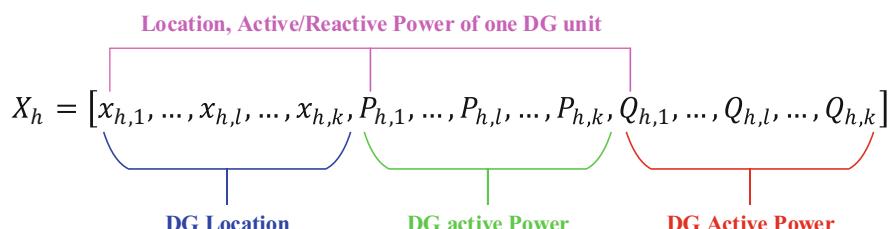


Fig. 17.2 Solution space formulation of ODGP problem. Source: Authors' own creation

between current and previous solution that need both to be met in order for the algorithm to conclude. This way the algorithm is given the opportunity to explore and exploit even more the solution space, thus augmenting its performance.

17.8 Reducing Perturbation

The first term of the sum in the second leg of Eq. (17.14), i.e., the previous velocity term, is the component that describes the previous status of the particle. As that, it infuses a degree of inertia. However, for the same reason, this term has also been connected to a certain degree of perturbation. As a result, the local minima entrainment risk is reduced, but at the expense of having the particles oscillate on broad ranges around the best positions found (Shi & Eberhart, 1998). Two solutions have been found and applied, that is, the inertia weight and the constriction factor parameters (Eberhart & Shi, 2000), as presented in the following equations:

$$v_h(t+1) = \omega v_h(t) + c_1 R_1 (P_h(t) - X_h(t)) + c_2 R_2 (P_g(t) - X_h(t)) \quad (17.19)$$

where:

$$\omega(t) = \omega_{\max} - (\omega_{\max} - \omega_{\min}) \frac{t}{T_{\max}} \quad (17.20)$$

where:

$\omega(t)$:	Current inertia weight
ω_{\max} :	Maximum inertia weight
ω_{\min} :	Minimum inertia weight
t :	Current iteration
T_{\max} :	Maximum iteration number

and:

$$v_h(t+1) = \psi [v_h(t) + c_1 R_1 (P_h(t) - X_h(t)) + c_2 R_2 (P_g(t) - X_h(t))] \quad (17.21)$$

where:

ψ :	Constriction factor
----------	---------------------

Applying these two solutions on a test bus system, i.e., the typical 16-bus system (Civanlar, Grainger, Yin, & Lee, 1988) presented in Fig. 17.3, bears the results depicted in Fig. 17.4. For 1000 iteration steps, it shows that the constriction factor method demonstrates better performance, both in terms of convergence speed and

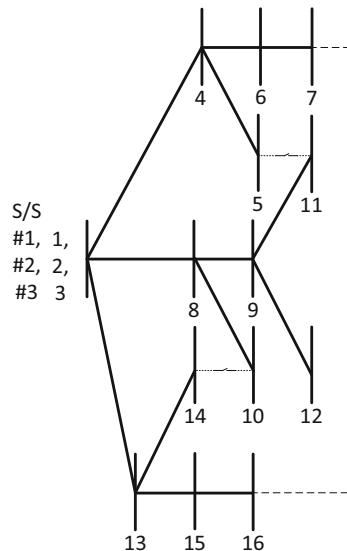


Fig. 17.3 Typical 16-bus system. Source: Authors' own creation

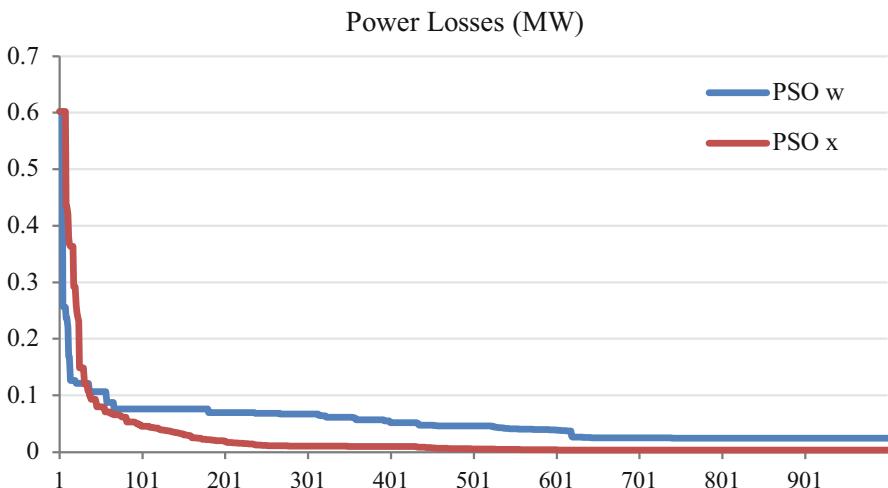


Fig. 17.4 Convergence comparison of inertia weight (PSO w) and constriction factor (PSO χ). Source: Authors' own creation

final solution. Therefore, this scheme proves to be more effective when contemplating the ODGP problem.

17.9 Reducing Local Minima Entrapment

17.9.1 Local PSO

In cases of multimodal or extremely complex environments, such as the ODGP problem, it has been proven that the swarm fragments because of complete diversity loss (Kennedy, 1999). This indicates that any further exploration of the solution space is no longer possible, and thus the particles are confined to exploit what information they already have and converge there. This hindrance can be attributed to the determination of the $P_g(t)$ term, in the third term of the sum in the second leg of Eq. (17.14). This term refers to the social knowledge gathered by each particle after communicating with other particles. If it is determined as the global best, after creating the global PSO version or GPSO, each particle communicates with all the other particles in the swarm. This provides great convergence capabilities but leads to the aforementioned problem, that is, lower exploration of the solution space and high local minima entrapment risk (Gkaidatzis, Bouhouras, Doukas, Sgouras, & Labridis, 2017).

However, if the social knowledge is diffused by using smaller overlapping groups of particles, this problem is being addressed. These smaller groups are called neighborhoods. There are various schemes under which these neighborhoods can be formed. One simple and effective schema has proven to be the ring topology (Hu & Eberhart, 2002; Mendes, Kennedy, & Neves, 2003), depicted in Fig. 17.5 and described mathematically in Eq. (17.20).

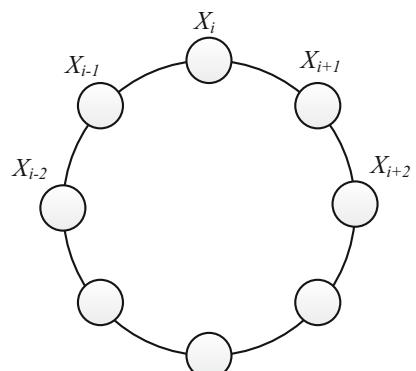
$$v_h(t+1) = \psi[v_h(t) + c_1 R_1(P_h(t) - X_h(t)) + c_2 R_2(P_l(t) - X_h(t))] \quad (17.22)$$

where:

$P_l(t)$:	Local best term
------------	-----------------

Fig. 17.5 Ring topology.

Source: Authors' own creation



Therefore, instead of using the global best, the local best is used, creating the local PSO version or LPSO.

17.9.2 uPSO

A comparison between the two, i.e., GPSO and LPSO, leads to the conclusion that the former has the advantage of fast convergence, which explains the reason of its broad use. However, it lacks in solution space exploration. Ergo the final solution will not be that optimal. The latter, although it provides greater exploration capabilities, lacks in convergence, leading to greater computation times. Therefore, a question is raised whether a PSO version can be developed by combining these two, enhancing their merits while avoiding their drawbacks. This has led to the development of the unified version of PSO, or uPSO, by Vrahatis and Pasropoulos (Parsopoulos & Vrahatis, 2002) which is described in the following equations:

$$V_h^G(t+1) = \psi[v_h(t) + c_1 R_1(P_h(t) - X_h(t)) + c_2 R_2(P_g(t) - X_h(t))] \quad (17.23)$$

$$V_h^L(t+1) = \psi[v_h(t) + c_1 R_1(P_h(t) - X_h(t)) + c_2 R_2(P_l(t) - X_h(t))] \quad (17.24)$$

with:

$$v_h(t+1) = \begin{cases} u_f R_3 V_h^G(t+1) + (1-u_f) V_h^L(t+1), & u \leq 0.5 \\ u_f V_h^G(t+1) + (1-u_f) R_3 V_h^L(t+1), & u > 0.5 \end{cases} \quad (17.25)$$

where $u_f \in [0, 1]$ is the unification factor that basically controls the combination schema of LPSO and GPSO and R_3 is an extra random variable uniformly distributed within $[0,1]$ that is applied alternatively on the GPSO and LPSO terms providing even further diversity and thus exploration to the technique. The rest of the parameters of the equation are the same as before.

17.9.3 Unification Factor Schemes

For the determination of the unification factor value, several processes have been proposed. They are categorized as swarm- and particle-level schemes, depending on the level in which the value assignment takes place. In the swarm level, the same value is provided for all particles. In the particle level, each particle presents its own scheme.

An approach could be an increasing unification factor, thus infusing exploration at the beginning of the process, by giving leverage to LPSO, over GPSO, and

gradually shifting the balance towards exploitation, by reversing the initial leverage (Parsopoulos & Vrahatis, 2005, 2007).

A few examples are as follows:

1. **Linear:** as in the inertia weight case, the unification factor is linearly increased, as described in the following equation:

$$u_f(t) = \frac{t}{T_{\max}} \quad (17.26)$$

2. **Modular:** the unification factor increases in repeated pattern, every z iterations, which is selected as a reasonable fraction of the total number of iteration steps:

$$u_f(t) = \frac{t \bmod(z+1)}{z} \quad (17.27)$$

3. **Exponential:** as stated, in this scheme the unification factor **increases** exponentially:

$$u_f(t) = \exp\left(\frac{t \cdot \log(2)}{T_{\max}}\right) - 1 \quad (17.28)$$

4. **Sigmoid:** the unification factor, in this scheme is increased gradually, i.e.:

$$u_f(t) = \frac{1}{1 + \exp\left[-\lambda\left(t - \frac{T_{\max}}{20}\right)\right]} \quad (17.29)$$

A few examples of particle-level schemes are the swarm partitioning (SP) and the self-adaptive (SA).

In the swarm partitioning scheme, the swarm is separated into nonoverlapping groups called partitions. In each partition a unification factor value is assigned, within the range of [0,1], and thus all particles belonging to the same partition share the same unification factor. Since the swarm is already divided into neighborhoods, due to LPSO, in order to avoid having particles with the same unification factor in the same neighborhood or, differently, increase the possibility to have particles with different unification factor values in the neighborhoods, an appropriate assignment scheme should be adopted. An approach would be to assign the first k particles to the partitions 1 to k , respectively, and then repeat the process for the next k particles and so on, thus having particle i in the $(1 + (i - 1) \bmod (k))$ partition.

In the self-adaptive scheme, the unification factor is considered as an additional dimension of the solution space and left to be determined by the optimization process itself.

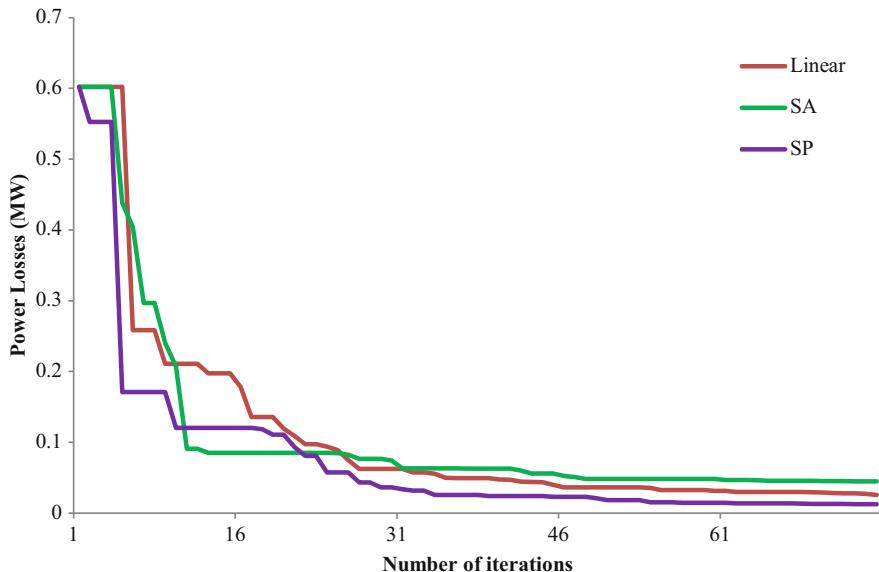


Fig. 17.6 Convergence of linear, SA, and SP uPSO version. Source: Authors' own creation

An indicative comparison between the linear, SP, and SA, when applied to the typical test 33-bus system, is demonstrated in Fig. 17.6 (Gkaidatzis, Bouhouras, Doukas, Sgouras, & Labridis, 2016; Gkaidatzis, Bouhouras, Sgouras, Doukas, & Labridis, 2016).

SP has been determined as the most promising scheme, providing both better final solutions and better convergence. This uPSO version is further compared to the two basic PSO versions, as shown in Fig. 17.7, when applied again in the typical test 33-bus system (Kashem, Ganapathy, Jasmon, & Buhari, 2000) depicted in Fig. 17.8. It becomes immediately apparent that uPSO indeed presents better performance both in terms of convergence and final solution than either GPSO or LPSO. From Fig. 17.7 additionally, the different exploration/exploitation ratios of the two basic PSO versions become also apparent, since GPSO, though converging fast enough, seems to be locked in a not optimal value, whereas LPSO, though slow at first, it reaches a far lower value which mean a far better solution.

17.10 Comparison with Other Heuristic Methods

In this section the evaluation of the PSO version as presented will be examined. More specifically, a comparative analysis among the PSO versions and how well they fare against other heuristic techniques such as GA, ABC, CS, and HS is analyzed. To that end, all the techniques have been given an ample time of 1000 iteration steps, i.e., they have been applied 1000 times. The techniques have been

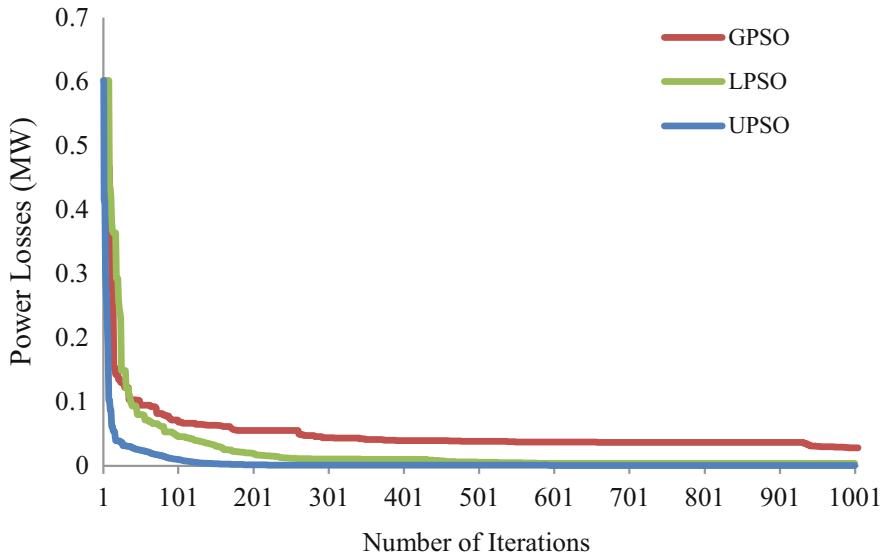


Fig. 17.7 Convergence comparison between uPSO, GPSO, and LPSO. Source: Authors' own creation

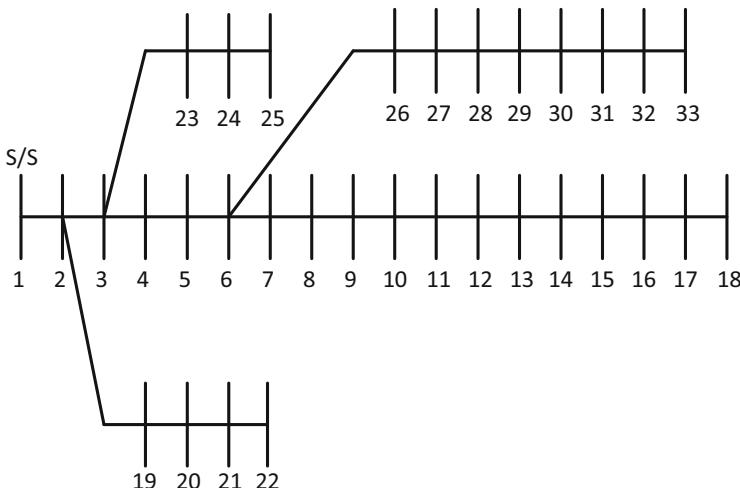


Fig. 17.8 Typical 33-bus system. Source: Authors' own creation

tested upon the IEEE-30 bus system (Yokoyama, Bae, Morita, & Sasaki, 1988), presented in Fig. 17.9.

In Table 17.1, results regarding the final solution reached by the various heuristic methods are shown. More specifically, the optimum loss achieved by each method and the respective percentage of loss reduction are shown. Moreover, the DG

Fig. 17.9 IEEE-30 bus system. Source: Authors' own creation

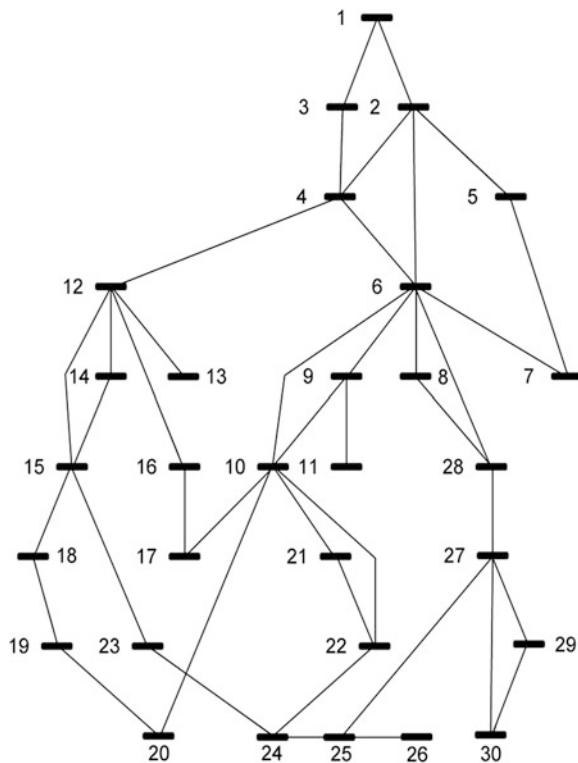


Table 17.1 Solution performance comparison of PSO with other heuristic techniques

Technique	Minimum power loss (kW)	Power loss reduction (%)	Total DG No.	Total DG installed (MVA) P+jQ
GPSO	792.85	67.56	8	52.94+j72.24
LPSO	795.06	67.47	8	53.02+j66.08
UPSO	742.10	69.63	12	53.66+j6.68
GA	1267.80	48.12	17	74.43+j35.46
ABC	761.10	68.86	13	50.12+j86.79
CS	947.67	61.22	19	55.71+j67.53
HS	824.24	66.27	10	52.86+j69.61

number for installation are presented and the total DG rated power in MVA, as they have been proposed by the best solution that each of the methods has reached. It is immediately evident that because of the adequate provided time, every method has reached a considerable reduction in losses and the deviations among their reached results are virtually insignificant. However, GA appears to be performing the least efficient than the rest. In Table 17.2 results related to the convergence performance of the various heuristic methods are presented. More specifically, the information provided concern the average trial execution time and the average iteration number

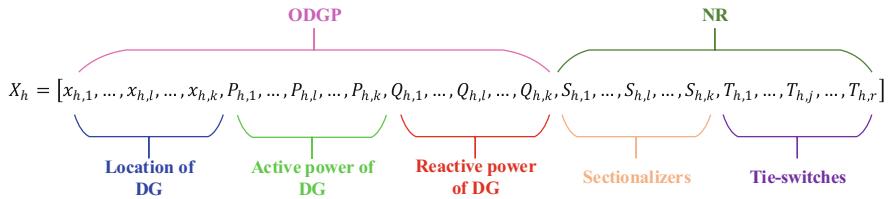


Fig. 17.10 Solution space formulation for combined ODGP-NR problem. Source: Authors' own creation

required for each method to reach within 10%, 1%, and 0.1% tolerance of the optimal solution proposed. For example, in uPSO, since its optimal solution amounts to 742.10 kW, this means that a 10% tolerance amounts to 816.31 kW loss. Regarding average execution time, requiring an execution time, less than 4 min, HS seems to perform the fastest. Additionally, another metric is introduced, and that is an iteration number required for each method to reach a certain amount of loss reduction. This amount is determined by the average loss reduction reached by the method that performs the least. This seems to be GA, and the amount is therefore set to 35.97%. Despite all methods evidently performing efficiently, it appears that the PSO versions fare rather more efficient than the rest, especially uPSO. With respect to convergence and iteration steps, for instance, they reached their final solution in the least amount of time overall.

Given that, an argument can be made that, though HS seems more efficient than uPSO, in terms of computation time, the latter can be applied for fewer iterations and ergo overcome this issue. This is further demonstrated in Fig. 17.10, where each method's average convergence of the 1000 trials is shown, and again in Fig. 17.11, where the same metric is presented, but zooming in particularly within the 10% iteration tolerance of the best performing technique, being uPSO.

Furthermore, as demonstrated in Fig. 17.12, the PSO versions, and uPSO in particular, have the least standard deviation convergence along the 1000 iteration sample. This means that during the 1000 trials, they do not deviate much from each other, demonstrating their robustness. This also indicates that even less trials are possible (Gkaidatzis, Doukas, Bouhouras, Sgouras, & Labridis, 2017).

17.11 PSO in Combination to ODGP with NR

In cases of fault occurrences and outages, network reconfiguration (NR) enables the distribution system operator (DSO) to rearrange the DN layout to continue to provide the same services. Over the last decades, this originally reliability-oriented mechanism has been also considered for loss reduction, since it was discovered that a DN layout modification could alter the loading of the DN lines (Bouhouras, Gkaidatzis, & Labridis, 2020).

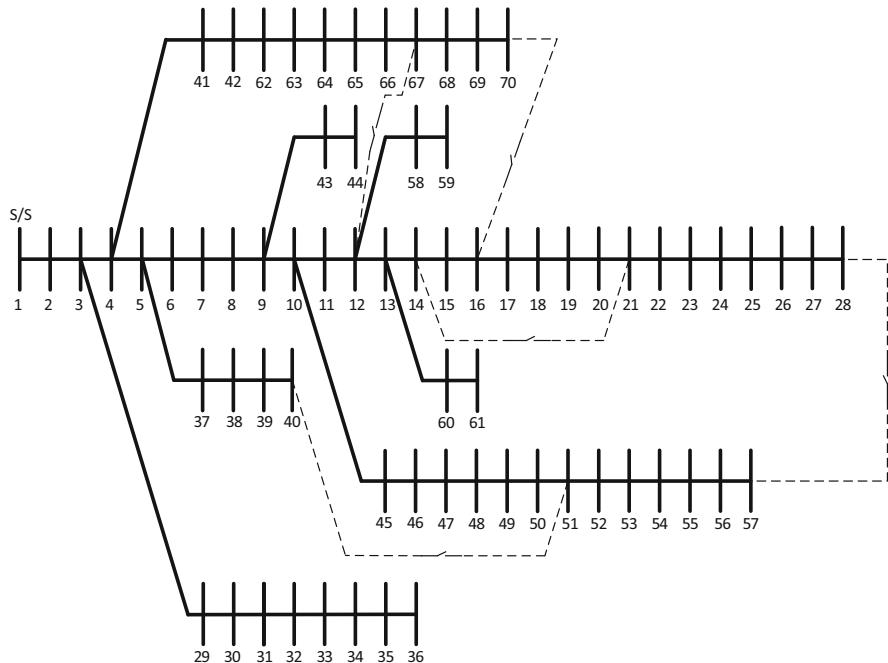


Fig. 17.11 Typical 69-bus system with tie switches. Source: Authors' own creation

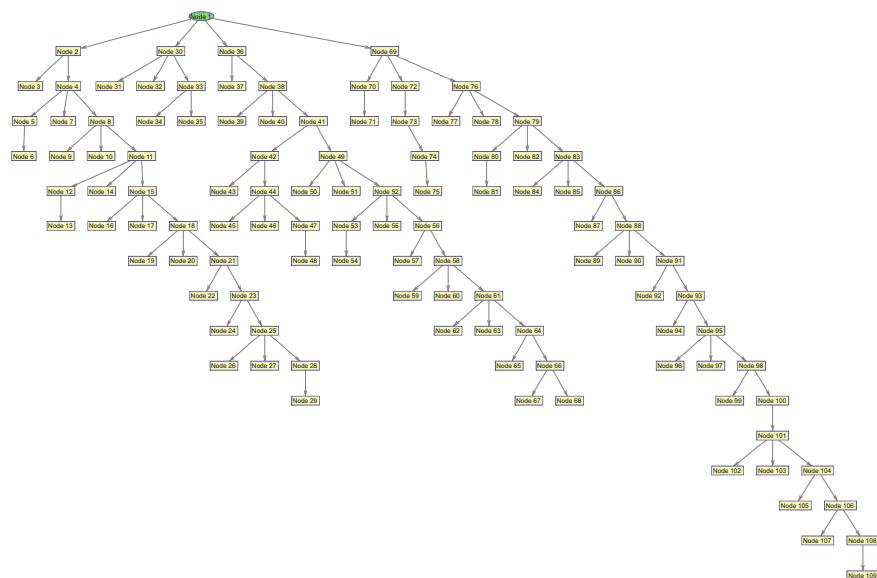


Fig. 17.12 EV examined DN. Source: Authors' own creation

Techniques for both the ODGP and NR problems have been regarded as efficient towards reducing power losses. The means used however in order to achieve this objective varies, since ODGP aims at the location and size of DG units to be installed, thus affecting the load composition of the DN, whereas NR aims at rearranging the DN layout, thus the topology of the DN (Bouhouras, Gkaidatzis, & Labridis, 2017).

When contemplated individually, the losses are reduced significantly. Therefore, there is high probability that a combination of them will have a considerable impact on the reached optimal solution with respect to the total amount of loss reduction. Let the highest possible loss reduction refer to the ideal 100%. Then, the order at which these problems are considered affects their contribution towards the solution of the overall problem. For example, when considering ODGP, a solution with 100% loss reduction could theoretically be yielded, in the ideal case where DG units are installed in every bus of the DN and generate the same amount of power requested by the load at each bus. In that particular case, to consider solving the NR towards loss reduction is rendered meaningless, since the objective is already achieved. However, if limited available DG units are considered during the ODGP solving process, as it is mostly the case, then further examining the NR could achieve more loss reduction and improve the solution even further.

If the reverse order is contemplated, that is, the NR is examined first and then the ODGP, then it presents great interest to examine the effect this would have on the siting and sizing of the DG, since the ODGP problem will now be solved for a modified DN, that is, with a modified layout, but with the same load composition. Additionally, it would be of great interest to investigate what the results would be, if both problems are solved simultaneously, that is, solving both ODGP and NR at the same time, therefore, formulating the solution space as it presented in Fig. 17.13.

The aforementioned analysis leads to three scenarios to be considered, with respect to the order of solving ODGP and NR:

- Scn#1: First NR is solved, then ODGP.
- Scn#2: First ODGP is solved, then NR.
- Scn#3: ODGP and NR are solved simultaneously.

The results when using the typical test 69-bus system (Soudi, 2013) are presented in Tables 17.2, 17.3, and 17.4. Scn#1 seems to bear the more advantageous results, due to the switching operations relying on the tie switches already in use. This leads also to less DG capacity required for loss minimization. In Scn#2, as previously analyzed, it seems hardly possible to reach a NR solution, particularly if the ODGP problem is solved rather efficiently towards high loss reduction. Finally, in Scn#3, where both problems are solved simultaneously, the total problem complexity seems to increase exponentially. The main reason behind this is the fact that the particle formulation is extended in order to accommodate both the ODGP and the NR variables, as demonstrated in the following equation:

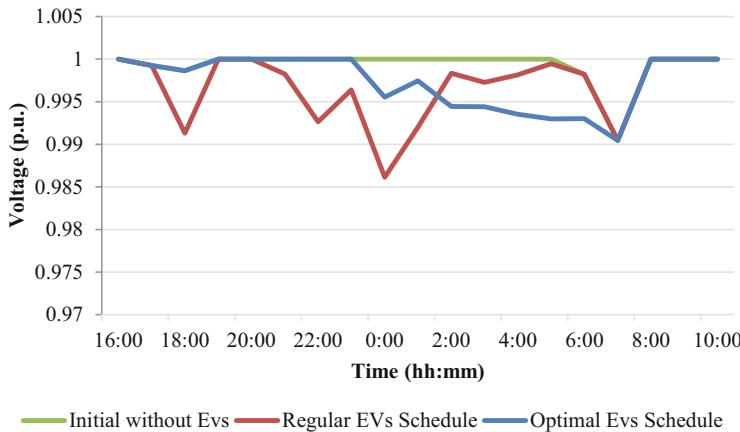


Fig. 17.13 Voltage profile of residence No. 79. Source: Authors' own creation

Table 17.2 Scn#1

NR results	Initial losses [kW]	Opened sectionalizers	Closed tie switches	Loss reduction %	Final losses [kW]
	229.8	14, 58, 62	Tie3–Tie5	54.7	104.1
ODGP results	Initial losses [kW]	DG location (bus number)	Active DG power [kW]	Reactive DG power [kVAr]	Loss reduction % and final losses [kW]
	104.1	5 9 12 22 40 53 56	901.7 241.6 427.4 338.3 0 1416.1 318.5	189.2 177.2 299.6 226.6 536.4 938.2 226.7	93.65% 6.6

$S_{h,l}$ and $T_{h,j}$, in contrast with $x_{h,l}$ which is an integer and $P_{h,l}$ and $Q_{h,l}$ that are real variables, are basically binary variables. PSO however has been mainly developed for continuous-valued solution spaces. In order to adjust to these new circumstances for these particular set of variables, the velocity and position equations are updated with the use of the following equations (Engelbrecht, 2007):

$$v_h(t+1) = \frac{1}{1 + e^{-p_{h(t)}}} \quad (17.30)$$

Table 17.3 Scn#2

ODGP results	Initial losses [kW]	DG location (bus number)	Active power of each DG unit [kW]	Reactive power of each DG unit [kVAr]	Loss reduction % and final losses [kW]
229.8	2	0	-53.2	97.35%	
	3	539	340	6.1	
	9	0	184.9		
	12	501.2	279.8		
	19	380.8	251.7		
	40	717	512		
	53	1674	1178.8		
NR results	Initial losses [kW]	Opened sectionalizers	Closed tie switches	Loss reduction %	Final losses [kW]
	6.1	-	-	0	6.1

Table 17.4 Scn#3

DG location (bus number)	Active power of each DG unit [kW]	Reactive power of each DG unit [kVAr]	Opened sectionalizers	Closed tie switches	Loss reduction %
					Final losses [kW]
57	2021.5	849.8	20, 42, 46, 58, 61	Tie1–tie5	68.28% 72.9

where:

$S_{h, i}$	Sectionalizer
$T_{h, j}$	To tie switch

$$x_h(t+1) = \begin{cases} 1, w < \frac{1}{1 + e^{-v_h(t+1)}} \\ 0, \text{otherwise} \end{cases} \quad (17.31)$$

where w is a randomly distributed variable within $[0,1]$, adding diversity in the process.

Due to this additional complexity, the algorithm seems unable to reach an effective solution. It still requires further examination, to establish if a better solution in this case is outweighed by the increased computational burden (Bouhouras, Andreou, Labridis, & Bakirtzis, 2010; Bouhouras & Labridis, 2012).

17.12 PSO in Optimal Charging Schedule of EVs

17.12.1 EV Integration in DN

The ever-growing integration of electric vehicles (EV) in LV networks (Nour, Ramadan, Ali, & Farkas, 2018) is expected to greatly affect the conventional load patterns both of the residential and commercial sectors. Charging the EVs will bear additional burden especially during the night. The EVs' owners are most commonly expected to connect and charge their EVs the moment they arrive at their homes, which usually occurs at the afternoon or evening. Moreover, they require to depart with the EV fully charged early in the morning of the next day (Antúnez, Franco, Rider, & Romero, 2016), especially during workdays. Thus, any random EV charging not controlled, or monitored, would cause an intense night load peak, leading this way to significant voltage drops and power quality issues. This issue can be tackled by employing optimized coordinated EV charging schedules, setting the time periods within which each individual EV will be charged with the goal to satisfy an objective function, for example, voltage profile improvement (Zheng, Song, Hill, & Meng, 2018), cost minimization (Thomas, Ioakimidis, Klonari, Vallée, & Deblecker, 2016; Wei, Li, & Cai, 2018), energy loss minimization (Zaidi, 2015), or combination of them.

17.12.2 Problem Formulation

In this case, as an objective function, the voltage improvement is in the epicenter, as described below (Bouhouras et al., 2018, 2019; Bouhouras, Gkaidatzis, & Labridis, 2018):

$$F_{vi} = \min \sum_{\Delta t=1}^{T_{\text{total}}} \sum_{m=1}^{l_n} |1 - V_m| \quad (17.32)$$

where:

Δt :	The time interval considered, e.g., an hour, half-hour, 15 min, etc.
T_{total} :	The total time period considered

uPSO is utilized to solve this problem and is applied to the DN, depicted in Fig. 17.12.

This DN constitutes a section of a Greek rural-residential DN. In each bus a residence is connected, and in each residence an EV is assigned in turn. For realistic purposes, variation of types of chargers, EV battery capacity, time of arrival, and state of charge at the time of arrival has been considered and is presented in Table 17.5.

Table 17.5 EV parameters

Characteristic	Type 1	Type 2	Type 3
EV capacity (kWh)	9.2	16	21.4
Number of nodes	54 (50% of the fleet)	43 (40% of the fleet)	11 (10% of the fleet)
Charger type (kW)	3.3 (1-phase AC)	7.4 (1-phase AC)	
Number of nodes	54 (50% of the residences)	54 (50% of the residences)	
SoC (%)	20–40		
Number of nodes	108 (all residences)		

Table 17.6 EV charging schedules for residence No. 79

EV at Residence No. 79	04/05/2014									
	20:00	21:00	22:00	23:00	00:00	01:00	02:00	03:00	04:00	05:00
Regular charge at arrival	3.3	3.3	3.3	3.3	2.97	0	0	0	0	0
Optimal schedule continuous	0	0	0	3.3	3.3	3.3	3.3	2.97	0	0
Optimal schedule intermittent	0	3.3	0	3.3	0	3.3	3.3	2.97	0	0

The DN technical constraints, as they have been described in the previous sections of this chapter, are taken into account, and the DG technical constraints are replaced by the EV ones, that is, the charger upper and lower operation limits. The penalty function formulation remains also the same with that described in previous sections of this chapter.

For practical purposes though, the EVs are assumed that they arrive no earlier than 20:00 and require to depart with a full battery until 06:00.

In that particular case, the solution space is a combination of binary and real variables, as presented in Eq. (17.31), where for each time step considered, e.g., 1 h, of the total amount of time examined, that is 20:00–06:00, it is to be determined if m EV will charge or not ($b_{i,m,\Delta t=k}$) and with how much power ($P_{i,m,\Delta t=k}$).

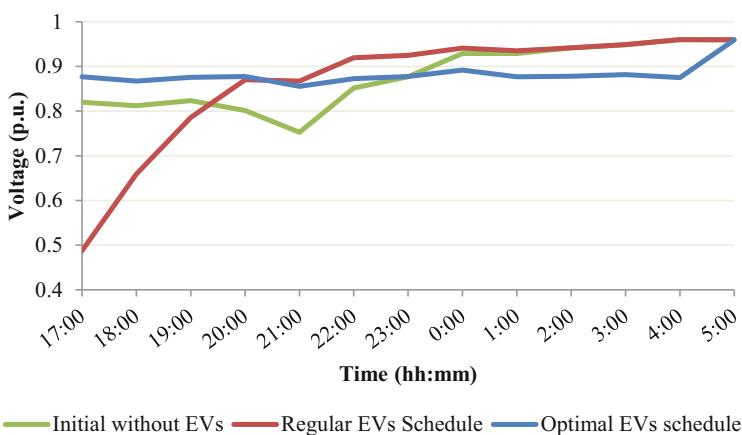
$$X_i = [b_{i,m,\Delta t=1} \cdot P_{i,m,\Delta t=1}, \dots, b_{i,m,\Delta t=k} \cdot P_{i,m,\Delta t=k}, \dots, b_{i,m,\Delta t=T_{\text{total}}} \cdot P_{i,m,\Delta t=T_{\text{total}}}] \quad (17.33)$$

This formulation provides the flexibility to examine both continuous and intermittent charging of the EVs, thus offering better EV portfolio management to either the DSO or the EV aggregator, in terms of day-ahead planning. The optimal schedule for the EVs of various residences is presented in Tables 17.6 and 17.7.

In Fig. 17.13 the results of the optimal schedule for residence No. 79 are presented, where the initial voltage profile without any EVs considered (green), a regular EV charging schedule without optimization considered (red), and an optimal

Table 17.7 EV charging schedules for residence No. 109

EV at Residence No. 109	04/05/2014										
	20:00	21:00	22:00	23:00	00:00	01:00	02:00	03:00	04:00	05:00	
Regular charge at arrival	7.4	5.1	0	0	0	0	0	0	0	0	
Optimal schedule continuous	0	0	0	0	0	0	0	0	7.4	5.1	
Optimal schedule intermittent	7.4	7.4	0	0	0	0	0	0	5.1	0	

**Fig. 17.14** Voltage profile of residence No. 109. Source: Authors' Own Creation

EV schedule (blue) are compared. A significant improvement is shown since voltage drop is decreased and formed in a smoother manner, i.e., not so abruptly. The same is evident for residence No. 109, i.e., the furthest residence from the substation, ergo the one with the most voltage drop, as shown in Fig. 17.14.

17.13 Conclusion and Discussion

In this chapter the applications of PSO in various modern power system problems have been presented, mainly in the solution of ODGP, either alone or in conjunction with NR and optimal EV charging schedules. PSO has been concluded as quite effective to all problems addressed. Moreover, since all of the problems have their own particular requirements, PSO has shown a considerable range and variety of applications, adding to its broad utilization and wide adoption by many different

topics, fields, and principles. Provided the dynamic and wide nature of the power system research sector, there is virtually no limitation as to where else PSO can be applied. A few examples might be in the ever-growing microgrid sector (Hossain, Pota, Squartini, & Abdou, 2019), reliability (Yang, Zhang, Ma, Zhou, & Yang, 2019), battery energy storage systems implementation (Yang, Gong, Ma, Wang, & Dong, 2020), and demand-side management and in particular demand response portfolio management (Sood, Ali, & Khan, 2020).

Acknowledgments This research has received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

References

- Abdmouleh, Z., Gastli, A., Ben-Brahim, L., Haouari, M., & Al-Emadi, N. A. (2017). Review of optimization techniques applied for the integration of distributed generation from renewable energy sources. *Renewable Energy*, 113, 266–280.
- Ali, E. S., Abd Elazim, S. M., & Abdelaziz, A. Y. (2016). Ant lion optimization algorithm for renewable distributed generations. *Energy*, 116, 445–458.
- Alsac, O., Bright, J., Prais, M., & Stott, B. (1990). Further developments in LP-based optimal power flow. *IEEE Transactions on Power Systems*, 5(3), 697–711.
- Antúnez, C. S., Franco, J. F., Rider, M. J., & Romero, R. (2016). A new methodology for the optimal charging coordination of electric vehicles considering vehicle-to-grid technology. *IEEE Transactions on Sustainable Energy*, 7(2), 596–607.
- Arya, L. D., Koshti, A., & Choube, S. C. (2012). Distributed generation planning using differential evolution accounting voltage stability consideration. *International Journal of Electrical Power & Energy Systems*, 42(1), 196–207.
- Atwa, Y. M., & El-Saadany, E. F. (2011). Probabilistic approach for optimal allocation of wind-based distributed generation in distribution systems. *IET Renewable Power Generation*, 5(1), 79–88.
- Bansal, R. C. (2003). Bibliography on the fuzzy set theory applications in power systems (1994–2001). *IEEE Transactions on Power Systems*, 18(4), 1291–1299.
- Bansal, R. C. (2005). Optimization methods for electric power systems: An overview. *International Journal of Emerging Electric Power Systems*, 2(1), 1021.
- Bouhouras, A. S., Andreou, G. T., Labridis, D. P., & Bakirtzis, A. G. (2010). Selective automation upgrade in distribution networks towards a smarter grid. *IEEE Transactions on Smart Grid*, 1 (3), 278–285.
- Bouhouras, A. S., Gkaidatzis, P. A., & Labridis, D. P. (2020). Network reconfiguration in modern power distribution networks. In *Handbook of optimization in electric power distribution systems* (pp. 219–255). Cham: Springer.
- Bouhouras, A. S., & Labridis, D. P. (2012). Influence of load alterations to optimal network configuration for loss reduction. *Electric Power Systems Research*, 86, 17–27.
- Chatzivasileiadis, S. (2018). *Lecture notes on optimal power flow (OPF)*. <https://arxiv.org/abs/1811.00943>.
- Chen, S., Chen, B., & Fath, B. D. (2015). Assessing the cumulative environmental impact of hydropower construction on river systems based on energy network model. *Renewable and Sustainable Energy Reviews*, 42, 78–92.
- Civanlar, S., Grainger, J. J., Yin, H., & Lee, S. S. H. (1988). Distribution feeder reconfiguration for loss reduction. *IEEE Transactions on Power Delivery*, 3(3), 1217–1223.

- Dai, X. Z., He, D., Fan, L. L., Li, N. H., & Chen, H. (1999). Improved ANN σ th-order inverse TCSC controller for enhancing power system transient stability. *IEE Proceedings-Generation, Transmission and Distribution*, 146(6), 550–556.
- Defourny, B., & Terlaky, T. (2015). *Modeling and optimization: Theory and applications*. Cham: Springer.
- Del Valle, Y., Venayagamoorthy, G. K., Mohagheghi, S., Hernandez, J. C., & Harley, R. G. (2008). Particle swarm optimization: Basic concepts, variants and applications in power systems. *IEEE Transactions on Evolutionary Computation*, 12(2), 171–195.
- Delfanti, M., Falabretti, D., & Merlo, M. (2013). Dispersed generation impact on distribution network losses. *Electric Power Systems Research*, 97, 10–18.
- Dillon, T. S., Edwin, K. W., Kochs, H. D., & Taud, R. J. (1978). Integer programming approach to the problem of optimal unit commitment with probabilistic reserve determination. *IEEE Transactions on Power Apparatus and Systems*, 6, 2154–2166.
- Eberhart, R., & Kennedy, J. (1995, October). A new optimizer using particle swarm theory. In *MHS '95. Proceedings of the sixth international symposium on micro machine and human science* (pp. 39–43). New York: IEEE.
- Eberhart, R. C., & Shi, Y. (2000, July). Comparing inertia weights and constriction factors in particle swarm optimization. In *Proceedings of the 2000 congress on evolutionary computation. CEC00 (Cat. No. 00TH8512)* (Vol. 1, pp. 84–88). New York: IEEE.
- Engelbrecht, A. P. (2007). *Computational intelligence: An introduction*. New York: John Wiley & Sons.
- Esmaili, M. (2013). Placement of minimum distributed generation units observing power losses and voltage stability with network constraints. *IET Generation Transmission and Distribution*, 7(8), 813–821.
- Gautam, D., & Mithulanthan, N. (2007). Optimal DG placement in deregulated electricity market. *Electric Power Systems Research*, 77(12), 1627–1636.
- Georgilakis, P. S., & Hatziargyriou, N. D. (2013). Optimal distributed generation placement in power distribution networks: Models, methods, and future research. *IEEE Transactions on Power Systems*, 28(3), 3420–3428.
- Hossain, M. A., Pota, H. R., Squartini, S., & Abdou, A. F. (2019). Modified PSO algorithm for real-time energy management in grid-connected microgrids. *Renewable Energy*, 136, 746–757.
- Hu, X., & Eberhart, R. (2002, May). Multiobjective optimization using dynamic neighborhood particle swarm optimization. In *Proceedings of the 2002 Congress on evolutionary computation. CEC'02 (Cat. No. 02TH8600)* (Vol. 2, pp. 1677–1681). New York: IEEE.
- Hung, D. Q., & Mithulanthan, N. (2011). Multiple distributed generator placement in primary distribution networks for loss reduction. *IEEE Transactions on Industrial Electronics*, 60(4), 1700–1708.
- Hung, D. Q., Mithulanthan, N., & Bansal, R. C. (2010). Analytical expressions for DG allocation in primary distribution networks. *IEEE Transactions on Energy Conversion*, 25(3), 814–820.
- Hyedt, G. T., & Grady, W. M. (1983). Optimal Var sitting using linear load flow formulation. *IEEE Transactions on Power Apparatus and Systems*, 92(5), 1214–1222.
- Iba, K. (1994). Reactive power optimization by genetic algorithm. *IEEE Transactions on Power Systems*, 9(2), 685–692.
- Ismail, M. S., Moghavvemi, M., & Mahlia, T. M. I. (2013). Current utilization of microturbines as a part of a hybrid system in distributed generation technology. *Renewable and Sustainable Energy Reviews*, 21, 142–152.
- Jordehi, A. R. (2016). Allocation of distributed generation units in electric power systems: A review. *Renewable and Sustainable Energy Reviews*, 56, 893–905.
- Kashem, M. A., Ganapathy, V., Jasmon, G. B., & Buhari, M. I. (2000, April). A novel method for loss minimization in distribution networks. In *DRPT2000. International conference on electric utility deregulation and restructuring and power technologies. Proceedings (Cat. No. 00EX382)* (pp. 251–256). New York: IEEE.

- Keane, A., & O'Malley, M. (2005). Optimal allocation of embedded generation on distribution networks. *IEEE Transactions on Power Systems*, 20(3), 1640–1646.
- Kennedy, J. (1999, July). Small worlds and mega-minds: Effects of neighborhood topology on particle swarm performance. In *Proceedings of the 1999 congress on evolutionary computation-CEC99 (Cat. No. 99TH8406)* (Vol. 3, pp. 1931–1938). New York: IEEE.
- Khalesi, N., Rezaei, N., & Haghifam, M. R. (2011). DG allocation with application of dynamic programming for loss reduction and reliability improvement. *International Journal of Electrical Power & Energy Systems*, 33(2), 288–295.
- Kothari, D. P. (1989). Optimal stochastic hydrothermal scheduling using nonlinear programming technique. Presented Australian Society, Melbourne (Australia), pp. 335–344.
- Kothari, D. P. (2012, March). Power system optimization. In *2012 2nd National conference on computational intelligence and signal processing (CISP)* (pp. 18–21). New York: IEEE.
- Lu, F. C., & Hsu, Y. Y. (1995). Reactive power/voltage control in a distribution substation using dynamic programming. *IEE Proceedings-Generation, Transmission and Distribution*, 142(6), 639–645.
- Mendes, R., Kennedy, J., & Neves, J. (2003, April). Watch thy neighbor or how the swarm can learn from its environment. In *Proceedings of the 2003 IEEE Swarm Intelligence Symposium. SIS'03 (Cat. No. 03EX706)* (pp. 88–94). New York: IEEE.
- Mohammadi, M., Rozbahani, A. M., & Montazeri, M. (2016). Multi criteria simultaneous planning of passive filters and distributed generation simultaneously in distribution system considering nonlinear loads with adaptive bacterial foraging optimization approach. *International Journal of Electrical Power & Energy Systems*, 79, 253–262.
- Moravej, Z., & Akhlaghi, A. (2013). A novel approach based on cuckoo search for DG allocation in distribution network. *International Journal of Electrical Power & Energy Systems*, 44(1), 672–679.
- Nour, M., Ramadan, H., Ali, A., & Farkas, C. (2018, February). Impacts of plug-in electric vehicles charging on low voltage distribution network. In *2018 International conference on innovative trends in computer engineering (ITCE)* (pp. 357–362). New York: IEEE.
- Paliwal, P., Patidar, N. P., & Nema, R. K. (2014). Planning of grid integrated distributed generators: A review of technology, objectives and techniques. *Renewable and Sustainable Energy Reviews*, 40, 557–570.
- Palz, W. (Ed.). (2013). *Solar power for the world: What you wanted to know about photovoltaics* (Vol. 4). Boca Raton: CRC Press.
- Parsopoulos, K. E., & Vrahatis, M. N. (2002). Particle swarm optimization method for constrained optimization problems. *Intelligent Technologies—Theory and Application: New Trends in Intelligent Technologies*, 76(1), 214–220.
- Parsopoulos, K. E., & Vrahatis, M. N. (2005, August). Unified particle swarm optimization for solving constrained engineering optimization problems. In *International conference on natural computation* (pp. 582–591). Berlin, Heidelberg: Springer.
- Parsopoulos, K. E., & Vrahatis, M. N. (2007). Parameter selection and adaptation in unified particle swarm optimization. *Mathematical and Computer Modelling*, 46(1–2), 198–213.
- Partanen, J. (1990). A modified dynamic programming algorithm for sizing, locating and timing of feeder reinforcements. *IEEE Transactions on Power Delivery*, 5(1), 277–283.
- Prakash, D. B., & Lakshminarayana, C. (2016). Multiple DG placements in distribution system for power loss reduction using PSO Algorithm. *Procedia Technology*, 25, 785–792.
- Rao, R. S., Ravindra, K., Satish, K., & Narasimham, S. V. L. (2012). Power loss minimization in distribution system using network reconfiguration in the presence of distributed generation. *IEEE Transactions on Power Systems*, 28(1), 317–325.
- Rau, N. S., & Wan, Y. H. (1994). Optimum location of resources in distributed planning. *IEEE Transactions on Power Systems*, 9(4), 2014–2020.
- Satoh, T., & Nara, K. (1991). Maintenance scheduling by using simulated annealing method (for power plants). *IEEE Transactions on Power Systems*, 6(2), 850–857.

- Schwaegerl, C., Bollen, M. H. J., Karoui, K., & Yagmur, A. (2005, June). Voltage control in distribution systems as a limitation of the hosting capacity for distributed energy resources. In *IET CIRED conference* (pp. 1–5).
- Seker, A. A., & Hocaoglu, M. H. (2013, November). Artificial Bee Colony algorithm for optimal placement and sizing of distributed generation. In *2013 8th International Conference on Electrical and Electronics Engineering (ELECO)* (pp. 127–131). New York: IEEE.
- Shi, Y., & Eberhart, R. (1998, May). A modified particle swarm optimizer. In *1998 IEEE international conference on evolutionary computation proceedings. IEEE world congress on computational intelligence (Cat. No. 98TH8360)* (pp. 69–73). New York: IEEE.
- Singh, D., Misra, R. K., & Singh, D. (2007). Effect of load models in distributed generation planning. *IEEE Transactions on Power Systems*, 22(4), 2204–2212.
- Song, Y. H. (2013). *Modern optimisation techniques in power systems* (Vol. 20). New York: Springer Science & Business Media.
- Sood, V. K., Ali, M. Y., & Khan, F. (2020). Energy management system of a microgrid using particle swarm optimization (PSO) and communication system. In *Microgrid: Operation, control, monitoring and protection* (pp. 263–288). Singapore: Springer.
- Soroudi, A., & Ehsan, M. (2011). Efficient immune-GA method for DNOs in sizing and placement of distributed generation units. *European Transactions on Electrical Power*, 21(3), 1361–1375.
- Soudi, S. (2013). Distribution system planning with distributed generations considering benefits and costs. *International Journal of Modern Education and Computer Science*, 5(9), 45.
- Sultana, U., Khairuddin, A. B., Mokhtar, A. S., Zareen, N., & Sultana, B. (2016). Grey wolf optimizer based placement and sizing of multiple distributed generation in the distribution system. *Energy*, 111, 525–536.
- Sun, D. I., Ashley, B., Brewer, B., Hughes, A., & Tinney, W. F. (1984). Optimal power flow by Newton approach. *IEEE Transactions on Power Apparatus and Systems*, 10, 2864–2880.
- Thomas, D., Ioakimidis, C. S., Klonari, V., Vallée, F., & Deblecker, O. (2016, October). Effect of electric vehicles' optimal charging-discharging schedule on a building's electricity cost demand considering low voltage network constraints. In *2016 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe)* (pp. 1–6). New York: IEEE.
- Tu, J., Yin, Z., & Xu, Y. (2018). Study on the evaluation index system and evaluation method of voltage stability of distribution network with high DG penetration. *Energies*, 11(1), 79.
- Wei, Z., Li, Y., & Cai, L. (2018). Electric vehicle charging scheme for a park-and-charge system considering battery degradation costs. *IEEE Transactions on Intelligent Vehicles*, 3(3), 361–373.
- Wen, F., & Chang, C. S. (1997). A tabu search approach to fault section estimation in power systems. *Electric Power Systems Research*, 40(1), 63–73.
- Willis, H. L. (2000, July). Analytical methods and rules of thumb for modeling DG-distribution interaction. In *2000 Power engineering society summer meeting (Cat. No. 00CH37134)* (Vol. 3, pp. 1643–1644). New York: IEEE.
- World Wind Energy Association. (2014). *Half-year Report. Bonn, Germany*.
- Yang, H., Gong, Z., Ma, Y., Wang, L., & Dong, B. (2020). Optimal two-stage dispatch method of household PV-BESS integrated generation system under time-of-use electricity price. *International Journal of Electrical Power & Energy Systems*, 123, 106244.
- Yang, H., Zhang, Y., Ma, Y., Zhou, M., & Yang, X. (2019). Reliability evaluation of power systems in the presence of energy storage system as demand management resource. *International Journal of Electrical Power & Energy Systems*, 110, 1–10.
- Yokoyama, R., Bae, S. H., Morita, T., & Sasaki, H. (1988). Multiobjective optimal generation dispatch based on probability security criteria. *IEEE Transactions on Power Systems*, 3(1), 317–324.
- Zaidi, A. H. (2015, June). Optimal electric vehicle load management for minimization of losses. In *2015 Power generation system and renewable energy technologies (PGSRET)* (pp. 1–6). New York: IEEE.

- Zheng, Y., Song, Y., Hill, D. J., & Meng, K. (2018). Online distributed MPC-based optimal scheduling for EV charging stations in distribution systems. *IEEE Transactions on Industrial Informatics*, 15(2), 638–649.
- Zhu, J. Z., & Irving, M. R. (1996). Combined active and reactive dispatch with multiple objectives using an analytic hierarchical process. *IEE Proceedings-Generation, Transmission and Distribution*, 143(4), 344–352.

Additional Readings

- Bouhouras, A. S., Gkaidatzis, P. A., & Labridis, D. P. (2017, June). Optimal application order of network reconfiguration and ODGP for loss reduction in distribution networks. In *2017 IEEE International conference on environment and electrical engineering and 2017 IEEE Industrial and commercial power systems Europe (EEEIC/I&CPS Europe)* (pp. 1–6). New York: IEEE.
- Bouhouras, A. S., Gkaidatzis, P. A., & Labridis, D. P. (2018). Optimal distributed generation placement problem for power and energy loss minimization. In *Electric distribution network planning* (pp. 215–251). Singapore: Springer.
- Bouhouras, A. S., Gkaidatzis, P. A., Panapakidis, I., Tsikalos, A., Labridis, D. P., & Christoforidis, G. C. (2019, April). A PSO based optimal EVs charging utilizing BESSs and PVs in buildings. In *2019 IEEE 13th International conference on compatibility, power electronics and power engineering (CPE-POWERENG)* (pp. 1–6). New York: IEEE.
- Bouhouras, A. S., Parisse, C., Gkaidatzis, P. A., Sgouras, K. I., Doukas, D. I., & Labridis, D. P. (2016, June). Energy loss reduction in distribution networks via ODGP. In *2016 13th International Conference on the European Energy Market (EEM)* (pp. 1–5). New York: IEEE.
- Bouhouras, A. S., Sgouras, K. I., Gkaidatzis, P. A., & Labridis, D. P. (2016). Optimal active and reactive nodal power requirements towards loss minimization under reverse power flow constraint defining DG type. *International Journal of Electrical Power & Energy Systems*, 78, 445–454.
- Bouhouras, A. S., Tsikalos, A., Emmanouil, E., Christoforidis, G. C., Gkaidatzis, P. A., Labridis, D. P., & Papagiannis, G. K. (2018, June). A UPSO based optimal BEVs charging for voltage quality improvement. In *2018 IEEE International energy conference (ENERGYCON)* (pp. 1–6). New York: IEEE.
- Doukas, D. I., Gkaidatzis, P. A., Bouhouras, A. S., Sgouras, K. I., & Labridis, D. P. (2016). *On reverse power flow modelling in distribution grids*.
- Gkaidatzis, P. A., Bouhouras, A. S., Doukas, D. I., Sgouras, K. I., & Labridis, D. P. (2016, June). Application and evaluation of UPSO to ODGP in radial Distribution Networks. In *2016 13th International conference on the European energy market (EEM)* (pp. 1–5). New York: IEEE.
- Gkaidatzis, P. A., Bouhouras, A. S., Doukas, D. I., Sgouras, K. I., & Labridis, D. P. (2017). Load variations impact on optimal DG placement problem concerning energy loss reduction. *Electric Power Systems Research*, 152, 36–47.
- Gkaidatzis, P. A., Bouhouras, A. S., Sgouras, K. I., Doukas, D. I., Christoforidis, G. C., & Labridis, D. P. (2019). Efficient RES penetration under optimal distributed generation placement approach. *Energies*, 12(7), 1250.
- Gkaidatzis, P. A., Bouhouras, A. S., Sgouras, K. I., Doukas, D. I., & Labridis, D. P. (2016). *Optimal distributed generation placement problem for renewable and DG units: An innovative approach*.
- Gkaidatzis, P. A., Doukas, D. I., Bouhouras, A. S., Sgouras, K. I., & Labridis, D. P. (2017). Impact of penetration schemes to optimal DG placement for loss minimisation. *International Journal of Sustainable Energy*, 36(5), 473–488.
- Gkaidatzis, P. A., Doukas, D. I., Labridis, D. P., & Bouhouras, A. S. (2017, June). Comparative analysis of heuristic techniques applied to ODGP. In *2017 IEEE International conference on*

- environment and electrical engineering and 2017 IEEE industrial and commercial power systems Europe (IEEEIC/I&CPS Europe)* (pp. 1–6). New York: IEEE.
- Ntomialis, S. P., Nakas, G. A., Gkaidatzis, P. A., Labridis, D. P., Bouhouras, A. S., & Christoforidis, G. C. (2018, June). Optimal siting of BESS in distribution networks under high wind power penetration. In *2018 IEEE International energy conference (ENERGYCON)* (pp. 1–6). New York: IEEE.
- Raptis, D. A., Periandros, P. S., Gkaidatzis, P. A., Bouhouras, A. S., & Labridis, D. P. (2018, September). Optimal siting of BESS in distribution networks under high PV penetration. In *2018 53rd International Universities power engineering conference (UPEC)* (pp. 1–6). New York: IEEE.
- Sgouras, K. I., Bouhouras, A. S., Gkaidatzis, P. A., Doukas, D. I., & Labridis, D. P. (2017). Impact of reverse power flow on the optimal distributed generation placement problem. *IET Generation Transmission and Distribution*, 11(18), 4626–4632.