Springer Tracts in Nature-Inspired Computing

Xin-She Yang
Yu-Xin Zhao *Editors*

Nature-Inspired Computation in Navigation and Routing Problems

Algorithms, Methods and Applications

Springer

Springer Tracts in Nature-Inspired Computing

The book series is aimed at providing an exchange platform for researchers to summarize the latest research and developments related to nature-inspired computing in the most general sense. It includes analysis of nature-inspired algorithms and techniques, inspiration from natural and biological systems, computational mechanisms and models that imitate them in various fields, and the applications to solve real-world problems in different disciplines. The book series addresses the most recent innovations and developments in nature-inspired computation, algorithms, models and methods, implementation, tools, architectures, frameworks, structures, applications associated with bio-inspired methodologies and other relevant areas.

The book series covers the topics and fields of Nature-Inspired Computing, Bio-inspired Methods, Swarm Intelligence, Computational Intelligence, Evolutionary Computation, Nature-Inspired Algorithms, Neural Computing, Data Mining, Artificial Intelligence, Machine Learning, Theoretical Foundations and Analysis, and Multi-Agent Systems. In addition, case studies, implementation of methods and algorithms as well as applications in a diverse range of areas such as Bioinformatics, Big Data, Computer Science, Signal and Image Processing, Computer Vision, Biomedical and Health Science, Business Planning, Vehicle Routing and others are also an important part of this book series.

The series publishes monographs, edited volumes and selected proceedings.

More information about this series at http://www.springer.com/series/16134

Xin-She Yang · Yu-Xin Zhao
Editors

Nature-Inspired Computation in Navigation and Routing Problems

Algorithms, Methods and Applications

Springer

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

Yu-Xin Zhao
College of Automation
Harbin Engineering University
Harbin, Heilongjiang, China

ISSN 2524-552X ISSN 2524-5538 (electronic)
Springer Tracts in Nature-Inspired Computing
ISBN 978-981-15-1841-6 ISBN 978-981-15-1842-3 (eBook)
https://doi.org/10.1007/978-981-15-1842-3

This Springer imprint is published by the registered company Springer Nature Singapore Pte Ltd.
The registered company address is: 152 Beach Road, #21-01/04 Gateway East, Singapore 189721, Singapore

Preface

Many applications involve a certain degree of navigation and routing, and examples are path planning for delivery vehicles, airline route planning, scheduling and underwater navigation. Most vehicle routing problems are non-deterministic polynomial-time (NP) hard combinatorial optimization problems, and a well-known example is the travelling salesman problem (TSP). There are no efficient algorithms for solving NP-hard problems. Though certain exact methods may manage to solve problem instances of small problem size, any problem instances of moderate sizes or large sizes become impractical to solve. This means that the main alternatives are approximation methods and heuristic methods. Approximation methods such as linear programming relaxation may give some indication of the approximation solutions and how far the potential solutions may be from the optimal solutions, there is no guarantee that an optimal solution can be found by such approximations.

Navigation and routing problems can be formulated as optimization problems, and they can in principle be solved by using optimization techniques. In recent years, heuristic methods and metaheuristic approaches start to show promising results. Among such metaheuristic algorithms are nature-inspired algorithms for optimization, and they are population-based algorithms that are flexible and yet efficient enough to obtain good solutions in a practically acceptable timescale. There are some significant developments concerning nature-inspired algorithms, especially those based on swarm intelligence, in the last decades, and new results appear almost every week in the last few years.

Thus, this edited book strives to provide a timely review and summary of the state-of-the-art developments in nature-inspired computation, with emphasis on navigation and routing problems. The topics include navigation in animals, navigation algorithms, nature-inspired algorithms, travelling salesman problem, vehicle routing problems, underwater vehicle path planning, flow shop scheduling, industrial crane path planning, mobile robot path planning, smartphone indoor localization and others. Therefore, this book will be useful to students, lecturers, researchers and professionals in computer science, management science, logistics, operations research and industrial applications.

We would like to thank the independent reviewers for their constructive comments on the manuscripts of all the chapters during the peer review process. We also would like to thank Dr Aninda Bose and staff at Springer for their help and professionalism.

London, UK
Harbin, China
September 2019

Xin-She Yang
Yu-Xin Zhao

Contents

Editors and Contributors

About the Editors

Dr. Xin-She Yang obtained his DPhil in Applied Mathematics from the University of Oxford, then worked at Cambridge University and UK's National Physical Laboratory as a Senior Research Scientist. Now he is Reader in modelling and optimization at Middlesex University London. He is also the IEEE CIS task force chair for business intelligence and knowledge management. With more than 250 research publications and 25 books, he has been named as a highly cited researcher by Clarivate Analytics (formerly Thomson Reuters Web of Science) for four consecutive years (2016, 2017, 2018 and 2019). His main research interests include computational intelligence, nature-inspired computing, applied mathematics, machine learning, modelling and simulation as well as data mining.

Dr. Yu-Xin Zhao received his Ph.D. degree in Navigation, Guidance, and Control from Harbin Engineering University (HEU) in 2005 and completed postdoctoral research in Control Science and Engineering at Harbin Institute of Technology (HIT) in 2008. He is currently the Dean and a Professor at the College of Automation, Harbin Engineering University. His research interests include artificial intelligence, filtering theory, navigation theory and application, as well as intelligent transportation systems. He has published more than 100 papers, including more than 40 international journal papers in these areas. He also has 18 national patents. Dr. Zhao has won numerous awards, including the Young ChangJiang Scholar (2018) and LongJiang Scholar. He has served the academic community in various capacities, including Fellow of IET, senior member of IEEE, a member of the Royal Institute of Navigation, and a Fellow of China Navigation Institute.

Contributors

Rafael Alexandrou Frederick University, Nicosia, Cyprus

Asma Chakri Laboratory of Energy and Mechanical Engineering, Faculty of Engineering Sciences, University M'Hamed Bougara of Boumerdes (UMBB), Boumerdes, Algeria;
Industrial Mechanics Laboratory, Department of Mechanical Engineering, University Badji Mokhtar of Annaba (UBMA), Annaba, Algeria

Sankha Deb Department of Mechanical Engineering, Indian Institute of Technology Kharagpur, Kharagpur, India

Javier Del Ser TECNALIA, Basque Research and Technology Alliance (BRTA), Derio, Spain;
University of the Basque Country (UPV/EHU), Bilbao, Spain

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

Ri-Xu Hao College of Automation, Harbin Engineering University, Harbin, China

Rabia Khelif Industrial Mechanics Laboratory, Department of Mechanical Engineering, University Badji Mokhtar of Annaba (UBMA), Annaba, Algeria

Andreas Konstantinidis Frederick University, Nicosia, Cyprus

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

Atul Mishra Department of Mechanical Engineering, Indian Institute of Technology Kharagpur, Kharagpur, India

Eneko Osaba TECNALIA, Basque Research and Technology Alliance (BRTA), Derio, Spain

Aziz Ouaarab Higher School of Technology, Essaouira, Cadi Ayyad University of Marrakesh, Marrakesh, Morocco

Harris Papadopoulos Frederick University, Nicosia, Cyprus

Haroun Ragueb Laboratory of Energy and Mechanical Engineering, Faculty of Engineering Sciences, University M'Hamed Bougara of Boumerdes (UMBB), Boumerdes, Algeria

Amar Skendraoui Department of Technology, Higher School of Industrial Technologies of Annaba (ESTI—Annaba), Annaba, Algeria

Ömür Tosun Department of Management Information Systems, Faculty of Applied Sciences, Akdeniz University, Antalya, Turkey

De-Quan Yang College of Automation, Harbin Engineering University, Harbin, China

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

Yu-Xin Zhao College of Automation, Harbin Engineering University, Harbin, China

Chapter 1
Navigation, Routing and Nature-Inspired Optimization

Xin-She Yang and Yu-Xin Zhao

1 Introduction

Navigation and routing problems are two classes of challenging problems, but we have to solve such problems in practice because they are relevant to many real-world applications. Navigation and migration in animals are both inspiring and interesting. Birds, insects and mammals use various cues for navigation and foraging [1, 2]. Navigation has been an important part of human activities since the dawn of civilization.

Navigation has become even more important, and closely linked to routing in daily life. For example, we often use navigation apps for driving and path planning. Airlines, postal services and logistics all have to solve complicated routing problems with dynamic information, subject to uncertainties such as weather conditions, traffic conditions and fluctuated demands.

Many successful characteristics in animals and biological systems have inspired researchers to design algorithms, and such nature-inspired algorithms can be effective to solve optimization problems, navigation and routing problems. Most of such algorithms are nature-inspired, population-based algorithms, often based on swarm intelligence [3, 4].

This chapter provides an overview of navigation, routing and optimization, with a focus on the introduction to nature-inspired algorithms.

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

Y.-X. Zhao
College of Automation, Harbin Engineering University, Harbin, China

X.-S. Yang and Y.-X. Zhao (eds.), *Nature-Inspired Computation in Navigation and Routing Problems*, Springer Tracts in Nature-Inspired Computing,
https://doi.org/10.1007/978-981-15-1842-3_1

2 Navigation in Animals

The literature on animal navigation and migration is vast, which can span a few subjects such as biology and neuroscience. Here, we only briefly outline some basic features in animal navigation.

Animals have ingenious ways of finding ways, and they use different kinds of cues for navigation [1, 5, 6]. A well-known example is that bats and dolphins use echolocation. Bees can use sunlight, light polarization and even the Earth's magnetic field for navigation and foraging [7], while homing pigeons can use not only light and magnetic field, but also visual landmarks, olfaction and even gravity anomalies [8, 9]. In addition, African dung beetles can even use star light and the Milky Way for navigation [10]. Obviously, to understand how they actually carry out navigation by different cues requires more advanced cognitive neurological studies [2].

Loosely speaking, animal navigation and migration can use various cues and information. For example, as early as 1873, Charles Darwin proposed the concept of dead reckoning or path integration [11], which implies that certain animals can do a running computation of their current locations, based on the speeds and directions of their past moves in their trajectories. It is believed that humans, ants and bats can navigate by path integration [12].

In addition, the Earth's magnetic field can be used as a magnetic compass for path integration [9, 13]. Furthermore, some animals can sense weather conditions such as rains and winds, jet screams, and ocean currents, and use them for navigation and migration [14].

The main navigation characteristics and cues, together with examples, are summarized in Table 1.

The study of animal navigation and migration is a vast subject with a rich set of literature. For detailed reviews on these topics, readers can refer to more specialized literature [1, 2, 6]. Now, we focus on the travelling salesman problems, routing and nature-inspired computation.

Table 1 Examples of animal navigation with various cues

Cues/information	Examples
Sunlight (sun compass)	Almost all animals (mammals, birds, insects . . .)
Moon and stars	Most animals, birds such as warblers, dung beetles
Polarized light	Bees, pigeons, . . .
Landmarks	Humans, pigeons, whales
Magnetic field	Bees, bats, mole rats, sea turtles, some bacteria
Scent/Olfaction	Ants, pigeons, salmons
Echolocation	Bats, dolphins
Gravity variations	Pigeons
Path integration (dead reckoning)	Humans, ants, birds, rodents, . . .

3 Navigation, Routing and Optimization

Routing and navigation are closely linked to optimization. Loosely speaking, navigation in nature is mainly to find the shortest path from the current position after foraging to the home position such as nests. Navigation and routing in modern settings can include a vast range of applications such as shipping across seas, road navigation via global position systems, routing of vehicles such as trucks and lorries for goods delivery and pickup, airline routing and many others. The main aim is to optimize a certain cost function, such as the minimization of the overall distance, transport costs and the travel time. Mathematically speaking, they can be formulated as optimization problems, and thus optimization algorithms and techniques can be used to solve such problems.

3.1 Optimization

An optimization problem consists of a set of design parameters or decision variables x_i $(i = 1, 2, \ldots, D)$, which form a decision or design space spanned by vectors in the form of $\boldsymbol{x} = (x_1, x_2, \ldots, x_D) \in \mathbb{R}^D$. The aim is to either minimize or maximize the cost or objective function $f(\boldsymbol{x})$, subject to some equality and inequality constraints. The optimal solution is a specific set of solutions (often a single solution) that can globally minimize or maximize $f(\boldsymbol{x})$, and such solutions should satisfy all the constraints.

In general, an optimization problem can be formulated as

$$\text{minimize} \quad f(\boldsymbol{x}), \tag{1}$$

subject to

$$h_i(\boldsymbol{x}) = 0, \quad (i = 1, 2, \ldots, I), \tag{2}$$

$$g_j(\boldsymbol{x}) \leq 0, \quad (j = 1, 2, \ldots, J), \tag{3}$$

where h_i and g_j are the equality constraints and inequality constraints, respectively. As maximization problems can be converted into minimization by simply multiplying the objective by -1, we can formulate all optimization problems as minimization.

If problem functions $f(\boldsymbol{x})$, $h_i(\boldsymbol{x})$ and $g_j(\boldsymbol{x})$ are all nonlinear, the problem becomes a nonlinear, potentially highly multimodal, optimization problem and it is usually very challenging to solve. If the search domain is relatively regular, and problems functions are sufficiently smooth, traditional methods can be used, including quadratic programming, interior-points, trust-region method and other [15]. In case of all functions are linear, it becomes a linear program. If the domain is convex and the objective is also convex, it becomes a convex optimization problem, and

thus can be solved efficiently. A recent trend is that nature-inspired algorithms for optimization are also widely used [16].

If decision variables x_i can take only discrete values, the optimization problem becomes a discrete optimization problem. In case of x_i can only take integer values, such optimization is called integer programming. If some variables are discrete, while others are continuous, the optimization problem becomes a mixed integer program. Both mixed integer programming and integer programming are discrete or combinatorial optimization problems that are challenging to solve. Existing methods can only deal with small-scale problems. For large-scale combinatorial problems, there are no efficient methods, and thus approximation methods, heuristic and metaheuristic algorithms become useful alternatives [4].

3.2 Travelling Salesman Problem

The travelling salesman problem (TSP) is a well-known combinatorial optimization problem where there are n cities, and the aim is to visit each city exactly once with the shortest possible overall distance [17, 18]. Extensive literature exists for TSP as it relates to many important problems. Different formulations can focus on different aspects, but we will use a binary integer linear program, first formulated by G. Dantzig et al. in 1954 [18].

Let us label the n cities as $i = 1, 2, \ldots, n$. Each city can be considered as a node and all the cities and their links (routes as edges) form a graph. If a path from city i to city j is taken, we can use a binary decision variable to record this as $x_{ij} = 1$, while $x_{ij} = 0$ means there is no direct connection between i and j for this particular route.

Let us denote all the cities as a set C and all the connections as the set P. The distance between any two city is d_{ij}, which should be symmetric $d_{ji} = d_{ij}$. Thus, the minimization of the total travel distance is to

$$\text{minimize} \sum_{i,j \in P, i \neq j} d_{ij} x_{ij}, \quad x_{ij} \in \{0, 1\}. \tag{4}$$

The requirement of visiting each city exactly once means that

$$\sum_{j=1}^{n} x_{ij} = 1, \quad (i \in C, i \neq j), \quad \sum_{i=1}^{n} x_{ij} = 1, \quad (j \in C, j \neq i), \tag{5}$$

which is equivalent to the case that there is exactly one edge entering a city and exactly one leaving the same city.

For the standard TSP, unconnected subtours are not allowed. This means that

$$\sum_{i,j \in S} x_{ij} \leq |S| - 1, \tag{6}$$

where $2 \leq |S| \leq n - 2$, and $|S|$ is the cardinality of any subset $S \in C$.

This is a binary integer linear programming problem, but it is non-deterministic polynomial-time (NP) hard [17]. For NP-hard problems, there are no efficient algorithms in the current literature. However, there are some exact methods (for small-scale problems) and approximation methods for slightly larger-scale problems [17]. For example, the cutting plane method and linear programming relaxation methods can be used to obtain good solutions. In addition, heuristic and metaheuristic approaches are also now widely used.

3.3 Routing Problems

Routing problems concern many applications, including airline path planning, routing of delivery trucks, postal services and others. Among the routing problems, vehicle routing is an important class with many variants [19–22]. Vehicle routing problems (VRP) can be considered as an extension to the TSP with multiple subtours that all come to the same depot and each vehicle has a limited capacity. In practice, there are many different vehicle types with different capacities, but for simplicity, here, we assume that all the vehicles belong to the same type with the same fixed capacity C. It is worth pointing out that the TSP can be considered as a special case of a VRP with an unlimited capacity $C = \infty$. That is, all goods can be loaded on a single vehicle and thus the VRP becomes a standard TSP.

There are many different ways for formulating a vehicle routing problem, and different formulations can take very different forms [19–21, 23]. Here, we use a relative general vehicle flow formulation, based on Munari et al. [20] where there are n customers each with a demand quantity $Q_i > 0$ ($i = 1, 2, \ldots, n$). The goods will be delivered by the same type of vehicle, each with the maximum capacity C. As there is a capacity constraint, the VRP is also called capacitated vehicle routing problem (CVRP) in the literature.

The transport costs t_{ij} are the cost (or time) sum of the route taken from node i (customer i) to node j (customer j). The decision variable x_{ij} is a binary variable, in the same way as in the TSP. Thus, $x_{ij} = 1$ means the route is from i to j, otherwise $x_{ij} = 0$.

For simplicity, we use the same convention given in [20] where the depot is denoted as $i = 0$ and $i = n + 1$, which means that a route starts with the depot $i = 0$ and ends with the same depot $i = n + 1$. This notation makes the formulation much easier to understand. However, to be consistent, it requires that $Q_0 = 0$ and $Q_{n+1} = 0$; that is, there is no demand at the depot.

Therefore, the objective function is

$$\text{Minimize} \quad \sum_{i=0}^{n+1} \sum_{j=0}^{n+1} t_{ij} x_{ij}. \tag{7}$$

To guarantee that each customer is only visited once, it is required that

$$\sum_{j \neq i, j=1}^{n+1} x_{ij} = 1, \quad (i = 1, 2, \ldots, n). \tag{8}$$

Similarly, we have to ensure that the route goes to the same customer i must come out exactly once, which requires

$$\sum_{i=0, i \neq k}^{n} x_{ik} - \sum_{j=1, j \neq k}^{n+1} x_{kj} = 0, \quad (k = 1, 2, \ldots, n). \tag{9}$$

Obviously, the maximum number of routes allowed is limited by the number N_v of available vehicles. That is

$$\sum_{k=1}^{n} x_{0k} \leq N_v. \tag{10}$$

In order to consider the demands and capacity consistently, we now use a cumulated demand ξ_j on the route leading to the current visit of customer j [20], and this quantity is also linked to the load left in the vehicle up to the current visit. This means that

$$Q_i \leq \xi \leq C, \quad \text{(capacity should not be exceeded)}, \tag{11}$$

and

$$(\xi_j - \xi_i) + C(1 - x_{ij}) \geq Q_j x_{ij}, \tag{12}$$

which gives $(\xi_j - \xi_i) \geq Q_j$ if $x_{ij} = 1$. This means that there is enough load left to satisfy demand Q_j. In case this route is not taken ($x_{ij} = 0$), we have $(\xi_i - \xi_j) \leq C$ that satisfies the capacity constraint. In addition, this equation can also avoid any subtours that potentially go through the depot multiple times [21].

In the real-world applications, different factors and constraints can come into play. Thus, there is a spectrum of different VRP variants [21, 22]. We do not intend provide a detailed review here. Instead, we briefly outline some VRP variants:

- VRP with time windows (VRPTW): If the delivery of goods to a customer i is constrained by a time window $[S_i, E_i]$, not before a starting time S_i and not after an end time E_i, then the VRP becomes a VRP with time windows (*i.e.*, VRPTW).
- VRP with pickup and delivery (VRPPD): In addition to the delivery tasks, if pickup tasks are added, the VRP becomes a VRP with pickup and delivery.

- VRP with last in first out (LIFO): In certain loading conditions and packaging requirements, the goods loaded in last may be unloaded first. This will complicate the delivery conditions with added constraints.
- Rich VRP (R-VRP): If different types of vehicles with multi-attributes are involved, the VRP becomes a rich VRP and its mathematical formulation may become much more complicated [22, 24, 25].
- Open VRP: In some applications, the vehicles may not need to return to the main depot after delivery, thus the routes become open, not closed in this case. Consequently, the mathematical formulation will be different.
- Dynamic VRP: As the demands can be dynamic and traffic conditions are time-varying, the VRP will become dynamic if any dynamically varying constraints are added. Such dynamic VRPs may become very challenging to solve.

There are other variants such as heterogenous VRPs, variable cost VRPs, forbidden path VRPs and others [24, 25]. As these VRPs are typically NP-hard, the solution methods tend to be approximate and heuristics. In recent years, nature-inspired algorithms have shown to be effective. For example, Osaba et al. considered a newspaper delivery system with recycling policy [26], and they used a discrete firefly algorithm (DFA) to solve it with good results. In the rest of this chapter, we will focus on the brief review of nature-inspired optimization algorithms [4].

4 Nature-Inspired Algorithms for Optimization

The literature of nature-inspired algorithms has expanded significantly in recent years [4], thus it is impractical to review all relevant studies. Instead, we only outline a few algorithms so as to highlight the main ideas and search mechanisms.

4.1 Deterministic or Stochastic

Before we introduce any nature-inspired algorithms, let us consider a key question about deterministicity and stochasticity in algorithms.

Loosely speaking, traditional algorithms are mostly deterministic without any randomness, which means that such algorithms have high exploitation capability, but low exploration capability. In contrast, nature-inspired algorithms always use some randomness and thus become non-deterministic. However, their exploration capability is enhanced, but exploitation capability is reduced [27]. However, there should be a fine balance, though such balance can be difficult to find and it may be problem-specific [28].

Therefore, when we use nature-inspired algorithms, we should bear in mind that randomness is an intrinsic part of such algorithms, and consequently, multiple runs and some parameter tuning are needed to ensure the consistency and quality of the final solutions.

4.2 Genetic Algorithms

Among nature-inspired algorithms, the genetic algorithm (GA) was probably the most well-known example. GA was developed by John Holland [29], which is an evolutionary algorithm.

Solution vectors to an optimization problem are encoded as binary strings of 0s and 1s, called chromosomes. Three genetic operators are used to modify the strings, and they are crossover, mutation and selection [4, 30]. Two new (child) solutions are produced via crossover where the solution strings from two parent solutions are mixed and exchanging corresponding segments. Mutation is also used to generate a new solution by mutating one bit, or multiple bits at multiple locations, of an existing solution as a binary chromosome.

The quality of a solution is determined by its fitness, which is a normalized value, proportion to the objective values for maximization problems and inverse proportion to these values for minimization problems. In addition, a selection mechanism is used so that the fittest individual or solution should be passed onto the next generation.

Though the details may depend on the actual implementations, crossover tends to occur more often, with a higher probability of typically 0.6 to 0.95, while mutation tends to be less frequent, with a much lower probability from 0.001 to 0.05.

There is no explicit mathematical equation in genetic algorithms, it is a detailed algorithmic procedure and thus many different variants exist in the literature [30].

4.3 Ant Colony Optimization

The ant colony optimization (ACO) was the first swarm intelligence-based algorithm, and was developed by Marco Dorigo in 1992 [31]. In essence, ACO intends to simulate the behaviour of social ants in a colony, and pheromone is used for simulating local interactions and communications among ants. Pheromone is deposited by each ant, which also gradually evaporates with time. The exact form of evaporation model may vary, depending on the variant and form of ACO used in implementations. Typically, incremental deposition and exponential decay of pheromone are implemented in most studies.

For the TSP and routing problems, a solution is encoded as a path or route, and each ant will explore the possible routing networks independently. A route is marked by pheromone deposited by ants going through the route. The quality of a route (a solution) is related to the pheromone concentration on the path. As pheromone will evaporate with time, the pheromone concentration will vary, depending on the detailed history of ant movements. At a junction with multiple possible routes, a route with the highest concentration of pheromone is preferred. However, a particular route is chosen with a probability, and this probability is determined by the normalized concentration of the route, the desirability of the route (such as the overall path distance) and relative fitness of this route, comparing with all others.

Similar to the genetic algorithm, there is no algorithmic equation for ACO in terms of the positions and moves of the ants. However, their movements are described as a mixed of procedure, in combination with the pheromone deposition and evaporation. Thus, there are many variants [31].

4.4 Particle Swarm Optimization

As a swarm intelligence-based algorithm, particle swarm optimization (PSO) was developed by Kennedy and Eberhart in 1995 [3], which simulates the swarming characteristics of birds and fish. In PSO, there is a population of n particles.

If the position and velocity of a particle are denoted by $\boldsymbol{x}_i$ and $\boldsymbol{v}_i$, respectively, the position vector can be considered as a solution vector to an optimization problem. For each particle, its position and velocity at any iteration t (or pseudo-time t) will be updated iteratively as follows:

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

and

$$\boldsymbol{v}_i^{t+1} = \boldsymbol{v}_i^t + \alpha \boldsymbol{\epsilon}_1 [\boldsymbol{g}^* - \boldsymbol{x}_i^t] + \beta \boldsymbol{\epsilon}_2 [\boldsymbol{x}_i^* - \boldsymbol{x}_i^t], \tag{14}$$

where $\boldsymbol{\epsilon}_1$ and $\boldsymbol{\epsilon}_2$ are two uniformly distributed random numbers in [0,1]. The time increment $\Delta t = 1$ can be used because all iterative systems can be considered as a discrete system with unit time increments. In addition, the best solution $\boldsymbol{g}^*$ is the fittest solution among all the particles in the population. Each particle has a short memory to record its individual best solution $\boldsymbol{x}_i^*$ during the iteration history. Here, parameters α and β are often referred to as learning parameters, and their values are usually in the range of [0,2]. It is worth pointing out that the notation t is for an iteration counter (or pseudo-time t). It should not be confused with an exponent.

There are many variants of PSO, and some variants introduced an inertia weight parameter (adding a mass to each particle), leading to a more stable PSO variant.

4.5 Firefly Algorithm

Firefly algorithm (FA) is also a swarm intelligence-based algorithm, developed by Xin-She Yang in 2008, inspired by the swarming behaviour and light-flashing characteristics of tropical fireflies [32]. A solution vector to an optimization problem is represented as the position of a firefly. With a population of n fireflies, a swarm of solutions will evolve according to the governing equation in FA. In fact, FA can be considered as a nonlinear system and its nonlinearity comes from the variation of light intensity in terms of the inverse-square law and the exponential decay of light due to absorption.

The position vector $\boldsymbol{x}_i$ of firefly i at iteration t is updated by

$$\boldsymbol{x}_i^{t+1} = \boldsymbol{x}_i^t + \beta_0 e^{-\gamma r_{ij}^2}(\boldsymbol{x}_j^t - \boldsymbol{x}_i^t) + \alpha\, \boldsymbol{\epsilon}_i^t, \tag{15}$$

where $\beta_0 > 0$ is the attractiveness of a firefly at the zero distance (i.e., $r_{ij} = 0$). In addition, α is a scaling factor controlling the step sizes or strength of random perturbations. In the attraction term, the factor γ in the exponent is a parameter, which should be linked to the scales of the problem under consideration because this parameter essentially controls the visibility of the fireflies.

A useful property of the FA as a nonlinear system is that the population initially as a single swarm can subdivide into multiple subgroups or sub-swarms, which arises from the fact that long-distance attraction is weaker than short-distance attraction. As each sub-swarm can swarm potentially around a local mode (including the mode with the global optimality), FA can be naturally suitable for multimodal optimization problems [33].

4.6 Cuckoo Search

The cuckoo search (CS) algorithm was developed by Xin-She Yang and Suash Deb in 2009 [34]. CS was based on the brooding parasitism of some cuckoo species who lay their eggs in the nests of host birds such as warblers. In the real cuckoo–host species system, the eggs laid by cuckoos are sufficiently similar to the eggs of the host species. Cuckoos' eggs can be discovered and abandoned with a probability p_a.

There are n eggs in the CS system. If we encode the position of an egg as a solution vector $\boldsymbol{x}_i$ to an optimization problem, the similarity of two eggs (solutions $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$) can be roughly measured by their difference $(\boldsymbol{x}_j - \boldsymbol{x}_i)$. Thus, the position at iteration t can be updated in the following way [35]:

$$\boldsymbol{x}_i^{t+1} = \boldsymbol{x}_i^t + \alpha s \otimes H(p_a - \epsilon) \otimes (\boldsymbol{x}_j^t - \boldsymbol{x}_k^t). \tag{16}$$

Here, the discovery of a cuckoo's egg is simulated by a simple Heaviside function $H(u)$ by drawing a random number ϵ from a uniform distribution, and s is the step size. Here, the notation $\otimes$ denotes an element-wise multiplication.

A distinct feature of the host birds is that they may abandon their nests and fly away if they suspect their eggs were replaced or contaminated. This can be mimicked as a step size s by Lévy flights [36, 37]:

$$\boldsymbol{x}_i^{t+1} = \boldsymbol{x}_i^t + \alpha L(s, \lambda), \tag{17}$$

where the Lévy flights are realized by drawing random numbers from

$$L(s, \lambda) \sim \frac{\lambda \Gamma(\lambda) \sin(\pi\lambda/2)}{\pi} \frac{1}{s^{1+\lambda}}, \quad (s \gg 0). \tag{18}$$

In addition, $\alpha > 0$ is the step size scaling factor.

A main advantage of using Lévy flights is that a fraction of steps or moves generated by Lévy flights are much larger than those used in Gaussian [36], which can enhance the exploration ability of the CS since search steps in CS are heavy-tailed and potentially more effective [33, 35, 38].

4.7 Bat Algorithm

The bat algorithm (BA) is also swarm intelligence-based algorithm, developed by Xin-She Yang in 2010. BA intends to mimic some characteristics of echolocation of microbats and their frequency-tuning ability [39, 40]. Both the pulse emission (with a rate r) and loudness A are used to partially control exploration and exploitation.

In BA, the position of a bat is used to represent a solution vector, and a set of n solutions form the population. The update of the position vectors is carried out by

$$f_i = f_{\min} + (f_{\max} - f_{\min})\beta, \tag{19}$$

$$\boldsymbol{v}_i^t = \boldsymbol{v}_i^{t-1} + (\boldsymbol{x}_i^{t-1} - \boldsymbol{x}_*) f_i, \tag{20}$$

$$\boldsymbol{x}_i^t = \boldsymbol{x}_i^{t-1} + \boldsymbol{v}_i^t, \tag{21}$$

where the frequency varies randomly from $f_{\min}$ to $f_{\max}$ by drawing a random number $\beta \in [0, 1]$.

In addition, the loudness $A(t)$ of each bat should vary from a high value to a lower value when zooming towards a promising region, while the emission rate r should vary at the same time from a lower value to a higher value. This means that

$$A_i^{t+1} = \alpha A_i^t, \quad r_i^{t+1} = r_i^0(1 - e^{-\gamma t}), \tag{22}$$

where the two parameters $0 < \alpha < 1$ and $\gamma > 0$ control the exact form of actual variations. In general, both numerical experiments and empirical observations suggested that BA could have a faster convergence rate in comparison with PSO. Recently, the standard BA has been extended to multiobjective optimization and hybrid versions [40].

4.8 Flower Pollination Algorithm

The flower pollination algorithm (FPA) is a multi-agent, population-based algorithm, based on the pollination characteristics of flowering plants [4, 41], though FPA is not a swarm intelligence-based algorithm. There are two main search mechanisms in the FPA, which mimic the main features of both biotic and abiotic pollination. In

addition, FPA also includes the flower constancy due to the co-evolution of some flower species with their corresponding pollinators such as insects.

If the position of a pollen particle is represented as a solution vector $\boldsymbol{x}_i$, then the moves of pollen can be simulated by the global search

$$\boldsymbol{x}_i^{t+1} = \boldsymbol{x}_i^t + \gamma L(\lambda)(\boldsymbol{g}_* - \boldsymbol{x}_i^t), \tag{23}$$

where γ is a scaling parameter and $\boldsymbol{g}_*$ is the best solution found so far at iteration t. Here, the random vector $L(\lambda)$ should be drawn from a Lévy distribution, and this distribution is typically characterized by exponent λ [36].

The current solution $\boldsymbol{x}_i^t$ can also be modified by local search or moves

$$\boldsymbol{x}_i^{t+1} = \boldsymbol{x}_i^t + U(\boldsymbol{x}_j^t - \boldsymbol{x}_k^t), \tag{24}$$

where U is drawn from a uniform distribution in [0, 1]. This search mechanism mimics local pollination and flower constancy. Metaphorically speaking, $\boldsymbol{x}_j^t$ and $\boldsymbol{x}_k^t$ can be considered as solutions representing pollen from different flower patches.

4.9 Other Algorithms

The above algorithms are mostly swarm intelligence-based algorithms, which have some sort of the main characteristics of swarm intelligence [42, 43]. Obviously, there are many other nature-inspired metaheuristic algorithms, including the artificial bee colony, differential evolution [44], gravitational search, artificial immune system and others. Interested readers can refer to more specialized literature such as journal articles and books [4, 33].

In the rest of this chapter, we will briefly discuss the main characteristics of nature-inspired algorithms and then focus on the solution representations and discretization of algorithms for continuous optimization.

5 Algorithmic Characteristics

Extensive studies have shown that the algorithms we have discussed and other algorithms can work well in practice. However, their mathematical analysis requires a more rigorous, formal approach, and there are still many open questions [28]. Obviously, there are different ways of analysing such algorithms [4, 27, 28], we will discuss the main characteristics of these algorithms.

5.1 Characteristics

Now, let us first look qualitatively at these nature-inspired algorithms by focusing on their search mechanisms, basic steps and algorithm dynamics.

Table 2 Characteristics and their role in nature-inspired algorithms

Algorithmic components	Role (or Properties)
Population/multiple agents	Diversity, sampling in search space
Randomization/modifications	Perturbation of states, escape local optima
Elitism, selection	Driving force for evolution, leading to convergence
Iterative update equations	Evolution of solutions as a time-discrete system

- All these algorithms use a population with multiple agents (e.g., particles, ants, bats, cuckoos, fireflies, bees, etc.), each agent represents a solution vector. The evolution of the population is often achieved by some operators (e.g., mutation, crossover), often in terms of some algorithmic formulas or equations.
- All algorithms have certain forms of both local and global search. Modifications and perturbations of existing solutions in the population are achieved by randomization with selection, biased towards solutions with higher fitness. Darwinian principle of the 'survival of the fittest' is used for selecting the better or best solution among the population.
- The evolution of solutions is iterative, leading to the evolution of solutions with varying properties. When all solutions become similar, the system may lead to some self-organized states or converged states. A converged population consists of some organized structure, though the diversity has been reduced.

Such characteristics and behaviour can also be analysed from the self-organization point of view [45, 46]. In fact, algorithms and self-organization systems have many similarities. These basic components and their role or properties can be summarized in Table 2.

It is worth pointing out that there is no free lunch in algorithms [47], and there is no single best algorithm that can solve all the problems. Thus, one of the main tasks of research is to identify what types of problems an algorithm can solve and what algorithm(s) should be used for a given type of problems.

5.2 Discretization and Solution Representations

Most nature-inspired algorithms were originally designed for solving optimization problems with continuous variables. For discrete and combinatorial optimization problems, the variables are discrete. Thus, we have to convert the standard algorithms such as the firefly algorithm and cuckoo search algorithm into their discrete versions.

One way to discretize a continuous variable x is to use the sigmoidal or logistic function

$$S(x) = \frac{1}{1 + e^{-x}}, \tag{25}$$

which is an S-shaped function. It essentially converts a continuous variable x into a binary variable S when $|x|$ is large.

However, this is not easy to implement in practice. So a random number $r \in [0, 1]$ is usually generated and used as a conditional switch. That is, if $S(x) > r$, $u = 1$, otherwise $u = 0$, which gives a binary variable $u \in \{0, 1\}$. Once we have a binary variable $u \in \{0, 1\}$, we can convert it to binary variables with other discrete values. For example, we can use $y = 2u - 1 \in \{-1, +1\}$.

It is worth pointing out that a useful property of $S(x)$ is that its derivative can be computed by multiplication

$$\frac{dS}{dt} = -\frac{1}{(1+e^{-x})^2}(-e^{-x}) = \frac{1}{1+e^{-x}} \cdot \frac{e^{-x}}{1+e^{-x}}$$

$$= \frac{1}{1+e^{-x}} \cdot \frac{[(1+e^{-x})-1]}{1+e^{-x}} = \frac{1}{1+e^{-x}} \cdot [1 - \frac{1}{1+e^{-x}}] = S(1-S). \quad (26)$$

Another way for discretization is to use round-up operations to get integer values. For example, we can use

$$y = \lfloor x \rfloor, \quad (27)$$

to convert x to integer y.

Alternatively, we can convert a continuous variable into m discrete integers by using a mod function

$$y = \lfloor x + k \rfloor \bmod m, \quad (28)$$

where k and $m > 0$ are integers.

Sometimes, a so-called random key can be used to generate a set of discrete values for nodal numbers (as in the travelling salesman problem) and customer numbers in routing problems. For example, a random key

$$x = [0.91,\ 1.1,\ 0.14,\ 0.09,\ 0.77, -0.23, 0.69], \quad (29)$$

can be converted to

$$J = [6,\ 7,\ 3,\ 2,\ 5,\ 1,\ 4]. \quad (30)$$

This is done by ranking the real number vector x first, and then transforming them into labels of ranks. In some applications, such continuous numbers are drawn from a uniformly distributed in [0,1]. For example, we have

$$\begin{bmatrix} \text{Real numbers} & 0.65 & 0.25 & 0.37 & 0.04 & 0.89 \\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ \text{Random keys} & 4 & 2 & 3 & 1 & 5 \end{bmatrix}. \quad (31)$$

Such random-keys-based approaches have been applied in many applications such as the travelling salesman problem (TSP) and vehicle routing problems [26, 33].

For TSPs and routing problems, there are many different ways of generating new routes and adjacent routes, including 2-opt, 3-opt and k-opt moves. Different metric measures such as Hamming distances can also be defined. We will not introduce them here, and interested readers can refer to more advanced literature [17, 21].

6 Conclusions

In this chapter, we have briefly introduced some interesting characteristics of animal navigation and migration. We have also discussed the relationship among navigation, routing problems and optimization. Then, we have introduced some of the widely used nature-inspired algorithms for optimization, followed by the representations of solutions for discrete optimization.

The applications of these algorithms are diverse, and more applications will be introduced in the other chapters of this book. It is hoped this brief overview can inspire more research, concerning navigation, routing problems and nature-inspired computation.

References

1. Alcock J (2005) Animal Behavior: An Evolutionary Approach, 8th Edition, Sinauer Associates Publishing, Sunderland. Mass, USA
2. Redish AD (1999) Beyond the cognitive map. MIT Press, Cambridge
3. Kennedy J, Eberhart RC ((1995)) Particle swarm optimization. In: Proceedings of IEEE international conference on neural networks, Piscataway, NJ, pp 1942–1948
4. Yang XS (2014) Nature-inspired optimization algorithms. Elsevier Insight, London
5. Dingle H, Drake VA (2007) What is migration? Bioscience 57(2):113–121
6. Gauthreaux SA (1980) Animal migration, orientation, and navigation. Academic Press, Edinburgh
7. von Frisch K (1967) The dance language and orientation of bees, Harvard University Press, Cambridge, Mass. (Translation from Tanzsprache und Orientierung der Bienen)
8. Blaser N, Guskov SI, Entin VA, Wolfer DP, Kanevskyi VA, Lipp H (2014) Gravity anomalies without geomagnetic disturbances interfere with pigeon homing—a GPS tracking study. J Exp Biol 217(22):4057–4067
9. Walcott C (1996) Piegon homing: observations, experiments and confusions. J Exp Biol 199(1):21–27
10. Dacke M, Baird E, Byrne M, Scholtz CH, Warrant EJ (2013) Dung beetles use the Milky Way for orientation. Curr Biol 23(4):298–300
11. Darwin C (1873) Origin of certain instincts. Nature 7(179):417–418
12. Whishaw IQ, Hines DJ, Wallace DG (2001) Dead reckoning (path integration) requires hippocampal formation: evidence from spontaneous exploration and spatial learning tasks in light (allotehetic) and dark (idiothetic) tests. Behav Brain Res 127(1–2):49–69
13. Kimchi T, Etienne AS, Terkel J (2004) A subterranean mammal uses the magnetic compass for path integration. PNAS 101(4):1105–1109
14. Lohmann KJ, Lohmann CMF, Endres CS (2008) The sensory ecology of ocean navigation. J Exp Biol 211(11):1719–1728

15. Boyd S, Vandenberghe L (2004) Convex optimization. Cambridge Univeristy Press, Cambridge
16. Yang XS (2010) Engineering optimization: an introduction with metaheuristic applications. Wiley, Hoboken
17. Applegate DL, Bixby RM, Chvátal V, Cook WJ (2007) Traveling salesman problem: a computational study. Princeton University Press, Princeton
18. Dantzig GB, Fulkerson R, Johnson SM (1954) Solution of a large-scale traveling salesman problem. Oper Res 2(4):393–410
19. Christofides N, Mingozzi A, Toth P (1979) The vehicle routing problem. Wiley, Chichester, UK
20. Munari P, Dollevoet T, Spliet R (2017) A generalized formulation for vehicle routing problems. arXiv:1606.01935v2. Accessed on 22 Aug 2019
21. Toth P, Vigo D (2014) Vehicle routing: problems, methods and applications, MOS-SIAM series on optimization, 2nd edn. Society for Industrial and Applied Mathematics
22. Wen M (2010) Rich vehicle routing problems and applications, Deparment of Management Engineering, Technical University of Denmark, Ph.D. thesis
23. Laporte G, Toth P, Vigo D (2013) Vehicle routing: historical perspective and recent contributions. EURO J Transp Logist 2(1–2):1–4
24. Lahyani R, Klemakhem M, Semet F (2015) Rich vehicle routing prblems: From a taxonomy to a definition. Eur J Oper Res 241(1):1–14
25. Vidal T, Crainic TG, Gendreau M, Prins C (2013) Heuristics for multi-attribute vehicle routing problems: A survey and synthesis. Eur J Oper Res 231(1):1–21
26. Osaba E, Yang XS, Diaz F, Onieva E, Masegosa AD, Perallos A (2017) A discrete firefly algorithm to solve a rich vehicle routing problem modelling a newspaper distribution system with recycling policy. Soft Comput 21(11):5295–5308
27. Blum C, Roli A (2003) Metaheuristics in combinatorial optimization: overview and conceptural comparision. ACM Comput Surv 35(2):268–308
28. Yang XS, He XS (2019) Mathematical foundations of nature-inspired algorithms. Springer, Cham, Switzerland
29. Holland J (1975) Adaptation in natural and arficial systems. University of Michigan Press, Ann Arbor, USA
30. Goldberg DE (1989) Genetic algorithms in search, optimisation and machine learning. Reading, Addison Wesley, Mass, Reading, MA
31. Bonabeau E, Dorigo M, Theraulaz G (1999) Swarm intelligence: from natural to artificial systems. Oxford University Press, Oxford
32. Yang XS (2010) Firefly algorithm, stochastic test functions and design optimisation. Int J Bio-Inspired Comput 2(2):78–84
33. Yang XS (2014) Cuckoo search and firefly algorithm: theory and applications, studies in computational intelligence, vol 516. Springer, New york
34. Yang XS, Deb S (2009) Cuckoo search via Lévy flights. In: Proceedings of world congress on nature and biologically inspired computing. (NaBic 2009), Coimbatore, India, IEEE Publications, USA, pp 210–214
35. Yang XS, Deb S (2014) Cuckoo search: recent advances and applications. Neural Comput Appl 24(1):169–174
36. Pavlyukevich I (2007) Lévy flights, non-local search and simulated annealing. J Comput Phys 226(2):1830–1844
37. Reynolds AM, Rhodes CJ (2009) The Lévy fligth paradigm: random search patterns and mechanisms. Ecology 90(4):877–887
38. Yang XS, Deb S (2013) Multiobjective cuckoo search for design optimization. Comput Oper Res 40(6):1616–1624
39. Yang XS (2010) A new metaheuristic bat-inspired algorithm, In: Nature-inspired cooperative strategies for optimization. (NICSO 2010), Springer, SCI 284, pp 65–74
40. Yang XS (2011) Bat algorithm for multi-objective optimisation. Int J Bio-Inspired Comput 3(5):267–274

41. Yang XS, Karamanoglu M, He XS (2014) Flower pollination algorithm: a novel approach for multiobjective optimization. Eng Optim 46(9):1222–1237
42. Fisher L (2009) The perfect swarm: the science of complexity in everyday life. Basic Books, London
43. Surowiecki J (2004) The Wisdom of crowds. Anchor Books
44. Storn R, Price K (1997) Differential evolution: a simple and efficient heuristic for global optimization over continuous spaces. J Glob Optim 11(4):341–59
45. Ashby WA (1962) Princinples of the self-organizing sysem. In: Von Foerster H, Zopf GW (eds) Principles of self-organization: transactions of the university of Illinois symposium. Pergamon Press, London, UK, pp 255–278
46. Keller EF (2009) Organisms, machines, and thunderstorms: a history of self-organization, part two. Complex Emergenece Stable Attractors Hist Stud Nat Sci 39(1):1–31
47. Wolpert DH, Macready WG (1997) No free lunch theorem for optimization. IEEE Trans Evol Comput 1(1):67–82

Chapter 2
Navigation and Navigation Algorithms

Yu-Xin Zhao and Ri-Xu Hao

1 Navigation Introduction

The development of navigation technology stems from the need for identifying direction and position in both the military and production activities of human society. With the advancement of society and the improvement of productivity, the space for human activities is constantly expanding, causing the requirements for navigation to be higher and higher. This is a huge traction force in the development of navigation technology. This chapter mainly introduces the basic concepts of navigation, the development history, and the main navigation algorithms.

1.1 Navigation Origin

Navigation comes with the emergence of political, economic, and military activities and continues to evolve with the development of these fields. With the continuous expansion of human activity and the continuous improvement of the vehicles used, the requirements for navigation are getting higher and higher.

Ancient people mostly lived along rivers. The initial range of human activities was mainly limited to the ancient Yellow River basin, Indus River basin, Mediterranean Sea, and Persian Gulf coast. By utilizing fire and stone axes, in order to meet the needs of fishing and river crossing, human beings created the earliest means of water transportation-canoe. With canoes, people's range of activities had expanded. From then on, they could cross the waters to explore far away. Vehicles were created to meet needs, the original canoes and bamboo rafts for people's exploration and fishing needs, and gradually to larger vessels and warships to meet the needs of explorers for

Y.-X. Zhao (✉) · R.-X. Hao
College of Automation, Harbin Engineering University, Harbin, China
e-mail: zhaoyuxin@hrbeu.edu.cn

X.-S. Yang and Y.-X. Zhao (eds.), *Nature-Inspired Computation in Navigation and Routing Problems*, Springer Tracts in Nature-Inspired Computing,
https://doi.org/10.1007/978-981-15-1842-3_2

longer voyages and wars. Until the mid-eighteenth century, transportation still relied mainly on human, animal, and wind power, and its development was relatively slow. At the beginning of the nineteenth century, the emergence of steam power led to the design of trains and ships, which greatly promoted the development of maritime and rail transport. At the end of the nineteenth century, a large number of vehicles were put into use to further boomed land transport. The rise of air transport in the early twentieth century has greatly accelerated the rhythm of human economic and military activities.

In order to meet the need for human beings to carry out economic, military and scientific research activities underwater, on land, in the air, and in outer space, a variety of vehicles are needed, such as land vehicles, ships, aircraft, rockets, satellites, and spacecraft. To ensure that vehicles and people reach their destinations safely and accurately, we require precise navigation information.

1.2 Navigation Definition

Navigation is a subject that studies the theory, technology, and methodology for determining the position, velocity, azimuth, and attitude of a moving vehicle, recording, planning, and controlling the behavior path of the vehicle. Its development is caused by the intersection of multiple disciplines such as aerospace, communications and electronics, computers, and geodesy. Navigation has gradually formed an engineering discipline with relatively complete theory and technology system. With the development and progress of science and technology, navigation connotation and extension are enriched and developed, which is an emerging discipline with broad development prospects and application requirements.

The basic tasks of navigation include:

1. Guiding the vehicle to enter and navigate along the scheduled route.
2. Guiding the vehicle to safely land or enter the port at night while under various weather conditions.
3. Provide other guidance and information consulting services needed for the vehicle to complete the navigation task accurately and safely.
4. Determine the current position of the vehicle and its navigation parameters.

A system that can complete the above navigation tasks is called a navigation system.

1.2.1 Positioning

Positioning is the process of determining the absolute or relative position of a space target with a given reference by some technical means. The positioning process is used to determine the space position of the vehicle, that is, the direction, pitch angle, and distance parameters of the vehicle, but not the velocity or attitude.

The direction and pitch angle measurements can be achieved by relatively simple techniques, such as theodolites and magnetic compasses. Similar to terrestrial landmarks, the sun, moon, and stars can also be used as reference objects.

Range measurements can be achieved by radio signal, laser, or radar. In passive ranging systems, the user receives the transmitting signal from the radio navigation station. In active ranging systems, the user transmits the signal to the reference object and receives the transmitting signal or the response signal from the reference object.

The parts outside of the user equipment in the position fixing system are called the auxiliary navigation facilities, which may include landmarks and radio navigation stations. After obtaining the measured value of the specific direction, the position of the reference point is also needed to determine the user's position. When the reference point is a radio navigation transmitter, its position can be broadcasted directly or identified by radio frequency. When the reference point is not a special transmitter, the reference point must be identified by manual participation or automatic feature matching technology.

1.2.2 Navigation

So far, the definitions of navigation have varied among different people in the world. There is no universally accepted and strict definition of navigation in academic circles. Some are confined to the old concept of navigation, which has no modern meaning, while others expand the field of navigation to other areas beyond the scope of the field of navigation. The definition of navigation before the age of general navigation cannot contain some of the concepts of modern navigation, but it goes beyond the real meaning of navigation itself to define navigation with all the functions of modern navigation equipment. With the development of science and technology, some devices in navigation systems are likely to accomplish many functions beyond navigation and can even be connected with other categories such as communications, weapons and equipment, and management. However, the definition of navigation cannot contain these concepts indefinitely.

The Concise Oxford Dictionary defines navigation as: "Methods for determining the position and course of a motion vector by any means such as geometry, astronomy, and radio signals." That is, a combination of one or more methods of geometric, astronomical, radio signal, and feature matching, to determine the position, velocity, and attitude of the motion carrier and to perform route planning techniques.

The parameters required to complete the navigation task during the navigation process include the position, velocity, and attitude of the motion carrier, the most important of which is the carrier's position parameter. Navigation consists of two concepts. Firstly, determine the position and the velocity of the motion carrier relative to the reference frame. Secondly, plan and maintain the route from the departure point to the destination. The most efficient method and the required navigation performance are used to guide the navigation process and avoid obstacles and collisions. It also could be called guidance, pilot, or path planning for different types of vehicles [1].

In this book, we define navigation as the path planning technique for different types of carriers in complex environments.

1.2.3 Path Planning

Sequential points or curves connecting the starting and ending positions are called paths, and the strategy of constructing paths is called path planning. According to the grasp degree of the environmental information, path planning can be divided into global path planning based on prior complete information and local path planning based on sensor information. From the perspective that the obstacle information obtained is static or dynamic, global path planning belongs to static planning and local path planning belongs to dynamic planning. Global path planning needs to master all of the environmental information and plans paths according to all of the information on the environmental map. In order to select the optimal path from the current node to a sub-target node, local path planning only needs sensors to collect real-time environmental information, understand environmental map information, and determine the position of the map and its local obstacle distribution.

According to the information characteristics of the research environment, path planning can also be divided into path planning in discrete domains and path planning in continuous domains. The path planning problem in a discrete domain belongs to one-dimensional static optimization problems, which is equivalent to the route optimization problem simplified by environmental information, while the path planning problem in continuous domains is a problem in the continuous multi-dimensional dynamic environment.

Path planning in continuous domains includes three steps: environment modeling, path searching, and path smoothing.

1. Environmental modeling. Environmental modeling is an important part of path planning. Its purpose is to establish an environment model which is convenient for computers to use in path planning. The actual physical space is abstracted into an abstract space that the algorithm can handle, and the mutual mapping is realized.
2. Path search. In the path search stage, the corresponding algorithm is applied to find a path based on the environment model to obtain the optimal value of the predetermined performance function.
3. Smooth path. The path searched by the corresponding algorithm is not a necessarily feasible path for a moving body to walk. Further processing and smoothing are needed to make it a practical and feasible path.

There are many methods of path planning. According to its own advantages and disadvantages, its scope of application is also different. Based on the research of common path planning algorithms in various fields, the time sequence of discovery, and the basic principle of the algorithm, the algorithm can be roughly divided into two categories: traditional algorithms and heuristic algorithms. Among them,

heuristic algorithms include the ecosystem algorithm, swarm intelligence algorithm, evolutionary algorithm, and artificial intelligence algorithm [2].

Ecosystem algorithms include the biogeography-based optimization and invasive weed optimization algorithm. Ecosystem algorithms provide the basic method of modeling, but the graphics method generally has a shortage of search ability. It often needs to combine special search algorithms to solve the problem inefficiently.

Swarm intelligence algorithms include the ant colony algorithm, particle swarm optimization, firefly algorithm, artificial bee colony algorithm, bat algorithm, and cuckoo search algorithm. The intelligence inspiration from nature swarm plays an important role in dealing with the path planning problem under complex dynamic environment information.

Evolution algorithms include genetic algorithms, differential evolution algorithms, harmony search algorithms, and differential evolution algorithms. Evolutionary algorithms come from the inheritance, evolution, and variation of natural organisms. These play a role in path planning under complex dynamic environment information.

Artificial intelligence algorithms include the neural network algorithm and deep learning algorithm. The neural network algorithm has excellent learning ability and strong robustness, but the poor generalization is its fatal disadvantage. Therefore, the combination of neural network algorithm and other algorithms has become a hot research topic in the field of path planning.

2 Development of Navigation

With the increasing complexity of terrestrial, oceanic, and aerospace environments and the increasing performance of motion vehicles, the requirement of the navigation system for motion vehicles in different environments is also increasing. It requires not only higher positioning accuracy, faster dynamic response, and better optimization capabilities of the navigation system, but also autonomous learning and three-dimensional navigational abilities of the navigation system. Therefore, heuristic algorithms such as the intelligent bionics algorithm and neural network algorithm could be more widely used in navigation systems for moving vehicles in complex environments.

In recent years, heuristic algorithms, especially meta-heuristic algorithms with the idea of natural evolution, have aroused an upsurge in the research of path planning algorithms for navigation systems and have received high attention and tracking. Compared with the traditional deterministic algorithms, heuristic algorithms have been widely recognized and applied for their intuitive and effective solutions to the navigation problem of moving vehicles in complex environments.

Although the theoretical basis of heuristic algorithms still needs to be developed, the academic thought comes from the long-term observation and practice of physical, biological, and social phenomena, as well as the deep understanding of these natural laws. It is the crystallization of wisdom that human beings gradually learn from

nature and imitate the operation mechanism of its natural phenomena. Therefore, its scientific nature and development potential are self-evident. Since Alan Turing first put forward the concept of the heuristic search in the process of deciphering German passwords during the Second World War, the research of heuristic algorithms has gone through four important stages: the initial stage, the low-speed stage, the prosperous stage, and the blooming stage.

2.1 Initial Germination Stage

During the initial period of the 1940s and 1950s, the team of scientists represented by Alan Turing innovatively put forward the heuristic search academic idea and applied it to major engineering fields such as the design of an automatic computing engine, which initiated the research work in this field.

In 1943, Warren McCulloch and Walter Pitts proposed an M-P neuron model based on biological neural network, which injected new vitality into the development of evolutionary computing and artificial intelligence.

In 1950, Alan Turing published his article Computer Machines and Intelligence, which opened the pioneer of contemporary artificial intelligence science and provided a good conceptual basis for the study of intelligent algorithms in the twenty-first century.

2.2 Low-Speed Development Stage

During the low-speed development period of the 1960s and 1970s, although researchers paid more and more attention to the research of heuristic algorithms and put forward optimization methods such as greedy algorithms and local search, there was still no way to solve large-scale optimization problems due to the limitations of computing conditions and theoretical development at that time.

In 1975, Professor J. Holland first proposed the classical genetic algorithm. Genetic algorithms are computational models which simulate the natural selection and genetic mechanism of Darwin's biological evolution theory. Genetic algorithms have been widely used in combinatorial optimization, artificial intelligence, and other fields.

2.3 Prosperity and Active Stage

In the 1980s and 1990s, after ten years of low-speed development of heuristic algorithms, the algorithm ushered in a period of prosperity and activity. The rapid development of industry, the development of computational complexity theory, and the

urgent need for industrial applications have greatly promoted the research of new and effective search mechanisms and optimization strategies. Hence, the research of heuristic algorithms is booming.

In 1983, S. Kirkpatrick and others successfully introduced the annealing idea into the field of combinatorial optimization and proposed the simulated annealing algorithm. The simulated annealing (SA) algorithm is a stochastic optimization algorithm based on the Monte Carlo iterative solution strategy. Its advantage is that it can effectively avoid falling into local minima and eventually reach global optimum by giving the search process a time-varying probability jump that eventually tends to zero.

In 1986, Fred Glover proposed a deterministic local minimal jump strategy, the Tabu search algorithm. So far, the Tabu search algorithm has achieved great success in combinatorial optimization, machine learning, and neural networks. In recent years, more research has been performed on the global optimization of functions, and there is a great trend in its development.

In 1986, Rumelhart and McClelland proposed the back propagation (BP) neural network. The BP neural network can classify arbitrary complex patterns, excellent multi-dimensional function mapping ability, and a flexible network structure. It is widely used in function approximation, combinatorial optimization, and data compression.

In 1992, Marco Dorigo first proposed the ant colony algorithm, which was inspired by the behavior of ants in finding their way to food. This algorithm has the characteristics of distributed computing, positive information feedback, and heuristic search and is essentially a heuristic global optimization algorithm among the evolutionary algorithm.

In 1995, Storn and Price proposed the differential evolution algorithm, an optimization algorithm based on modern intelligence theory. The differential evolution algorithm is mainly used to solve real-number optimization problems. It is a group-based adaptive global optimization algorithm. Because of its simple structure, easy implementation, fast convergence, and strong robustness, it is widely used in data mining, artificial neural networks, and other fields.

In 1995, J. Kennedy proposed the particle swarm optimization algorithm. Particle swarm optimization (PSO) is a kind of evolutionary algorithm, which attracts the attention of academia for its advantages of easy implementation, high accuracy, and fast convergence, and it shows its superiority in solving practical problems.

2.4 Blooming Stage

Since the beginning of the twenty-first century, the inspiration from nature continues to give birth to the rapid development of heuristic algorithms. In the ten years at the beginning of the century, scholars not only generalized the traditional optimization algorithm, but also explored various novel heuristic algorithms. It is the traditional

heuristic algorithm and the new heuristic algorithm that coordinate competition and promote each other to set off the research climax in this field again.

In 2001, Geem Z W, a Korean scholar, proposed a novel intelligent optimization algorithm, i.e., the harmony search algorithm. This method has been widely used in the optimization of multi-dimensional and multi-dimensional extremum functions, and it shows better optimization performance than genetic algorithms in solving multi-dimensional function optimization problems.

The random leaping frog algorithm is a new sub-heuristic cooperative search algorithm. It was first proposed by Eusuff and Lansey in 2003 to solve combinatorial optimization problems. It has many advantages, such as being easy to understand, being easy to program, and having a strong ability for direct optimization.

The artificial bee colony algorithm was proposed by Turkish scholar Karaboga in 2005. The algorithm has the characteristics of simple operation, few control parameters, high search accuracy, and strong robustness. It has been successfully applied in many fields such as artificial neural network training, combinatorial optimization, and power system optimization.

Inspired by the theory of biogeography, Simon proposed an intelligent optimization algorithm named biogeography-based optimization in 2008, which has good convergence and stability, based on studying the mathematical model of bio-species migration.

In 2008, Xin-She Yang proposed an advanced heuristic algorithm, the firefly algorithm, based on the simplification and simulation of firefly population behavior. The algorithm has high search accuracy and a fast convergence speed. It is an effective swarm intelligence optimization algorithm to provide a new idea for intelligent optimization.

In 2009, Xin-She Yang proposed the cuckoo search algorithm based on the research of mathematical modeling, engineering optimization, and other fields. As this algorithm is easy to understand, easy to implement, and excellent in the random search path, it has been successfully applied to some difficult problems by researchers.

In 2010, Xin-She Yang proposed the bat algorithm, which has made breakthroughs in multi-objective optimization, engineering optimization, the knapsack problem, classification, resource scheduling, and other fields. Compared with other bionic algorithms, the bat algorithm has the advantages of a simple model, fewer parameters, strong robustness, and potential distribution, and its performance is better than genetic algorithms and particle swarm optimization.

3 Navigation Algorithms

Nature is extremely diverse, dynamic, robust, complex, and fascinating; it provides sufficient inspiration for humans to solve complex computing problems. The metaheuristic algorithm follows the natural law of "survival of the fittest" and realizes the evolution of species mainly through selection and mutation. The choice is the basis

of optimization, and variation is fundamental to random search or non-deterministic search [3]. In the past few decades, a large number of research studies have been concentrated in this field. According to the "survival of the fittest" strategy, many meta-heuristic algorithms have been developed and widely used in computer networks, robots, control systems, parallel processing, data mining, and many other areas [4, 5].

This section introduces the meta-heuristic algorithm from four aspects: ecosystem simulation algorithms, group intelligence algorithms, evolution algorithms, and artificial intelligence algorithms.

3.1 Ecosystem Simulation Algorithm

Learning from the nature system, people created several ecosystem simulation algorithms such as the distribution of habitats of biological species, biogeography-based optimization algorithm for migration and extinction, and an invasive weed optimization algorithm that simulates weed survival and evolution. Currently, these ecosystem simulations and optimization methods have been widely used in science, engineering, and other fields and have achieved encouraging results.

3.1.1 Biogeography Optimization Algorithm

In the nineteenth century, Alfred Wallace and Charles Darwin proposed the theory of biogeography to study the distribution, migration, and extinction of the biological species habitats. Inspired by the theory of biogeography, Simon proposed a new intelligent optimization algorithm, biogeography-based optimization (BBO), based on the study of mathematical models of biological species migration [6].

The basic idea of the BBO algorithm comes from the theory of biogeography. In the BBO algorithm, biological species live in multiple habitats, each of which is represented by the habitat suitability index (HSI). The factors associated with the HSI are rainfall, vegetation diversity, landform, land area, temperature, and humidity. This is called the appropriate index variable (AIV).

The HSI is one of the most important factors affecting the distribution and migration of species on habitats. There are many species in higher HSI habitats; conversely, there are fewer species in lower HSI habitats. It can be seen that habitat HSI is directly proportional to biodiversity. Due to the saturation of living space in high HSI habitats, a large number of species will migrate out to adjacent habitats with a small number of species moving in; low HSI habitats would have fewer species and more species would move in and fewer species move out. However, when the habitat HSI is kept at a low level, the species in the habitat tend to become extinct or find other habitats, namely mutations. Migrations and mutations are important operations of the BBO algorithm. Habitats enhance the exchange and sharing of information between species and enhance species diversity through migration and mutation. The BBO

algorithm has simple and effective features of evolutionary algorithms, and it has similar characteristics of other evolutionary algorithms:

1. The habitat suitability index (HSI) represents the value of the fitness function of the optimization problem and is a criterion for evaluating the quality of the solution.
2. The habitat represents the candidate solution, and the appropriate exponential variable AIV represents the solution feature.
3. The habitat migration and removal mechanism provide an understanding of the centralized information exchange mechanism.

The high HSI solution shares information to the low HSI solution at a certain eviction rate. In a single habitat species migration model, the population size of the habitat is directly proportional to the migration ratio. Set $\lambda(s)$, $u(s)$ as the population migration and immigration rate, respectively. When the population number is 0, the population migration rate is 0 and the population immigration rate is the largest; when the population size is $S_{\max}$, the population immigration rate is 0 and the population migration rate is the largest. When the populations of $\lambda(s)$ and $u(s)$ are equal, the migration rate and the immigration rate are equal, and the dynamic equilibrium state is reached. The immigration rate and the migration rate are obtained as follows

$$\begin{cases} \lambda(s) = I(1 - S/S_{\max}) \\ u(s) = ES/S_{\max} \end{cases} \tag{1}$$

4. Habitats will perform mutation operations according to the number of species; this can improve population diversity and make the algorithm more adaptive.

The mutation operation simulates the habitat ecological environment mutation, changes the number of habitat species, provides species diversity for the habitat, and provides more search targets for the algorithm. The probability of habitat mutation is inversely proportional to the probability of its species number, which is

$$m_s = m_{\max}\left(1 - \frac{P_S}{P_{\max}}\right) \tag{2}$$

where $m_{\max}$ is the maximum mutation rate; P_S is the probability that the number of species in the habitat is s; $P_{\max}$ is the maximum value of P_S;and m_s is the mutation probability that the number of species in the habitat is s.

3.1.2 Invasive Weed Optimization Algorithm

The invasive weed optimization algorithm (IWO) is a new intelligent optimization algorithm proposed by A. R. Mehrabian in 2006. The IWO algorithm is derived from the principles of weed evolution in nature and simulates the entire weed invasion process [7]. In the whole process of survival and reproduction of the population,

individuals who have always followed the adaptability can gain more chances of survival, that is, the most basic principle of survival of the fittest.

The weed invasion process needs to be realized through four processes: seed space diffusion, growth, reproduction, and competitive extinction. At the same time, this process is also the basic step of IWO algorithm optimization:

1. Population initialization

The main initialization parameters are initial weed number *G_SIZE*, population maximum size *P_SIZE*, maximum iteration number $\text{iter}_{\max}$, problem dimension *D*, maximum and minimum seed generation number $\text{seed}_{\max}$ and $\text{seed}_{\min}$, nonlinear index *n*, step initial value σ_{init}, and final value σ_{final}. *G_SIZE* initial weeds $\{x_1, x_2, \ldots, x_i, \ldots, x_{G\_SIZE}\}$ are randomly generated.

2. Population reproduction

In the IWO algorithm, fitness is the measure of individual environmental adaptability and individual reproductive ability. The IWO algorithm calculates the number of breeding seeds according to the individual fitness value. The number of seeds produced by a weed has a linear relationship with the weed fitness value. The calculation method is as follows

$$\text{num} = \text{ceil}\left(\left(\text{seed}_{\max} - \text{seed}_{\min}\right) \cdot \frac{f - f_{\min}}{f_{\max} - f_{\min}} + \text{seed}_{\min}\right) \tag{3}$$

where num is the number of seeds produced by weed individuals, $\text{seed}_{\max}$ and $\text{seed}_{\min}$ are the maximum and minimum number of seeds, f is the fitness of weeds, $f_{\max}$ and $f_{\min}$ are the maximum and minimum fitness values of the population, and ceil$(\cdot)$ is the upward rounding function. From Eq. (3.3), it can be seen that the greater the fitness value, the greater the number of seeds produced by weeds, that is, the weed reproduction ability increases with increases in fitness value.

3. Spatial diffusion.

In the IWO algorithm, weeds produce seeds with 0 as the mean value. σ_{iter} is a normal distribution of standard deviations distributed around weeds

$$\sigma_{\text{iter}} = \frac{(\text{iter}_{\max} - \text{iter})^n}{(\text{iter}_{\max})^n} \cdot (\sigma_{\text{init}} \cdot \sigma_{\text{final}}) + \sigma_{\text{final}} \tag{4}$$

where σ_{iter} is the standard deviation, $\text{iter}_{\max}$ and iter are the maximum and the current iteration algebra, σ_{init} and σ_{final} are the standard deviation initial value and the final value, and n is the nonlinear index. From Eq. (3.4), it can be seen that the dynamic change characteristic of the current standard deviation is that the search step length is farther away from the parent individual, so that the individuals with better fitness are gathered and the individuals with poor fitness value are eliminated.

4. Competitive selection.

The population size of weeds increased rapidly after several generations. When the population size is larger than the largest population size *P_SIZE*, the weeds and seeds are arranged according to their fitness values from largest to smallest. The top *P_SIZE* of the weeds and seeds with better fitness are selected as the next generation weeds, and the others are eliminated, which is in line with the natural pattern of survival of the fittest.

3.2 Swarm Intelligence Algorithm

Swarm intelligence (SI) is a general term for collective intelligence behavior in a class of decentralized self-organizing systems. It was proposed by Gerardo Beni in the molecular robot system in 1989. The SI system can be regarded as a group of simple individuals. There are interactions between one individual and the other, between the individual and its environment, and ultimately, intelligent behavior is characterized [8]. Although each individual follows extremely simple rules and the entire group has no central control, ultimately the interaction between local individuals leads to a global level of intelligent emergence.

The group intelligence algorithm has the following characteristics: the individual cooperatives in the group are distributed; there is no central control, and it is robust; it can cooperate through inter-individual communication and has scalability; and the individual ability is simple and convenient to implement. In addition to solving optimization problems, group intelligence algorithms can also be used in areas such as control and prediction [9, 10]. This section only discusses group intelligence optimization algorithms, including ant colony optimization, particle swarm optimization, artificial bee colony, fireflies, and bat algorithms.

3.2.1 Particle Swarm Algorithm

Particle swarm optimization (PSO) is a group intelligence algorithm proposed by Kennedy in 1995 based on the study of bird group behavior. The idea is derived from artificial life and evolutionary computation theory, which mimics the bird's flight foraging behavior. The group is optimized through bird group collaboration [11].

PSO is a branch of evolutionary computing and is an iterative-based optimization tool. The principle and mechanism of PSO are simple. It only evolves to the global optimal solution by updating the speed and position [12]. No gradient information is needed, the adjustable parameters are few, the algorithm is easy to implement, and the operation efficiency is high. Each optimization problem in PSO is treated as a particle in the search space. All particles have an adaptation value determined by the optimization function and have a velocity to determine the direction and velocity of the motion. The particles follow the current optimal particle search in the solution space [13]. The algorithm first initializes a bunch of random particles and then finds

the optimal solution through iteration. In each iteration, the particle updates its speed and position by tracking individual and global extrema. In the D-dimensional target search space, in particle swarm with the population of m, wherein the position of the ith particle in the dth dimension is x_{id}, the flying speed is v_{id}, the optimal position currently searched for the particle is $p_{id}(p\text{Best})$, and the current optimal position of the entire particle swarm is $p_{gd}(g\text{Best})$. The speed and position update equation is as follows

$$v_{id+1} = v_{id} + c_1 \times \text{rand}() \times (p_{id} - x_{id}) + c_2 \times \text{rand}() \times (p_{gd} - x_{id}) \quad (5)$$

$$x_{id+1} = x_{id} + x_{id} \quad (6)$$

where rand() is a random number in the set [0,1], and c_1, c_2 are acceleration factors.

The PSO algorithm framework is described below. Figure 1 shows the algorithm flow of the PSO.

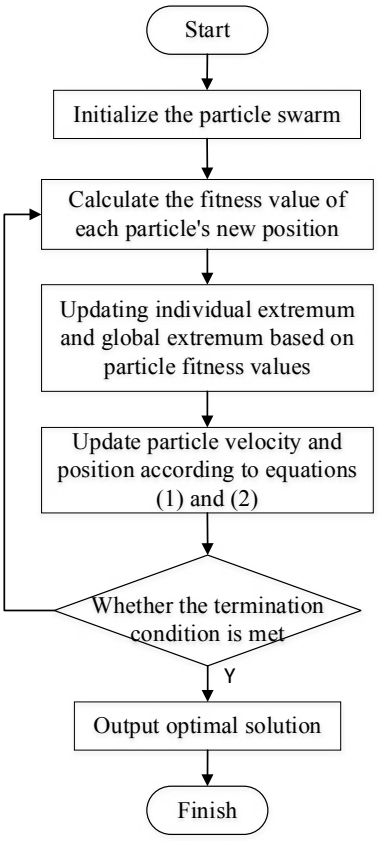


Fig. 1 Process of PSO algorithm

1. Initialize all individuals (particles), initialize their speed and position, and set the individual's historical optimality to the current position, while the best individual in the group is current.
2. In contemporary evolution, calculate the fitness function values of individual particles.
3. If the current fitness function value of the particle is better than the historical optimum, replace the historical optimum with the current position.
4. If the historical optimum of the particle is better than the global optimum, then the global optimum will be replaced by the historical optimum of the particle.
5. Update the speed and position for each particle according to Eqs. (5) and (6).
6. The evolution algebra is incremented by 1. If it has not reached the end condition, turn to the step of (2), otherwise output the optimal solution and then end the algorithm.

3.2.2 Ant Colony Algorithm

The ant colony algorithm is a population-based heuristic bionic evolution algorithm proposed by the Italian scholar Colorni in the early 1990s by simulating the collective path-seeking behavior of ants in nature. It adopts a distributed parallel computer system, which is easy to combine with other methods and has strong robustness. However, the long search time and being easy to fall into the local optimal solutions are its shortcomings [14].

The ant colony algorithm consists of two basic phases: the adaptation phase and the collaboration phase. In the adaptation phase, each candidate solution continuously adjusts its structure according to the accumulated information; in the cooperation phase, the better performance solutions can be generated through the information exchange of the candidate solutions, which is similar to the learning mechanism of learning automata. In order to understand the basic principles of the ant colony algorithm more clearly, the classic symmetric traveling salesperson problem (TSP) is explained:

If $C = \left\{c^1, c^2, \ldots, c^n\right\}$ is an n-city collection, $L = \left\{l_{ij}\ c_i, c_j \subset C\right\}$ is the two-two collection set of elements in set C, $d_{ij}(i, j = 1, 2, \ldots, n)$ is the Euclidean distance of l_{ij}, and $G = (C, L)$ is a directed graph. The purpose of the TSP problem is to find the shortest Hamilton circle from the directed graph G.

It is assumed that $\tau_{ij}(t)$ is the pheromones number on path (i, j) at time t, m is the number of ants in the ant colony, $\Gamma = \left\{\tau_{ij}(t)\ c_i, c_j \subset C\right\}$ is the set of residual pheromones on the l_{ij} of the set of elements in the set C at time t. The number of pheromones on each path is equal at the initial moment. The ant colony algorithm optimization is implemented on the directed graph $g = (C, L, \Gamma)$ by the following criteria:

1. Transition probability criteria

During the movement of the ants $k(k = 1, 2, \ldots, m)$, the direction of the transfer is determined according to the number of pheromones of each path. Artificial ants in the algorithm have a memory function.

The Tabu table $\text{Tabu}_k = (k = 1, 2, \ldots, m)$ is used to record the city where the ants k are currently traveling. During the search process, the ant calculates the transition probability based on the number of pheromones and path heuristic information on each path. $p_{ij}^k(t)$ represents the transition probability of ant k moving from element i to element j at time t, which is

$$p_{ij}^k(t) = \begin{cases} \frac{[\tau_{ij}(t)]^\alpha [\eta_{ik}(t)]^\beta}{\sum [\tau_{ij}(t)]^\alpha [\eta_{ik}(t)]^\beta} & j \in \text{allowed}_k, s \subset \text{allowed}_k \\ 0 & \text{else} \end{cases} \tag{7}$$

where $\text{allowed}_k = \{C - \text{Tabu}_k\}$ indicates that the ant k next allows the city to be selected, α indicates the relative importance of the trajectory, and β indicates the relative importance of visibility. $\eta_{ij}(t)$ is a heuristic function, which is defined as

$$\eta_{ij}(t) = 1/d_{ij} \tag{8}$$

These heuristic functions represent the degree to which an ant moves from element to an element.

2. Local adjustment criteria

Local adjustment is performed by each ant during the establishment of a solution. After h units of time, the number of local pheromones between the two element states is adjusted according to the following equation

$$\tau_{ij}(t + h) = (1 - \zeta)\tau_{ij}(t) + \zeta\tau_0 \tag{9}$$

$$\tau_0 = 1/(nl_{\min}) \tag{10}$$

where $\zeta \subset [0, 1]$, $l_{\min}$ represents the distance between the two nearest elements in the set C.

3. Global adjustment guidelines.

Ants that have generated a global optimal solution have an opportunity to make global adjustments. The global adjustment rule is

$$\tau_{ij}(t + n) = (1 - \rho)\tau_{ij}(t) + \rho\Delta\tau_{ij}(t) \tag{11}$$

$$\Delta\tau_{ij}(t) = \sum_{k=1}^{m} \Delta\tau_{ij}^k(t) \tag{12}$$

where ρ is the volatilization coefficient, $\rho \subset [0, 1]$, $\Delta\tau_{ij}(t)$ is the increment of the number of pheromones on the path ij in this cycle; and $\Delta\tau_{ij}^{k}(t)$ is the amount of information left by the kth ant in the path ij in this cycle.

3.2.3 Firefly Algorithm

The firefly algorithm is a simulation of the biological characteristics of natural firefly luminescence and is a stochastic optimization algorithm based on group search. The algorithm proposed by Cambridge scholar Xin-She Yang is called the firefly algorithm (FA).

The bionic principle of FA is to simulate the firefly individual in the natural world by using the points in the search space and to simulate the search and optimization process into the individual attraction and movement process of the firefly, and to measure the problem function of the problem into the individual's position and the superiority and inferiority of the individual. Process analogy is an iterative process that replaces poorly feasible solutions with good feasible solutions in the search and optimization process [15].

The FA contains two elements of brightness and attractiveness. The brightness reflects the position of the firefly and determines its moving direction. The degree of attraction determines the moving distance of the firefly, and the target is optimized through continuous updating of brightness and attractiveness [16]. From a mathematical point of view, the firefly algorithm optimization mechanism is described as follows:

The relative fluorescence of fireflies is

$$I = I_0 \times \mathrm{e}^{-\gamma r_{ij}} \tag{13}$$

where I_0 is the maximum fluorescence brightness of fireflies, that is, the fluorescence intensity at $r = 0$, which is proportional to the objective function value; γ is the light intensity absorption coefficient; and r_{ij} is the spatial distance between fireflies i and j.

The firefly attraction is

$$\beta = \beta_0 \times e^{-\gamma r_{ij}^2} \tag{14}$$

where β_0 is the maximum attraction, that is, the attraction rate at $r = 0$; γ is the light absorption coefficient, and r_{ij} is the distance between fireflies i and j.

The location update of the fireflies that are attracted to the fireflies is determined by

$$x_i = x_i + \beta \times (x_j - x_i) + \alpha \times (\mathrm{rand} - 1/2) \tag{15}$$

where x_i and x_j are the spatial positions of fireflies i and j; α is the step factor, which is a constant on [0, 1]; and rand is a uniformly distributed random factor at [0, 1].

The FA optimization process randomly distributes the firefly population in the solution space. Each firefly emits different fluorescence brightness depending on the location. By comparison, it can be seen that the high brightness fireflies can attract low brightness fireflies to move to themselves. The moving distance depends mainly on the degree of attraction. In order to increase the search area and avoid prematurely falling into local optimum, a disturbance item $\alpha \times (\text{rand} - 1/2)$ is added during the location update process to calculate the updated position according to Eq. (15) [17, 18]. After multiple moves, all individuals would gather at the highest brightness fireflies' position to achieve optimization.

3.2.4 Artificial Bee Colony Algorithm

The artificial bee colony (ABC) was proposed by the Turkish scholar Karaboga in 2005. The basic idea inspires the bee colony to collaborate to complete the task of collecting honey through the individual division of labor and information exchange. Although a single bee has limited capacity, the entire bee colony can easily find high-quality honey sources without a unified command.

Compared with the classical method, the ABC algorithm has almost no requirements for the objective function and the constraint. In the search process, external information is not used and the fitness function is used as an evolutionary basis. The artificial intelligence technology characterized by "generation + test" is formed [19]. The ABC algorithm has the characteristics of simple operation, fewer control parameters, high search accuracy, and strong robustness.

When the ABC algorithm solves the optimization problem, the honey source position is abstracted as the solution space point, representing the potential solution of the problem, the honey source quality corresponds to the fitness value of the solution, and NP is the number of honey sources. The ABC algorithm divides the bee colony into three types: leading bees, following bees, and scout bees. The leading bees and follow bees each account for half of the bee colony, the number is equal to the number of honey sources, and each honey source has only one leading bee at the same time.

Let solve the problem dimension as D. The honey source position is represented as $X_i^t = [X_{i1}^t, X_{i2}^t, \ldots, X_{iD}^t]$, where t indicates the current number of iterations. In $x_{id} \in (L_d, U_d)$, L_d and U_d represent the lower and upper limits of the search space, and $d = 1, 2, \ldots, D$. The initial position of the honey source i is randomly generated in the search space according to Eq. (16)

$$x_{id} = L_d + \text{rand}(0, 1)(U_d - L_d) \tag{16}$$

At the beginning of the search, the leading bee searches and generates a new honey source around the honey source i according to Eq. (17)

$$v_{id} = x_{id} + \varphi(x_{id} - x_{jd}) \tag{17}$$

where d is a random number in the set [1, D], and it indicates that the leading bee randomly selects a one-dimensional search; $j \in \{1, 2, \ldots, NP\}$, $j \neq i$ indicates that randomly selecting one of the honey sources does not equal i; φ is an uniformly distributed random number in the set [–1,1], determining the magnitude of the disturbance. When the new honey source $V_i = [v_{i1}, v_{i2}, \ldots, v_{id}]$ has better fitness than X_i, V_i is used to replace X_i by the greedy choice method, otherwise X_i is reserved. After all the bees have completed the operation Eq. (17), they fly back to the information exchange area to share the honey source information. The following bee share information according to the leading bee, and Eq. (18) is used to calculate the probability to follow

$$p_i = \text{fit}_i / \sum_{i=1}^{\text{NP}} \text{fit}_i \tag{18}$$

The following bee produces a uniformly distributed random number r at [0, 1]. If p_i is bigger than r, then the following bee generates a new honey source around the honey source i according to Eq. (17), and then the same greedy selection method as the leading bee is used to determine the retained honey source.

During the search, if the honey source X_i reaches the threshold limit after trial search iterations and does not find a better honey source, this honey source X_i will be given up. Correspondingly, this leading bee character is turned into the scout bee. The scout bee will randomly generate a new honey source in the search space to replace X_i, and the above process is expressed as Eq. (19)

$$X_i^{t+1} = \begin{cases} L_d + \text{rand}(0, 1)(U_d - L_d), \text{trial}_i \geq \text{limit} \\ X_i^t, \text{trial}_i < \text{limit} \end{cases} \tag{19}$$

For the sake of generality, to minimize the optimization problem in the ABC algorithm, the solution fitness evaluation is calculated according to Eq. (20)

$$\text{fit}_i = \begin{cases} 1/(1 + f_i) & f_i \geq 0 \\ 1 + \text{abs}(f_i) & f_i < 0 \end{cases} \tag{20}$$

where f_i is the function value representing the solution.

3.2.5 Bat Algorithm

By simulating the biological characteristics of bats using ultrasonic search and prey on prey, Xin-She Yang proposed a bat algorithm based on stochastic optimization. The algorithm has the characteristics of a simple model, fast convergence, and parallel processing [20].

The accurate positioning and predation of bats in complex environments are inspired by the optimization of its biological mechanism. The bat algorithm is based on the algorithm of population evolution. Firstly, the population is randomly initialized in the feasible solution space, that is, the initial position and velocity of the individual are determined. The optimal position of the group is found by evaluating the group. Then, the individual flight speed and position are updated according to Eqs. (21) and (22)

$$v_i^t = v_i^{t-1} + \left(x_i^t - x^*\right) \cdot f_i \tag{21}$$

$$x_i' = x_i^{t-1} + x_i^t \tag{22}$$

where v_i^{t-1} and v_i^t indicate the flying speed of bat i at the moment of $t-1$ and t; x_i^t indicates the spatial position of bat i at t time; x^* indicates the position of the best bat in the current group; f_i is the pulse frequency used for bat i to search for prey; and $f_i \in \left[f_{\min}, f_{\max}\right]$ is the search pulse frequency range for bat i.

It can be seen from the biological mechanism that in the process of searching for prey, the pulsing ultrasonic pulse sound in the initial stage of the bat is a powerful and low frequency, which helps to search a wider space. After the prey is found, the pulse intensity is gradually reduced and the number of pulse emission is increased. In order to grasp the spatial position of the prey accurately, Eqs. (23) and (24) are used to simulate the search characteristics

$$r_i^{t+1} = r_i^0[1 - \exp(-\gamma \times t)] \tag{23}$$

$$A_i^{t+1} = \alpha \times A_i^t \tag{24}$$

where r_i^0 indicates the maximum pulse frequency of bat i; r_i^{t+1} indicates the bat i pulse frequency at time $t+1$; γ is a constant greater than zero which indicates the pulse frequency increase factor; A_i^t indicates the intensity of the bat i firing a pulse at time t; and α is the pulse intensity attenuation coefficient which is usually a constant selected in the range of [0, 1].

The bat algorithm flowchart is shown in Fig. 2, where R_1, R_2 are randomly generated numbers.

3.2.6 Cuckoo Algorithm

In 2009, Xin-She Yang proposed a cuckoo search algorithm based on long-term research in mathematical modeling and engineering optimization. The algorithm is proposed based on the long-term study of the habit of the cuckoo (the specific behavior of the cuckoo lying eggs is observed) and combined with the Lévy flight of the flying creature. Since this algorithm is easy to understand, easy to perform,

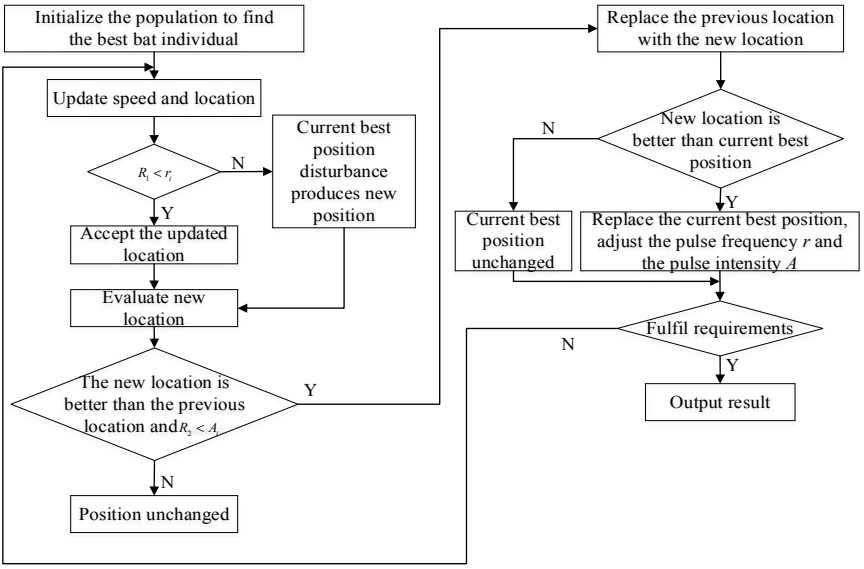


Fig. 2 Process of bat algorithm

and has a good random search path, it has been successfully applied to some difficult problems [21].

To understand the cuckoo's search for nests, the algorithm is described using the idealized conditions described below:

1. Each female cuckoo produces only one egg at a time. After observing the suitable bird's nest, randomly select one of the nests to place your own eggs.
2. After a large number of calculations, the bird eggs (optimal solution) in the best bird nest after calculation are saved.
3. The number of nests capable of placing cuckoo eggs is a fixed value and is set according to the algorithm.

Set the probability that the nest bird will find the foreign bird. If the main bird of the nest discovers the foreign bird's egg, the nest bird has two treatment methods: Discard the eggs of the foreign bird; discard the current bird's nest, and build a nest in the new position. Treating the bird eggs in each nest as a solution, each cuckoo bird egg represents a new solution, by replacing the bad bird eggs with good bird eggs, and thus obtaining a global optimal solution. Through the above ideal state assumption, the equation of cuckoo searching position and path is updated as follows

$$x_i^{(t+1)} = x_i^{(t)} + \alpha \oplus L(\lambda) \quad i = 1, 2, \ldots, n \tag{25}$$

where $x_i^{(t)}$ is the position of the ith bird's nest in the tth generation, $\oplus$ is point-to-point multiplication, the step size factor is represented by α, $L(\lambda)$ is the Lévy random search path, $L \sim u = t^{-\lambda}$, $(1 < \lambda \leqslant 3)$. After the position update, the

random number $r \in (0, 1)$ is compared with p_a. Therefore, if $r > p_a$, the bird's nest position $x_i^{(t+1)}$ is randomly changed; if $r < p_a$, the bird's nest position remains unchanged. After the location is updated, the bird's nest position with a good test value is kept.

3.3 Evolutionary Algorithm

The evolutionary algorithm is an "algorithm cluster." Although it has many changes, including different genetic expression patterns, different crossovers, mutation operators, and different regeneration and selection methods. Its inspiration comes from the evolution of natural organisms. Compared with traditional optimization algorithms, evolutionary computation is a mature global optimization method with high robustness and wide applicability. It has self-organizing, self-adaptive, and self-learning properties. This algorithm can effectively deal with the complex problems that traditional optimization algorithms are difficult to solve without being limited by the nature of the problem.

3.3.1 Genetic Algorithm

In 1975, Professor Holland of the USA first proposed the genetic algorithm in "Adaptability of Nature and Artificial Systems," which was derived from Darwin's theory of evolution, Weizmann's species selection theory, and Mendelian population genetics. It is a random search algorithm that draws on the natural selection and natural genetic mechanisms of the biological world.

The genetic algorithm is an adaptive and global probabilistic search algorithm that simulates the genetic and evolutionary process of the survival of the fittest in the natural environment. It starts with a population that represents a potential set of problems. First, the phenotype is mapped to the genotype, i.e., the code, so that the solution space is mapped to the coding space. Each code corresponds to a solution, called a chromosome or an individual. After the initial population is produced, according to the principle of survival of the fittest, the better and better approximate solution is generated by the evolution of the generations. Each generation selects individuals according to the degree of individual fitness in the problem domain and uses natural genetic operators to combine and mutate, producing a population representing the new solution set [22]. This process is similar to natural evolution, making the offspring population more adaptable to the environment than the previous generation. The best individual in the last generation's population can be decoded as the approximate optimal solution. The steps to solve a problem using a genetic algorithm are as follows:

1. Establish a mathematical model.
2. Coding, which uses a well-designed algorithm to map phenotypes to individual genotypes.
3. Decoding, the genetic operator only works on the encoded chromosome, and the objective function value is calculated by the individual phenotype to determine the pros and cons of the chromosome.
4. Determine the fitness conversion rules, the values of the solution space corresponding to the chromosomes may vary greatly, and a certain conversion is needed to make it suitable for a quantitative evaluation of the individual's advantages and disadvantages.
5. Estimate genetic operators, i.e., design crossovers, mutations, and selection operators. The genetic operator has a great relationship with the problem to be optimized and the coding scheme of the chromosome.
6. Determine the operating parameters, including the crossover probability, the probability of variation, and the number of populations.

The process for solving the problem using the genetic algorithm is shown in Fig. 3.

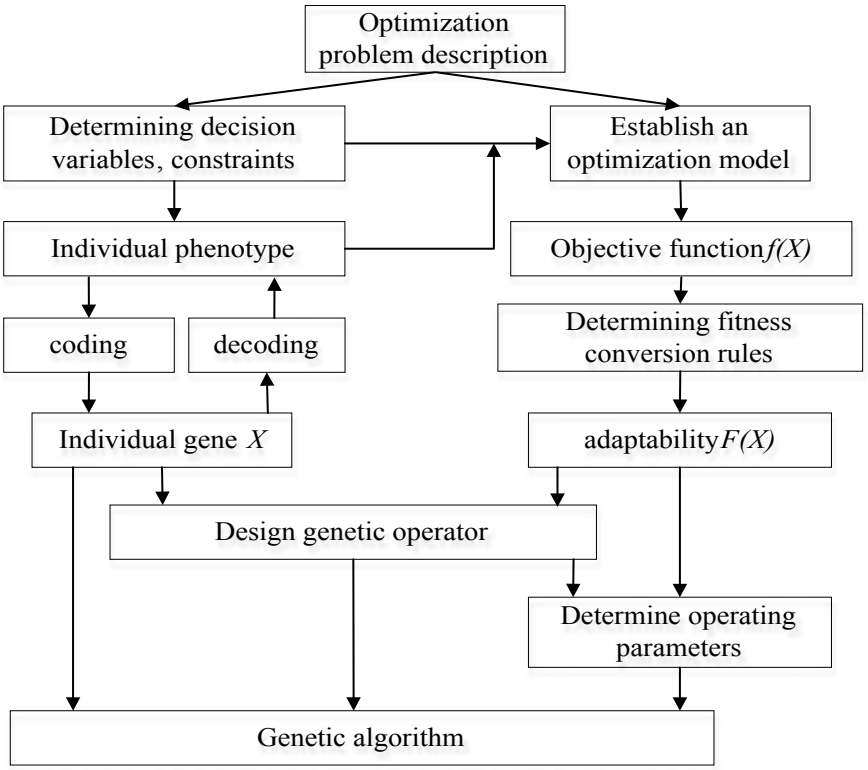


Fig. 3 Process of genetic algorithm

3.3.2 Differential Evolution Algorithm

Differential evolution (DE) is a heuristic random search algorithm based on group difference, which was proposed by R. Storn and K. Price for solving Chebyshev polynomials [23]. Due to its simple principle, few controlled parameters, and strong robustness, the differential evolution algorithm has attracted more and more scholars' attention. In recent years, DE has been widely used in constrained optimization calculations, cluster optimization calculations, nonlinear optimization control, and neural network optimization.

DE uses real-number coding, and its algorithm principle is similar to the genetic algorithm. The evolution process is the same as the genetic algorithm: mutation, crossover, and selection. The selection strategy in the DE algorithm is usually the tournament selection, and the crossover operation is similar to the genetic algorithm. However, the differential strategy is used in the mutation operation, that is, the individual is perturbed by the difference vector between the individuals in the population to realize the individual variation. The DE mutation method effectively utilizes the population distribution characteristics, improves the search ability of the algorithm, and avoids the deficiency of the variation method in the genetic algorithm.

For optimization problems

$$\min f(x_1, x_2, \ldots, x_D) \quad \text{s.t} \quad x_j^L \leqslant x_j \leqslant x_j^U, \quad j = 1, 2, \ldots, D \tag{26}$$

where D is the dimension of the solution space, and the x_j^L and x_j^U pairs represent the upper and lower bounds of the range of values of the jth component x_j. The DE algorithm flow is shown as follows:

1. Initialize the population

   To initialize the population, the initial population is randomly generated by

$$x_{j,i}(0) = x_{j,i}^L + \text{rand}(0, 1) \cdot \left(x_{j,i}^U - x_{j,t}^L\right) \tag{27}$$

   where $x_i(0)$ represents the ith "chromosome" (or individual) of the 0th generation in the population, and $x_{j,i}(0)$ represents the jth "gene" of the ith "chromosome" of the 0th generation. NP represents the population size, and rand(0, 1) represents a random number uniformly distributed in the interval of (0, 1).

2. Variation operation

   DE implements individual variation through a differential strategy, which is also an important indicator which is different from the genetic algorithms. In DE, the two different individuals in the population are selected in the common differential strategy. Then, the vector difference is scaled and performed the vector synthesis with the individual to be mutated.

$$v_i(g + 1) = x_{n1}(g) + F \cdot (x_2(g) - x_{t3}(g)) \quad i \neq r_1 \neq r_2 \neq r_3 \tag{28}$$

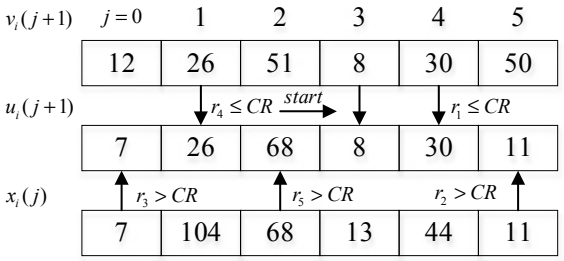


Fig. 4 Process of cross-operation

where F is the scaling factor, and $x_i(g)$ is the ith individual in the gth generation population. In the process of evolution, in order to ensure the validity of the solution, it is necessary to judge whether each "gene" in the "chromosome" satisfies the boundary condition or not. If the boundary condition is not satisfied, the "gene" is regenerated by a random method (the same as the initial population generation method).

After the mutation of the gth generation population $\left\{ x_i(g) | x_{j,i}^L \leqslant x_{j,i}(g) \leqslant x_{j,i}^U, i = 1, 2, \ldots, NP; j = 1, 2, \ldots, D \right\}$, an intermediate $\left\{ v_i(g+1) | v_{j,i}^L \leqslant v_{j,i}(g+1) \leqslant v_{j,i}^U, i = 1, 2, \ldots, NP; j = 1, 2, \ldots, D \right\}$ is produced.

3. Cross-operation

The inter-individual crossovers are carried out among the gth generation population $g\{x_i(g)\}$ and the variant intermediate $\{v_i(g+1)\}$.

$$u_{j,i}(g+1) = \begin{cases} v_{j,i}(g+1) & \text{if } \text{rand}(0,1) \leqslant CR \text{ or } j = j_{\text{rand}} \\ x_{j,i}(g) & \text{otherwise} \end{cases} \tag{29}$$

where CR is the crossover probability, and j_{rand} is a random integer of $[1, 2, \ldots, D]$.

Figure 4 is a schematic diagram of the crossover of six gene positions "chromosomes." In order to ensure that at least one "gene" of each "chromosome" of the variant intermediate $\{v_i(g+1)\}$ is passed on to the next generation, the first cross-operated gene is randomly taken out of the $j_{\text{rand}}^{\text{th}}$ "gene" in $v_i(g+1)$ as the j_{rand} − position "gene" of "chromosome" $u_i(g+1)$. Subsequent cross-operations are performed by crossover probability CR to select alleles of $x_i(g)$ or $v_i(g+1)$ as the alleles.

4. Select the operation

The greedy algorithm is used in DE to select individuals who enter the next generation of the population

$$x_i(g+1) = \begin{cases} u_i(g+1) & \text{if } f(u_i(g+1)) \leqslant f(x_i(g)) \\ x_i(g) & \text{otherwise} \end{cases} \tag{30}$$

3.3.3 Harmony Search Algorithm

The harmony search (HS) algorithm is an intelligent optimization algorithm proposed by Korean scholar Geem in 2001. The algorithm simulates music creation in which musicians rely on their own memories to achieve a wonderful harmony state by repeatedly adjusting the pitch of each instrument in the band. The harmony search algorithm compares the harmony of the instrument tones with the solution vector of the optimization problem, and the evaluation is the value of each corresponding objective function. The algorithm introduces two main parameters, namely the memory value probability HMCR and fine-tuning probability PAR [24]. Related research shows that the harmony search algorithm demonstrates better optimization performance than genetic algorithms and simulated annealing algorithms in solving multi-dimensional function optimization problems.

The harmony search algorithm first generates HMS initial solutions and places them into the harmony memory HM. Then, the specific steps of the harmony memory randomly searching for a new solution are: (1) a random number of rand in 0–1 is randomly generated; (2) if rand $<$ HMCR, the new solution is obtained through randomly searching in HM; and (3) otherwise, the new solution is searched within the possible range of variables outside of the harmony memory. The new solution taken from HM is then locally perturbed by the fine-tuning probability of PAR. Finally, it is judged whether the value of the new solution objective function is better than the worst solution in HM or not. If it is, the harmony library is updated and iterated until the predetermined number of iterations $T_{\max}$ is reached. The calculation steps for harmony search are shown as follows.

1. Defining problems and parameter values

   Suppose the problem is minimized and its form is as follows

$$\min f(x) \quad \text{s.t.}\, x_i \in X_i, \quad i = 1, 2, \ldots, N \tag{31}$$

where $f(x)$ is the objective function, x is the solution vector formed by the decision variable $x_i (i = 1, 2, \ldots, N)$, and the value range of each decision is X_i. For the discrete variable $X_i = (x_i(1), x_i(2), \ldots, x_i(K))$, the continuous variable $X_i : x_i^L \leq X_i \leq x_i^U$, N is the number of decision variables, and K is the number of possible values of the discrete variable. The algorithm parameters obtain the size of the harmony memory HMS, the probability of the acoustic memory HMCR, the pitch fine-tuning probability PAR, the pitch trimming bandwidth bw, and the number of creations $T_{\max}$.

2. Initialization and sound memory

   There are HMS harmony $x^1, x^2, \ldots, x^{\text{HMS}}$ randomly generated and being placed into the harmony memory. The form of the harmony memory is described as follows

$$\mathrm{HM} = \left[\begin{array}{c|c} x^1 & f(x^1) \\ x^2 & f(x^2) \\ \vdots & \vdots \\ x^{\mathrm{HMS}} & f(x^{\mathrm{HMS}}) \end{array}\right] = \left[\begin{array}{cccc|c} x_1^1 & x_2^1 & \cdots & x_N^1 & f(x^1) \\ x_1^2 & x_2^2 & \cdots & x_N^2 & f(x^2) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_1^{\mathrm{HMS}} & x_2^{\mathrm{HMS}} & \cdots & x_N^{\mathrm{HMS}} & f(x^{\mathrm{HMS}}) \end{array}\right] \tag{32}$$

3. Generate new harmony

To generate a new harmony $x_i' = \left(x_1', x_2', \ldots, x_N'\right)$, each tone of the new harmony $x_i'(i = 1, 2, \ldots, N)$ is generated by a learning harmony memory, pitch tuning, and a randomly selected pitch mechanism.

4. Update the harmony memory

The new solution in step (3) is evaluated. If it is better than the one with the worst function value in HM, the new solution is updated to HM. The specific operations are as follows

$$f(x) < f\left(x^{\mathrm{worst}}\right) = \max_{j=1,2,\ldots,\mathrm{MHS}} f(x^j), \quad \text{then } x^{\mathrm{worst}} = x' \tag{33}$$

5. Check if the algorithm termination condition is reached

Repeat steps (3) and (4) until the number of iterations $T_{\max}$ is reached. The flowchart of the harmony search algorithm is shown in Fig. 5.

3.4 Artificial Intelligence Algorithm

Artificial intelligence (AI) is not a new term. It has been more than 60 years since the birth of the Dartmouth Conference in 1956. Recently, artificial intelligence has ushered in the third development climax with the breakthrough of the computer, and the rise of big data, cloud computing, and deep learning. Artificial intelligence is an algorithm that uses computers to simulate certain thinking processes and intelligent behaviors of humans [25]. It mainly includes the principle of computer-implemented intelligence and the manufacture of computers similar to human brain intelligence, enabling computers to achieve higher-level applications. Because artificial intelligence algorithms have excellent learning abilities and strong robustness, it has become a hotspot in the field of path planning.

3.4.1 Artificial Neural Network Algorithm

The artificial neural network algorithm is a very good algorithm in the field of artificial intelligence. It mainly simulates animal neural network behavior and performs distributed parallel information processing. However, its application in path planning

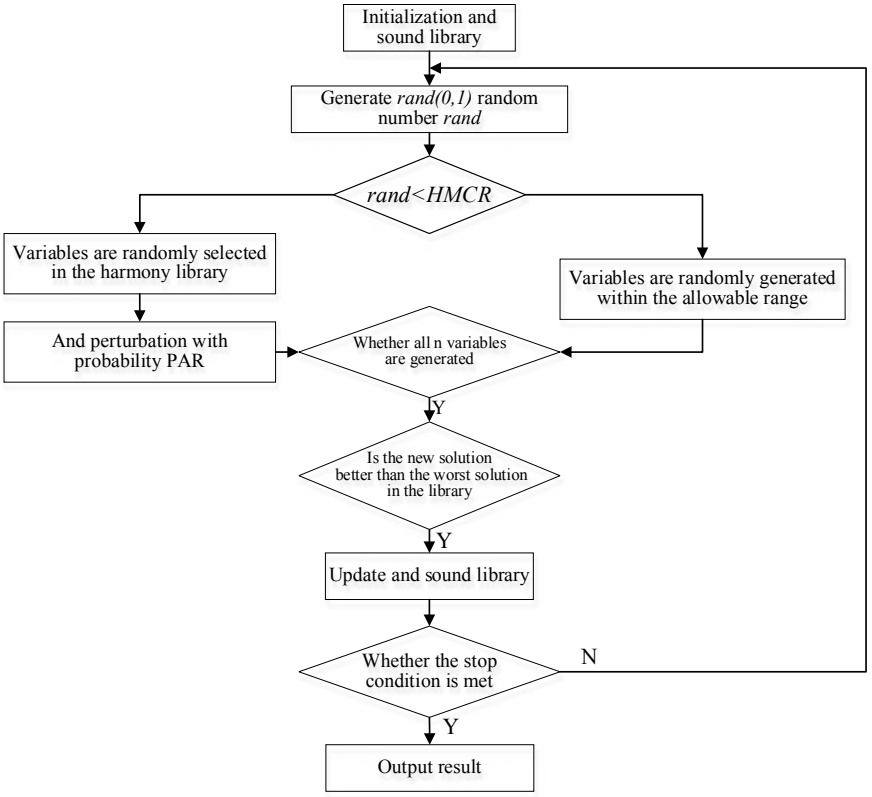


Fig. 5 Process of harmony search algorithm

is not successful because the complex and variable environment in path planning is difficult to describe with mathematical equations. If a neural network is used to predict points outside the distribution space of the learning samples, the effect will be rather small. Although neural networks have excellent learning ability, poor generalization is a fatal flaw. However, because of its strong learning ability and robustness, its combination with other algorithms has become a hot topic in the field of path planning.

In the twentieth century, Hopfield successfully applied artificial neural networks to combinatorial optimization problems. The multi-layer feedback learning algorithm constructed by McClelland and Rumelhart successfully solved the "exclusive OR problem" of single hidden layer cognitive networks and other identification problems. The principle of artificial neural network algorithm is as follows.

The collision penalty function of a path is defined as the sum of the collision penalty functions of each path point, and the collision penalty function of one point is obtained through its neural network representation of each obstacle. The obstacle is assumed to be a polygon [25].

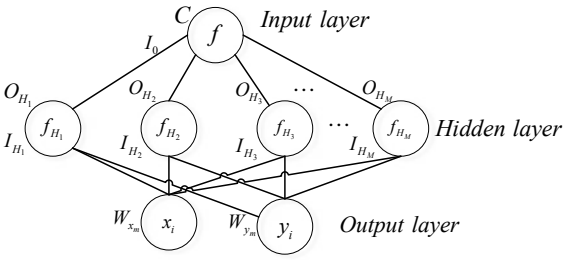


Fig. 6 Neural network from point to obstacle penalty function

Figure 6 shows a neural network of a penalty function from point to obstacle. The two nodes of the bottom layer represent the coordinates x, y of a given path point. Each node of the middle layer corresponds to the inequality constraint of one side of the obstacle. The connection weight coefficient of the bottom layer and the middle layer is equal to the coefficient before x, y in the inequality. The threshold of each node in the middle layer is equal to the constant term in the corresponding inequality.

The operational relationship of the continuous network is as follows

$$C = f(I_o) \tag{34}$$

$$I_o = \sum_{m=1}^{M} O_{Hm} + \theta_T \tag{35}$$

$$O_{Hm} = f(I_{Hm}) \tag{36}$$

$$I_{Hm} = w_{xm}x_i + w_{ym}y_i + \theta_{Hm} \tag{37}$$

where C is the top node output; I_O is the top node input; θ_T is the top node threshold; O_{Hm} is the mth node output of the middle layer; I_{Hm} is the mth node input of the middle layer; θ_{Hm} is the mth of the middle layer node threshold; and w_{xm}, w_{ym} are the mth inequality constraint condition coefficients. The excitation function is a commonly used sigmoid function

$$f(x) = \frac{1}{1 + \mathrm{e}^{-x/T}} \tag{38}$$

where T is the "temperature" in the simulated annealing method, and it changes according to the following rules

$$T(t) = \frac{U_0}{\log(1 + t)} \tag{39}$$

The energy of the entire path corresponding to the part of the collision function is as follows

$$E_C = \sum_{i=1}^{N} \sum_{k=1}^{K} C_i^k \tag{40}$$

where K is the number of obstacles, N is the number of path points, and C_i^k is the collision function of the ith path point $P(x_i, y_i)$ to the kth obstacle.

The energy corresponding to the length of the path is defined as the sum of the squares of the lengths of all the segments. That is, all path points $P(x_i, y_i), i = 1, 2, \ldots, N$ is defined as follows

$$E_1 = \sum_{i=1}^{N-1} \left[(x_{it}, 1 - x_i)^2 + (y_{it}, -y_i)^2 \right] \tag{41}$$

The total energy function for the entire path is defined as

$$E = w_l E_l + w_c E_c \tag{42}$$

where w_c and w_l represent the weighting of each part.

Since the entire energy is a function of each path point, by moving each path point, it moves in the direction of energy reduction, and finally, the total energy minimum path can be obtained. The dynamic motion equation of the point $P(x_i, y_i)$ is

$$\begin{aligned} \dot{x}_i &= -Z\left[2w_l(2x_i - x_{i-1} - x_{i+1}) + w_C \sum_{k=1}^{K} f'\left((I_O)_i^k \sum_{m=1}^{M} f'\left((I_{Hm})_i^k\right) w_{xm}^k \right) \right] \\ \dot{y}_i &= -Z\left[2w_l(2y_i - y_{i-1} - y_{i+1}) + w_C \sum_{k=1}^{K} f'\left((I_O)_i^k \sum_{m=1}^{M} f'\left((I_{Hm})_i^k\right) w_{ym}^k \right) \right] \end{aligned} \tag{43}$$

where

$$f'(\cdot) = \frac{1}{T} f(\cdot)(1 - (\cdot)) \tag{44}$$

3.4.2 Deep Learning Algorithm

In 2006, Hinton et al. proposed a deep self-encoding and self-decoding network, which solved the problem of deep neural network training and completely pushed the neural network into the era of deep learning.

Advances in deep learning have made it possible to perform path planning on the original image, enabling end-to-end models with learning generalization capabilities. At present, research on the path planning problem with deep learning is in its infancy,

and it is necessary to model the path planning problem with the reinforcement learning idea. In the framework of reinforcement learning theory, path planning can be regarded as a continuous control problem consisting of a series of single-step decisions. The method of using deep learning to represent decision-related functions is the deep-enhanced learning [26]. The mainstream model for path planning is a neural network based on a reflective strategy that lacks the ability to deal with long-term planning problems. Moreover, the agency experience needed to enhance learning is sometimes difficult to obtain and requires much time and money. Therefore, at the theoretical and practical application level, the demand for intelligent algorithms with end-to-end learning and planning capabilities is imminent, and its implementation and improvement still face difficulties.

Although the history of deep learning and deep-enhanced learning-related research is rather short, its use in path planning is numerous. According to the research progress, related research can be divided into two levels.

1. The path planning algorithms based on deep learning

Representative research has applications in motion planning for planar robots. The essence of this method is to use the deep neural network to parse the original input data tensor, get a set of representations, and then combine the A^* algorithm to plan. The advantage of this method is that reliable data can be obtained without the loss of data representation of the original data processing. The disadvantage is that the ability to generalize the environment is weak.

2. The path planning algorithms based on deep-enhanced learning

The features of the depth-enhanced learning path algorithm are that it only needs to specify the planning target, such as no collision, reaching the destination, and short path, and it iteratively updates the neural network through environmental training or simulation experiments. The neural network used for deep-enhanced learning can judge the current input and output the decision for path planning. The working principle is a neural network-based conditional reflection strategy. Depending on the fitting ability of the neural network, sample training, through a large number of scenes, makes the neural network react in similar scenarios to achieve the purpose of path planning. If the training sample is insufficient, or the number of iterations is insufficient, the model could be unstable and lack generalization ability.

Deep-enhanced learning is the algorithm closest to human thinking action patterns in path planning algorithms, and it is an important attempt to implement general artificial intelligence.

Deep-enhanced learning integrates both perception and decision making. The core idea of depth-enhanced learning based on the value function is to use the deep neural network to approximate the value function in learning and then make decisions based on the value of the value function. The optimal representative integration is a deep Q network.

The core idea of the deep Q network is to approximate the optimal action-value function with the neural network and obtain the neural model parameters through the Q learning algorithm, so as to obtain the optimal strategy corresponding to the state

and action [23]. Here, the $Q(s,\ a)$ function is used to represent the evaluation function. The Q learning algorithm in the traditional reinforcement learning is repeated using the following equation

$$Q(s,a) = Q(s,a) + \alpha\left(R_{t+1} + \gamma \max_{a'} Q\left(s', a'\right) - Q(s,a)\right) \tag{45}$$

where s, a are the current actual state and the current actual action, s', a' are the successor state and subsequent actions, and α is the update step size.

In the deep Q network, the iterative update is the weight of the neural network, and the weight of the neural network is differentiable. Here, only the objective function needs to be defined, and the derivative of the neural network weight is obtained. The model parameters are updated according to the update algorithm of the neural network until convergence. The neural network weight is recorded as θ, and the training data set is the agent experience D. The mean squared loss function of the i^{th} iteration can be defined without using other neural network training techniques in the following form

$$L_i(\theta_i) = E_{(s,a,r,s)-U(D)}\left(R_{t+1} + \gamma \max_{a'} Q\left(s', a'; \theta_i\right) - Q(s,a;\theta_i)\right)^2 \tag{46}$$

According to the equation, the training of the model can be realized by calculating the derivative about θ of the loss function using back propagation through the computational graph of the neural network and updating the model parameters by the derivative.

4 Development Tendency of Navigation Algorithms

Bionics was founded in the middle of the last century. Many scientists seek new inspiration for artificial systems from biology. Some scientists independently developed simulated evolutionary algorithms suitable for the optimization of complex problems in the real world from the mechanism of biological evolution. The development status and tendency of heuristic optimization algorithms applied to navigation problems are described below.

4.1 Development Status of Navigation Algorithms

After the initial idea of ant colony optimization was put forward by Dorigo et al. in 1991, not much attention was paid to it. It was not until 1996 that the theory and methods of ant colony optimization were systematically elaborated that an upsurge in research occurred [27]. A large number of improvement and application researches appeared in this area.

In 1998, the first ant colony algorithm seminar was held in Brussels, Belgium. Researchers from all over the world discussed this new optimization method. Dorigo published a research review in Nature in 2000 that pushed the ant colony algorithm to the most influential academic journals. In terms of particle swarm optimization, since Eberhart proposed the particle swarm optimization algorithm, foreign research has mainly focused on parameter and topology improvements.

The earliest research on the ant colony algorithm in China is a paper published by Zhang Jihui of Northeastern University in "Theory and Practice of Systems Engineering." Other related research also includes the adaptive ant colony algorithm proposed by Wu Qinghong and others of Northeastern University, combining the improved algorithm of mutation and exchange. Qin Gangli and others of Tsinghua University proposed an ant colony algorithm to adjust pheromones adaptively. The stochastic disturbance ant colony algorithm for the TSP is proposed by Hao Jin and others of Chongqing University. Chen Jun of Yangzhou University proposed an adaptive ant colony algorithm based on distribution uniformity. Ma Liang of Shanghai University proposed a restricted minimum tree ant algorithm. The ant colony algorithm for convex integer programming and the ant colony algorithm for continuous function optimization are proposed by Lin Jin and others of Fuzhou University. Wang Ying and others proposed an adaptive ant colony algorithm by adaptively changing the volatilization coefficient in the ant colony algorithm. Wu Bin in the Institute of Computing Technology Chinese Academy of Sciences proposed improved ant colony algorithms such as the division of labor and cooperation model based on swarm intelligence and clustering model based on swarm intelligence and their applications in data mining. Cheng Junsheng of Anhui University proposed to use randomly connected artificial neural networks to describe the mathematical model of swarm intelligence. Li Youmei of Xi'an Jiaotong University proposed a global hybrid heuristic algorithm by combining the discrete neural network algorithm with ant colony optimization.

Earlier research on particle swarm optimization in China is a paper published by Xu Hai of Fudan University in Computer Engineering and Application. He proposed a self-learning algorithm for a fuzzy logic system based on improved particle swarm optimization. The computer simulation of fuzzy identifiers proves the effectiveness of the improved algorithm. Other related researches include Dong Ying of Northeastern University, who proposed a hybrid particle swarm optimization algorithm to solve nonlinear programming problems and designed a method to prioritize constraint fitness to deal with constraint conditions. The dynamic neighborhood operator and variable inertia weight are combined to evolve to obtain a global optimal solution. Huang Lan and others of Jilin University proposed a special particle swarm optimization algorithm by using the concepts of the swap operator and swap sequence. This algorithm is then applied to solve the traveling quotient problems. Wang Suihua and others of Henan Normal University proposed a network learning algorithm based on particle swarm optimization. Lv Zhensu and others of Lanzhou University put forward an improved algorithm of adaptive mutation particle swarm optimization, and they made numerical simulation of classical functions. Zhang Libiao and others of Jilin University proposed to solve multi-objective optimization problems based on

the particle swarm algorithm. They made corresponding improvements to the selection methods of global extrema and individual extrema. Liu Jingming and others from Shanghai Jiaotong University proposed a *K*-means clustering algorithm based on particle swarm optimization. Chen Hongzhou of Harbin Engineering University proposed a particle swarm algorithm with sensory characteristics to enhance the ability of local search. Peng Yu of Harbin Institute of Technology proposed a hybrid swarm intelligence optimization method based on the particle swarm algorithm and simulated annealing algorithm.

4.2 Development Tendency of Navigation Algorithm

With the continuous development of science and technology, the environment faced by path planning and navigation algorithms is becoming more complex and changeable. This requires the algorithm to have the ability to quickly respond to complex environmental changes. The meta-heuristic optimization algorithm, as a hot research field in intelligent science and computational science, has achieved a large number of research results in recent years and showed strong attraction and vitality. The meta-heuristic algorithm and its application research are unfolding. Human's endless exploration of natural systems and their thirst for advanced machine intelligence would surely promote the vigorous development of this field. Looking at the development history and current situation of meta-heuristic algorithms, we can briefly summarize its main development direction in the future:

1. Algorithm mathematical analysis

Despite the great success of the meta-heuristic algorithm, the mathematical analysis of the algorithm is still limited because its idea includes complex random behaviors, and there is no universal framework. Theoretical analysis on complexity, convergence, and computational capability is still not effectively solved; thus, further development of the meta-heuristic algorithm is still needed.

2. High-dimensional multi-objective optimization problem

The question of how to effectively solve the high-dimensional, multi-objective optimization problem is one of the most difficult problems in the optimization field today. In recent years, researchers have been attempting to solve the multi-objective, optimization problem by using distributed estimation algorithm, ant colony algorithm, and particle swarm optimization algorithm. However, there are still many challenges in terms of theory, system, and testability.

3. Hyper-heuristic algorithm

The hyper-heuristic algorithm provides some high-level strategy to obtain a new heuristic algorithm by operating or managing a set of low-level heuristic algorithms [28]. Although the hyper-heuristic algorithm has just been proposed, due to its more

advanced intelligent system characteristics, it could potentially become one of the mainstream research directions in the field of heuristic optimization in the future.

4. Effective combination of path planning and navigation algorithms

Any single algorithm cannot solve all practical path planning and navigation problems. Especially when it comes to the new problems of interdisciplinary research, it is difficult to study new algorithms. Complementary advantages among path planning algorithms make it possible to solve this problem.

5. The combination of environmental modeling technology and path planning algorithm

In the face of complex, continuous, dynamic environment information in two-dimensional or even three-dimensional space, what the algorithm can do is limited. The combination of excellent modeling technology and the path planning algorithm could become a new idea to solve this problem.

6. Design of multi-agent parallel path planning algorithm

With the wide application of multi-agent parallel cooperation algorithms, the path conflict problem of multi-agent collaboration and dual-mechanical arm cooperation has received increased attention. How to realize its collision-free path planning may become one of the hottest issues.

5 Application of Navigation Algorithm

Modern society has long been inseparable from unmanned systems. Unmanned aerial vehicles, unmanned ground vehicles, unmanned surface boats, unmanned underwater vehicles, and so on are all over the land, air, and sea. All countries expect to enhance the synergy of unmanned systems to better serve society. For unmanned systems, dynamic planning capability based on path planning and dynamic collision avoidance is an important embodiment of intelligence. Excellent dynamic planning algorithms can assist unmanned systems to complete their missions in the safest and most optimal way.

5.1 Application of Aviation Navigation Algorithm

Unmanned aerial vehicles are aerial vehicles which need no pilots and can be controlled autonomously. They can also perform ground and sea combat missions. Typical unmanned aerial vehicles include cruise missiles and unmanned airships. In the military field, unmanned aerial vehicles have the unique advantages of low cost, small size, and strong survivability. Unmanned aerial vehicles play an important role

in the war due to their high efficiency in detection, surveillance, accurate target capture, and strong combat capability. In the civil-use field, unmanned aerial vehicles have broad application prospects [29]. They can be used in many applications such as environmental monitoring, agricultural production, aerial photography, and aerial surveying.

The USA has an absolute leading position in unmanned aerial vehicle technology and its applications. The USA has strong technical strength in cruise missile technology and other aspects, and other countries are finding it difficult to catch the USA in route planning. The representative research results on route planning are used in the route planning of low observable aircrafts such as the X-47B, Global Hawk, Predator unmanned aerial vehicles, and missiles launched outside joint air-to-ground defense zones.

China started the research on path planning technology of unmanned aerial vehicles in the early 1980s. In the 1990s, the research on this technology developed rapidly. A variety of unmanned aerial vehicles with different performances such as "Wing Loong UAV" and "CH-4 UAV," displayed at the Zhuhai Air Show, showed the world the strength of China's unmanned aerial vehicle route planning technology.

5.2 Application of Land Navigation Algorithm

Unmanned ground vehicles have broad application prospects in intelligent transportation and assisted driving. In industry, unmanned ground vehicles can be applied to the automatic guided vehicles in logistics operation departments such as automatic warehouses, ports, docks, and workshops. They can improve cargo handling efficiency and reduce production costs. At the same time, they can also be used in environments where human beings cannot work to complete the tasks of cargo handling and equipment detection in harsh and toxic environments. Thus, they can help avoid the harm caused to human by some harmful substances and environments. In military affairs, unmanned ground vehicles can automatically drive on the battlefield according to the designer's intention. Unmanned ground vehicles can replace humans to complete patrols, reconnaissance, demining, and sampling of toxic substances. They can quickly and accurately collect relevant information, thereby improving the efficiency of military mission execution and effectively avoiding casualties of military personnel. In the development of intelligent weapon systems, unmanned ground vehicles can be used as an installation platform, which can automatically search and attack targets and improve the attack power and safety. In aerospace science research, space autonomous mobile vehicles are an important component [30]. Therefore, unmanned ground vehicles can also be used for the exploration of the outer planets. They could play an important role in promoting the exploration of the outer planets and the development and utilization of the outer planets.

The US Department of Defense has successively developed a path planning system for DEMO-I, DEMO-II, and DEMO-III series of military unmanned ground vehicles for military applications. The ASV project led by Japan's Ministry of

Transport and the SSVS project led by the Ministry of Communications and Industry both include the development of autonomous driving (path planning) systems for unmanned ground vehicles. The intelligent vehicle research team of the Federal Defense University of Munich, Germany, in cooperation with Mercedes-Benz, Germany, is committed to the research of autonomous vehicle navigation system and has been successfully applied to VaMoRs unmanned ground vehicles. China National University of Defense Technology has developed a vision-based CITAVT series unmanned ground vehicle path planning system. Tsinghua University began to research and develop the autonomous navigation system of THMR series unmanned ground vehicles with the support of the "863 Program."

5.3 Application of Sea Surface Navigation Algorithm

With the rapid development of modern science and technology, maritime intelligence transportation has become an important part of science and technology strategy. The development of maritime intelligence transportation is mainly to realize the navigation automation of ships and the intelligent management of maritime traffic. However, the research hotspot, unmanned surface vehicles, which has arisen in recent years, has attracted a large number of researchers due to its small size, fast speed, intelligence, and high degree of automation. Unmanned surface vehicles can be used not only in military areas such as mine clearance, reconnaissance, and anti-submarine warfare, but also in civilian areas such as hydrologic detection, meteorological detection, environmental monitoring, and maritime search [31]. Path planning, as the core issue of unmanned craft research, represents the intelligence level of unmanned craft to a certain extent.

The USA and Israel have always been in the lead in the research and use of unmanned surface vehicle path planning systems. The Spartan Scout and Ghost Guard, developed by the USA, have highly intelligent path planning systems, which can realize autonomous navigation, intelligent obstacle avoidance, and mission execution of unmanned boats. The Protector, Silver Marlin, and Stingray unmanned surface vehicle, launched by Israel's Ministry of Defense, have the advantages of flexibility and concealment. They are equipped with intelligent cruise and autonomous collision avoidance path planning systems. Shanghai University and Donghai Navigation Safety Administration jointly developed China's first unmanned surface vehicle "Jinghai No. 1." It consists of detecting unmanned offshore platforms and uses ground control systems to achieve remote control and autonomous navigation, route planning, route tracking, automatic sea surface and underwater obstacle avoidance collision avoidance, autonomous long-range navigation, and other functions.

5.4 Application of Underwater Navigation Algorithm

Unmanned underwater vehicles are underwater vehicles that do not need to be controlled by personnel in the submersible. They are mostly used to perform underwater operations, long-range transportation, ocean monitoring, intelligence collection, resource investigation, forecasting and early warning, scientific research, and other tasks. With the rapid development of computer technology, artificial intelligence technology, navigation equipment, and command and control hardware, unmanned underwater vehicles have been greatly developed. As the unmanned underwater vehicle has removed the mooring line, it is more flexible in underwater operations. More and more attention has been paid to this technology by the military and marine technology departments of developed countries. In the research of unmanned underwater vehicles, path planning technology is an important part to perform and complete the navigation.

The research on the path planning of unmanned underwater vehicles has lasted more than 20 years. Many coastal countries, especially developed countries, are committed to the research of navigation technology for unmanned underwater vehicles. Battlespace Preparation Autonomous Underwater Vehicle (BPAUV) is developed by the US Naval Research and Planning Agency in cooperation with the Bluefin Institute of Robotics. It uses artificial intelligence and sensors for navigation and has the capability of autonomous avoidance and route planning. The Saab Company of Sweden devotes itself to the research of navigation systems for unmanned submarines. It has developed the AUV62-MR type unmanned underwater vehicle with long-distance operation and high-order autonomous capability. A new type of unmanned underwater vehicle, Oilster, developed by the French company ECA, uses various sensors to detect surrounding targets and carry out self-protection navigation. The "TYPHLONUS" type unmanned underwater vehicle, developed by Russia, is equipped with an autonomous navigation system, mainly used for mine detection, mine hunting, searching, and detecting submerged nuclear submarines.

References

1. Baritzhack IY, Berman MN (2012) Control theoretic approach to inertial navigation systems. J Guid Control Dyn 10(10):1442–1453
2. Binitha S, Sathya S Siva (2012) A survey of bio-inspired optimization algorithms. Int J Soft Comput Eng 2(2):137–151
3. Xu J (2007) A study on the theory and applications of meta-heuristic optimization algorithms. Beijing University of Posts and Telecommunications
4. Teng Fei, Liyi Zhang (2015) Research on modern intelligent optimization algorithm. Inf Technol 10:26–29
5. Yang XS, Deb S, Fong S et al (2016) From swarm intelligence to metaheuristics: nature-inspired optimization algorithms. Computer 49(9):52–59
6. Zhang G, Nie L, Zhang L (2015) Review on biogeography-based optimization algorithm and applications. Comput Eng Appl 51(3):12–17

7. Zuo X, Su S (2014) Research and application of multi sub-population invasive weed optimization algorithm. Comput Eng 40(2):184–188
8. Yu JP, Zhou XM, Chen M (2010) Research on representative algorithms of swarm intelligence. Comput Eng Appl 46(25):1–4
9. Xin-She Y, Xing-Shi HE (2013) Swarm intelligence and smart optimization algorithms. Basic Sci J TextE Univ 26(3):287–296
10. Yang XS, Deb S, Zhao Y et al (2018) Swarm intelligence: past, present and future. Soft Comput 22(18):5923–5933
11. Bonabeau E, Dorigo M, Theraulaz G (2000) Inspiration for optimization from social insect behavior. Nature 406(6791):39–42
12. Wei Y, Qiqiang L (2004) Survey on particle swarm optimization algorithm. Eng Sci 5:87–94
13. Kennedy J, Eberhart RC (1995) Particle swarm optimization. In: Proceedings of the IEEE international conference on neural networks. IEEE Service Center, Piscataway, NJ, pp 1942–1948
14. Gao Z (2012) Study on optimization methods based on biological swarm intelligence and applications. Northeastern University
15. Yang XS (2009) Firefly algorithms for multimodal optimization. In: International conference on stochastic algorithms: foundations and applications
16. Dong J (2013) Study on firefly algorithm and its application in path planning of underwater vehicles. Harbin Engineering University
17. Ma Y, Zhao Y, Wu L et al (2015) Navigability analysis of magnetic map with projecting pursuit-based selection method by using firefly algorithm. Neurocomputing 159:288–297
18. Zhao Y, Jia R, Jin N et al (2016) A novel method of fleet deployment based on route risk evaluation. Inf Sci 372:731–744
19. Karaboga D, Akay B (2009) A comparative study of artificial bee colony algorithm. Appl Math Comput 214(1):108–132
20. Xue F (2016) Research and application of heuristic intelligent optimization based on bat algorithm. Beijing University of Technology
21. Yang XS, Deb S (2010) Cuckoo search via Lévy flights. In: World congress on nature and biologically inspired computing
22. Holland JH (1992) Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. MIT Press, Cambridge
23. Storn R, Price K (1997) Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. J Global Optim 11(4):341–359
24. Zong WG, Kim JH, Loganathan GV (2001) A new heuristic optimization algorithm: harmony search. Simulation 2(2):60–68
25. Glasius R, Komoda A, Gielen SC (1995) Neural network dynamics for path planning and obstacle avoidance. Neural Netw. 8(1):125–133
26. Volodymyr M et al (2015) Human-level control through deep reinforcement learning. Nature 518(7540):529–533
27. Cong M, Wang L (2003) Survey on the theory of meta-heuristic algorithms. Chin High Technol Lett 5:105–110
28. Cowling P, Kendall G, Soubeiga E (2001) A Hyper-heuristic approach to scheduling a sales summit. In: Practice and theory of automated timetabling Iii, third international conference, Patat, Konstanz, Germany, August, Selected Papers. DBLP
29. Hu Z (2011) Research on some key techniques of UAV path planning based on intelligent optimization algorithm. Nanjing University of Aeronautics and Astronautics
30. Sheng L, Bao L, Wu P (2018) Application of heuristic approaches in the robot path planning and optimization: a review. Electron Opt Control 9:58–64
31. Zhao Y, Li W, Shi P (2016) A real-time collision avoidance learning system for unmanned surface vessels. Neurocomputing 182:255–266

Chapter 3
Is the Vehicle Routing Problem Dead? An Overview Through Bioinspired Perspective and a Prospect of Opportunities

Eneko Osaba, Xin-She Yang and Javier Del Ser

1 Introduction

Routing problems and all their variants are intensive studies in the current optimization and operations research communities. The success of these problems resides in two different aspects: their practical nature and their challenging complexity. On the one hand, most of the routing problems arise from the necessity of efficiently dealing with a real-world situation, often related to logistics, business or urban transportation. This is the reason why their solving entails either economic or social benefits. On the other hand, routing problems are usually categorized as NP-hard ones, making the searching of optimal solutions a computational demanding task, even in medium-sized scenarios. This last fact leads the related community to the design and implementation of a wide variety of artificial intelligence approaches, with the intention of solving them in a computationally admissible fashion. This research trend is easily ascertainable just taking a quick glance to the rich literature regarding different formulations and practical applications.

In this regard, two are the problems which historically have attracted the majority of the researchers' attention: the traveling salesman problem (TSP, [1]) and the

E. Osaba (✉) · J. Del Ser
TECNALIA, Basque Research and Technology Alliance (BRTA), 48160 Derio, Spain
e-mail: eneko.osaba@tecnalia.com

J. Del Ser
e-mail: javier.delser@tecnalia.com

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

J. Del Ser
University of the Basque Country (UPV/EHU), 48013 Bilbao, Spain

X.-S. Yang and Y.-X. Zhao (eds.), *Nature-Inspired Computation in Navigation and Routing Problems*, Springer Tracts in Nature-Inspired Computing,
https://doi.org/10.1007/978-981-15-1842-3_3

vehicle routing problem (VRP, [2]). This specific study is related to the second case, the VRP, which is widely recognized for efficiently dealing with business logistic scenarios [3].

Furthermore, a myriad of optimization solvers has been proposed in the last decades for tackling with the VRP and all its variants. Specifically, three are the most recurrent ones: exact methods [4, 5], heuristics [6, 7], and metaheuristics. Arguably, the ones that have demonstrated a better performance are the later ones, especially in the last decade. This is why this study devotes a greater effort to this concrete solving schemes. In this category, we could place the simulated annealing (SA, [8]) and Tabu search (TS, [9]) as the most recognized approaches, along with ant colony optimization (ACO, [10, 11]), genetic algorithm (GA, [12, 13]), and particle swarm optimization (PSO, [14, 15]).

Besides this classical and extensively used schemes, the formulation of novel solvers remains a hot topic in the community, resulting in the proposal of successful methods which have made a remarkable impact in the literature. Some examples of such groundbreaking metaheuristics are the firefly algorithm (FA, [16]), imperialist competitive algorithm (ICA, [17]), cuckoo search (CS, [18], artificial bee colony (ABC, [19]), and bat algorithm (BA, [20]).

The adaptation and improvement of all these methods to the VRP and their variants have led to a rich and bountiful literature, composed by a plethora of high-quality studies. This extensive research keeps alive the interest in this specific routing problem, despite being firstly proposed more than half a century ago. In this line, the main contribution of this paper can be divided into three different points. First, we devote a section for describing the VRP and its most studied variants, emphasizing the features considered in the last years. Secondly, we dedicate a comprehensive section for the systematically outline the research made recently around the VRP. Hereof, we focus our attention on its tackling through the use of metaheuristic schemes. Finally, an additional contribution to this study is our personal envisioned status of the research field, presented in the form of open opportunities and challenges that should guide the research of upcoming years.

Thus, this paper is structured as follows: In Sect. 2 the VRP and some of its most important variants are described and mathematically formulated. Section 3 is committed to the systematic overview of the recent research done around the VRP. Finally, Sect. 5 concludes the paper with a general outlook for the wide audience.

2 Problem Statement

Along the history, the VRP has been studied in many forms and using many formulations. Regarding the basic version of the VRP, it can be defined as a complete graph $G = (V, A)$, where $V = \{v_0, v_1, \ldots, v_n\}$ is the set of vertexes and $A = \{(v_i, v_j) : v_i, v_j \in V, i \neq j\}$ is the set of arcs. The vertex v_0 represents the depot, while the rest are the customers that should be served. Moreover, each edge (v_i, v_j) has an associated cost $c_{ij} \in \mathbb{R}^+$, denoting the traveling weight of this arc.

Besides that, an additional set $Q = \{q_1, q_2, \ldots, q_n\}$ is deemed, depicting the demand of each client. Specifically, this means that q_i is the demand associated to v_i. Furthermore, a fleet of vehicles K is available with a limited capacity represented as C. In some instances of the VRP, this fleet of vehicles is composed of a fixed number of units. Anyway, in the majority of problem variants, this number is unlimited, being its minimization one of the main objectives to optimize.

Hence, the principal objective of this basic version of the VRP is to find a number of routes with a minimum cost such that (1) each route starts and ends at v_0, (2) each customer is visited exactly once, and (3) the summation of the demands served in each of the routes does not exceed the C [21]. This problem could be formulated as follows [5]:

$$Minimize: \ f(X) = \sum_{i=0} \sum_{i \neq j, j=0} d_{ij} x_{ij} \quad \forall i, j \in V \tag{1}$$

$$Subject\ to\ constraints: \sum_{i=0, i \neq j} x_{ij} = 1, \quad \forall j \tag{2}$$

$$\sum_{j=0, i \neq j} x_{ij} = 1, \quad \forall i \in V \tag{3}$$

$$\sum_{i} x_{ij} \geq |S| - v(S), \quad \{S : S \subseteq V/\{1\}, |S| \geq 2\} \tag{4}$$

$$\sum_{i \in S} q_i y_i^r \leq = C, \quad \forall r \in K \tag{5}$$

$$Where: y_i^r \in \{0, 1\}, \quad \forall r \in K \tag{6}$$

$$And: x_{ij} \in \{0, 1\}, \quad \forall \{i, j\} \in A; \ i \neq j \tag{7}$$

First clause (1) is devoted to the objective function, which depicts the total distance traveled by all the vehicles. The variable (6) is a binary parameter which takes a value equal to 1 if vehicle r satisfies the demand of the client i, and 0 otherwise. Moreover, the binary variable (7) takes 1 if the arc (i, j) is used in the solution. Restrictions (3) and (4) ensure that every client is visited by one and only one vehicle, and exactly once. Lastly, formula (4) serves to eliminate sub-tours, where $|S|$ is the number of customers and $r(S)$ the minimum number of routes to attend all. Finally, clause (5) guarantees that the sum of all the demands of a route does not exceed the maximum capacity C of the vehicle.

The formulation depicted up to now represents the VRP variant widely considered as canonical. Besides that, many different additions or modifications have been conducted over this formulation with the aim of addressing more complex transportation and logistics situations and conditions. Probably, the most studied variant is the so-called VRP with time windows (VRPTW, [22]). In the VRPTW, in addition to the

basic constraints above mentioned, each customer has an associated time window $[e_i,l_i]$. This window is comprised of a lower limit e_i and an upper limit l_i, which must be respected. In other words, the service in every customer must be performed after their associated e_i and before l_i. Additionally, a time window could be also considered for the depot, which restricts the period of the whole activity.

One of the principal reasons why VRPTW is so interesting for the researchers is its dual nature, being deemed as a two-phase problem. The first step regards the assignment of vehicles to customers, while the second one is related to the planning phase or customer scheduling. As has been said, the VRPTW has been extensively studied either in past decades [23, 24] and nowadays [25, 26]. Additionally, we recommend [27] to readers interested in the mathematical formulation of this variant. Finally, is it noteworthy that a specific variant of the VRPTW can be found in the literature, which enables the non-compliance of some time windows (under penalization in the objective function). This variant is called VRP with soft time windows [28].

Other important variant is the one related with heterogeneous fleet of vehicles. In the basic version of the VRP, all considered vehicles have the same characteristics, something considered a limitation for many real-world applications. The so-called heterogeneous VRP (HVRP) tackles this issue by deeming different road units, each one with diverse capacities and features. As can be seen in the remarkable survey study [29], many sub-variants exist within this specific topic, each one depending on the concrete characteristics of the vehicles. Thus, in Table 1, the most often used formulations are summarized.

As additional constraint often considered by authors is the consideration of not only delivery of goods to customers, but also the pickup of certain materials. This feature is especially valuable for waste management and for implementing recycling policies. As above mentioned for the HVRP, several sub-variants have arose depending on the specific characteristics of the scenario. These are some of the most treated ones:

Table 1 Summary of HVRP variants

Variant name	Size of the fleet	Costs of use	Routing costs
HVRP with fixed cost and vehicle-dependent routing costs	Limited	Dependant	Considered
HVRP with vehicle-dependent routing cost	Limited	Not considered	Dependant
Fleet size and mix VRP with fixed costs and vehicle-dependent routing costs	Unlimited	Considered	Dependant
Fleet size and mix VRP with vehicle-dependent routing costs	Unlimited	Not considered	Dependant
Fleet size and mix VRP with fixed costs	Unlimited	Considered	Non-dependant

- *VRP with Simultaneous Pickup and Delivery*: In addition to the usual demand, customers can also ask for the pickup of materials. Vehicles conduct simultaneously pickup and delivery each time they visit a client [30].
- *VRP with Pickup and Delivery*: In this variant, customer can only ask for pickup or delivery, but not both simultaneously. Furthermore, the route can alternate clients of different kind along the route [31].
- *VRP with Backhauls*: Similar to the previous one, with the distinction that vehicles should serve all clients asking for the delivery of good and then the one asking for pickups [32].

We list here further restrictions and conditions which have inspired additional formulations of advanced versions of the VRP:

- *Split deliveries*: In this case, deliveries can be served by more than one vehicle seeking for cost reductions.
- *Time-dependent travel costs*: The principal idea behind this feature is that the cost of traveling between two clients depends on the moment the trip is made. Thus, travel times could significantly change over the time schedule.
- *Open paths*: In these scenarios, routes are not obligated to finish in the starting depot.
- *Asymmetric travel costs*: In these variants, although there may be arcs where $c_{ij} = c_{ji}$, in general $c_{ij} \neq c_{ji}$.
- *Wide periods*: This characteristic is appropriate in scenarios where the deliveries should be made in a time horizon greater than one day. Thus, different plans are built for each considered day.
- *Multiple depots*: In these environments, multiple depots are spread over the scenario which could be used by the vehicles for constructing routes that optimize the distance traveled.
- *Dynamism*: The main philosophy behind this feature is to contemplate the uncertainty nature of real-world situations. In this variant, some of the aspects of the problem are unknown at the beginning of the planning procedure. Furthermore, elements such as the demands or the geographic situation of customers could be also dynamically modified.
- *Satellite Facilities*: Several satellite depots are available for road units as recharging points. These facilities could be used to take some materials, deposit recovered waste or even charge electric vehicles.

In addition, an interesting research activity in this field gravitates around the so-called rich or multi-attribute VRPs (RVRP). Currently, as can be read in several works [33, 34], these complex problems are catching the attention of the scientific community. As we can be found in these surveys, RVRPs are special cases of the conventional VRP characterized for having multiple restrictions and variables, and a highly complex formulation. This increase in the problem complexity inherently entails a major scientific interest, linked to the challenge that supposes their solving. Furthermore, because of the high amount of conditions, these variants are especially important for dealing with real-world logistic and transportation situations.

As can be seen, the number of VRP variants existing in the literature is overwhelming, making impossible the listing and consideration of all the valuable formulations in this sole section. We have outlined here some of the most frequently recurred ones, with the goal to establish in the reader the idea that there is a vibrant scientific activity behind this problem and its multiple forms.

Furthermore, one direct consequence of the existence of such a high number of VRP variants is the generation and proposal of multiple valuable benchmarks, used by the community for solving these VRP instances through diverse algorithmic approaches. Undoubtedly, the most well-known benchmark of this kind is the one proposed by Solomon in 1987 for the VRPTW [22]. Moreover, this influential benchmark has served on many occasions as the base for the generation of new ones, such as the ones created by Osaba et al. for the VRPB [35]. In addition to the Solomon's one, several often employed benchmarks can be found also for the VRPTW. Four different remarkable sets of instances can be highlighted in this regard: Benchmarks of Cordeau [36], Breedam [37], Homberger [38], and Russell [39]. Related to the CVRP, many interesting benchmarks have been proposed along the years, being the one generated by Christofides and Eilon in [40] the most frequently used one for testing purposes. An additional set of cases for this specific variant are the ones developed by Fisher [41] or the proposed by Golden et al. in [42]. Further interesting benchmarks can be found in the literature for additional VRP variants, such as the ones introduced by Gillet and Johnson [43] for the Multiple Depot VRP, or the generated by Cordeau et al. for the Periodic VRP [44]. For interested readers seeking for the obtaining of valuable set of instances, we recommend some of the multiple Web platforms dedicated to the VRP, its variants and benchmarks, such as VRP Web[1] managed by the University of Malaga, or the Web dedicated to Solomon's instances.[2]

3 Recent Advances in Vehicle Routing Problem

Since its first formulation, the VRP had rapidly turned into one of the most studied problems, both for solving real-world transportation problems and for benchmarking purposes in performance analysis of discrete optimization algorithms. Thanks to this protracted activity around the VRP, a deluge of solvers has been used for dealing with this problem and its variants in the last decades. Because of being classical metaheuristics, GA [45, 46], SA [47, 48], and TS [44, 49] are three of the most used alternatives, producing valuable scientific material since their inception. Also, the hybridization of these methods have also been explored in the literature in a remarkable way [50–53]. Advancing a step in the history of optimization, more recent algorithms which today are also deemed as classical have also been broadly employed for solving the VRP. We can place in this category schemes such as the PSO [54, 55], ACO [10, 56] or the variable neighborhood search (VNS) [57–59].

[1] http://neo.lcc.uma.es/vrp/.

[2] http://w.cba.neu.edu/~msolomon/problems.htm.

Finally, the VRP and its multiple formulations have been also object of study from the perspective of recently proposed nature-inspired approaches, mostly devoted to measure the quality of these newly proposed metaheuristics. Some of these solvers are the bat algorithm [60], cuckoo Search [61], artificial bee colony [62], or the flower pollination algorithm [63].

This brief bibliographic summary helps us to support the theory that the VRP and its derivations have been extensively addressed by operation research and computational intelligence communities from different mindsets and viewpoints, and using many different algorithmic proposal. The research over the VRP is such wide that we will sharpen our attention on highlighting the advances and research carried out in the last few years by bioinspired algorithms. Of course, we are completely aware that the related literature is much bigger than the one outlined in this systematic review. Thus, we refer interested readers to surveys such as [4, 64–66].

3.1 Vehicle Routing Problem and Genetic Algorithms

Following the same trend as other optimization problems, such as the TSP [67] or the job-shop scheduling problem [68, 69], genetic algorithm is, arguably, the most frequently used metaheuristic for solving VRP-related problems. Thanks to its easy adaptation and efficiency, a surfeit of GA models has been proposed since the first formulation of this algorithmic scheme. On this regard, a notable amount of valuable studies has been published in the last years. In the recent paper [70], for example, Ruiz et al. developed a biased random key GA for solving an open VRP with capacity and distance constraints. The main feature of the modeled problem is that the vehicles finish their corresponding routes after servicing the last client, without the need of returning to the depot. Furthermore, vehicles cannot exceed the trunk capacity and the previously fixed maximum distance. In [71], a heterogeneous VRP with fixed cost and green reverse logistics network is tackled using a GA. A valuable rich variant of the VRP is handled in [72] by Baniamerian et al. In that work, profitable heterogeneous VRP with cross-docking is presented, in which the main objective is to increase the total profit of a cross-docking system, and it is solved through the application of a hybrid method based on a VNS and a GA. Results obtained by this advanced approach are compared with the those obtained with an ABC and a SA. Another hybrid scheme is proposed in [73] using as base a GA and a local search mechanism. In that case, the VRP variant to solve is coined as *prize-collecting*, and it has the particularity of adding a customer selection process. This process is determined because of the unavailability of the fleet of vehicles for visiting all the clients. Besides that, an influential hybrid GA is implemented in [74], designed for tackling a group of instances of the heterogeneous fleet VRP with time windows. Another outstanding work, due to its application to the real-world, is presented in [75]. In that work, a VRP is used for modeling the problem of student transportation to the Universiti Tenaga Nasional (in Malaysia), employing a fleet composed by eight different buses. The designed VRP is addressed by a GA also in that work.

The success of hybrid methods using a GA in combination with other methods or mechanisms is also testified in [76]. In that paper, a local search mechanism is used for improving the performance of a fitness-scaling adaptive GA. The VRP instance dealt, in that case, is the well-known multiple depot VRP. As can be checked, the literature around the GA and VRP is abundant nowadays, being the subject of myriad of works year by year. Interested readers are referred to additional interesting studies such as [77, 78] or [79].

3.2 Vehicle Routing Problem and Tabu Search

Along the last decades, TS has been successfully applied to a wide range of problems, raising it to a privileged status in the combinatorial optimization scientific community. In any case, as time goes by, sophisticated methods have eclipsed some classical approaches such as the TS in some crucial problems. Fortunately, this is not the case of the VRP, for which TS still enjoys great popularity, being the focus of many researches around its many variations. Some valuable examples can be found in [80, 81]. In the former, a heterogenous fleet is considered as the main feature of the VRP, while in the latter case, discrete split deliveries and pickups are deemed as principal restriction. In both cases, the TS is implemented solely, and the development of advance memory structures are the key of their successful application. In [82], Sicilia et al. presented a TS in combination with a VNS for tackling a rich VRP. The formulation of the problem modeled in that work, which has been conceived for capillary distribution of goods in major urban areas, counts with six different features: capacity constraints, time windows, pick and delivery, workload limitation, compatibility between orders, and open paths. Another interesting study is presented in [83] by Silvestrin and Ritt, in which a TS is developed for addressing a multi-compartment VRP. This characteristic is especially efficient to deal with scenarios in which several products of different types are available that must be handled separately. Also, valuable are the contributions proposed in [84, 85], in which two green variants of the VRP are tackled. The main objective of these works is to build routes which maximize the profit of the user while pollutant emissions and fuel consumption are minimized. Additional valuable related research can be found in papers such as [86–88].

3.3 Vehicle Routing Problem and Simulated Annealing

As in the case of the TS, SA is still the focus of many scientific activities dedicated to the VRP family of problems. In line with the last works described in the previous sections, in the last years a remarkable amount of relevant works have gravitated around the concept of green logistics. In this sense, SA has emerged as a successful alternative for dealing with this specific problem. This statement can be easily

verified analyzing works such as [89–93]. All these works proposed different implementations of the SA, using diverse movement strategies, codifications, and cooling schemes. In any case, all of them share the same objective, which is the design of paths which minimize the environmental footprint. Furthermore, in [94] four different variants of the VRP with two-dimensional loading constraints are proposed. For solving those problems, authors propose the use of a SA with a mechanism of repeatedly temperature cooling and rising. In [95], a SA is presented for solving the open VRP, in which vehicles do not return to the distribution center after attending all clients. In [96], Mahmudy proposed an interesting improvement of the basic SA for solving the well-known VRPTW. The same restriction is considered in [97], adding the particularity of considering pickup and deliveries in its formulation. For dealing with such variant, authors of that study developed a parallel SA with local search, in which list of temperatures and cooling schemes are used. Time windows are also considered in [98], in which a group of metaheuristics are designed and applied to solve a VRPTW with synchronized visits. Finally, further valuable papers using SA can be found in [99–101].

3.4 Vehicle Routing Problem and Particle Swarm Optimization

Thanks to its fast execution time, convergence, and efficiency, the PSO has become the most frequently employed swarm intelligence method. Likewise, PSO is one of the most influential methods in bioinspired computation, serving in this way as inspiration for the development of additional cornerstone approaches. Very briefly explained, PSO was proposed in 1995 by Eberhart and Kennedy using as motivation the behavior of bird flocks, fish schools, and human communities. As many other swarm intelligence algorithms, the PSO was firstly developed for solving continuous problems. Nevertheless, many different discrete adaptations were introduced by the community after conducting few ad hoc modifications in its main scheme. Regarding the VRP, a bunch of high-quality papers have been published in the last decades. Some worth-citing ones are [54, 55, 102]. This interest on the PSO has not only be maintained but increased in the last years, leading to the conduction of interesting studies such as [103, 104]. The first of these works is devoted to the resolution of a business logistic problem, contextualized in a carton distribution industry with recycling policy. For properly facing the modeled rich VRP, authors implement an improved PSO, endowed with a self-adaptive inertia weight and a local search strategy. The second of the works above mentioned deals with a production and pollution VRP with time windows. That work is especially valuable due to two reasons. On the one hand, authors treat the problem through the multi-objective perspective. On the other hand, the PSO developed is improved using a self-learning mechanism. In line with this, the work presented in [105] also addresses the problem using a multi-objective formulation. In that case, a dynamic VRP is solved using a PSO, using time

seed-based solutions for dealing with the dynamism of the environment. Moreover, authors contemplate four objectives to optimize, namely, number of vehicles, expected reachability time, satisfaction level, and economical profit. Another interesting dynamic VRP is tackled in [106], using as solving approach a PSO-based hyper-heuristic. In addition, in the recent work published in [107], a neural-like encoding PSO for solving an industrial-oriented periodic VRP. Further, recently published works are [108, 109] or [110].

3.5 Vehicle Routing Problem and Artificial Bee Colony

Since its first formulation by Karaboga and Basturk, the artificial bee colony (ABC, [19]) has also been adapted for solving different discrete problems such as the TSP or VRP. Very recent is the work proposed in [111], in which a rich variant of the VRP coined as waste collection problem. The main objective of this problem is to build effective waste collection routes for different vehicles. Furthermore, the variant modeled by the authors contemplates also the so-called midway disposal pattern, which is used for dumping the wastes dynamically in some of the disposal facilities spread over the scenario. Another interesting research is presented in [112] for tackling the well-known CVRP. The method implemented in that paper is an enhanced version of the ABC, in which a crossover mechanism is deemed for guiding the individual updating. A valuable experimentation is conducted in [113], carrying out a comparison between an ABC and a CS for the basic VRP. The main objective of that paper is to assess the performance of some important discrete mechanisms for properly dealing with this problem. In any case, among this specific stream, we can find as most influential work the one proposed by Ng et al. in [114], in which a multi-populational version of the ABC is proposed for dealing with a CVRP. The main feature of this problem is the re-routing strategies implemented for properly dealing with the time-dependent traffic congestion. This last characteristic endows the research with a very valuable practical adaptability. Two additional practical studies related to the VRP and ABC can be found in [115, 116]. The first of these papers deals with a real western-style food delivery problem in Dalian city, China, modeling the problem as a VRPTW. On the other hand, the second one proposes a hybrid method combining an ABC and an advanced adaptive memory mechanism for the resolution of a green VRP with cross-docking feature. Further remarkable studies can be found in [117, 118] or [119].

3.6 Vehicle Routing Problem and Ant Colony Optimization

Ant colony optimization arguably belongs to the group of most intensively used and well-reputed metaheuristics. Since its inception, ACO has shown an outstanding performance being applied to a wide variety of problems, being the VRP one of these

cases. In the last years, many prominent papers have seen the light, highly contributing to the non-stopping advance of the field. In [120], an influential work is proposed, in which an ACO with clustering capabilities and three immigrant schemes is proposed for solving a VRP variant widely known as location routing problem. Authors went a step forward with this problem endowing it with dynamic conditions. Another dynamic VRP is deemed in [121], in which the demands can appear and disappear dynamically, as well as the geographical situation of unserved clients. The technique developed in that work is an ACO enhanced with two different mechanisms: an improved K-means algorithm and crossover operations. Especially interesting is the research that can be found in [122], in which Wang et al. develop a benchmark of ACO variants for efficiently simplifying the construction of solutions in VRP-type problems. A hybrid scheme is proposed in [123] combining an ACO with a VNS for solving the VRP with simultaneous pickups and deliveries. In [124], a valuable multi-objective multi-depot green VRP is addressed. In that work, an improved ACO is developed for optimizing four different objectives: revenue, costs, time, and emission. The main novelty of the implemented ACO stems from an innovative approach in updating the pheromones. Diverse variants of the green variants of the VRP are also tackled by advanced versions of the ACO is works such as [125, 126]. A worth-citing real-world research can be seen in [127]. In that work, Yao et al. face the problem of fresh seafood delivery routing problem. The case study of that is placed in Dalian city, China. To properly solve that problem, authors modeled an ACO improved by the adoption of scanning strategies and crossover operations. For readers interested in additional interesting works focused on the ACO studies such as [128–130] are highly recommended.

3.7 Vehicle Routing Problem and Cuckoo Search

Focusing our attention in recently proposed nature-inspired methods, cuckoo Search is a method introduced by Yang and Deb [18] a decade ago, inspired by the brood parasitism of some cuckoo species. In addition, rather than using simple isotropic random walks, CS is enhanced by Levy flights [131], which is one of the keys to its success. Along this decade, CS has been applied in many knowledge fields, being transportation and logistics one of them. This is the reason why many interesting studies related to CS and VRP can be found in the literature. In [132], for example, Teymourian et al. develop a group composed by four metaheuristics for dealing with the CVRP. One of these methods is the CS, which is not only adapted but enhanced with some mechanisms for improving its performance. Some of these enhancements are the dynamic adaptation of parameters, or the introduction of a new group of cuckoos in charge of performing smart local searches. The same problem is considered in the research published in [133], in which Levy flights are conducted based on the well-known 2-opt and double-bridge operations. Additional worth-mentioning works around the CVRP can be found in [134, 135]. These studies introduce some additional novelties, such as the use of the Taguchi-based parameter

setting mechanism. An interesting hybrid technique is developed by Chen et al. in [136] for solving the basic VRP which combines the advantages of a CS, PSO, and optical optimization. Another hybrid scheme is proposed in [61], in this case, exploiting the synergies between the CS and the greedy randomized adaptive search procedure. Furthermore, in [137], a real-world-oriented problem is modeled for the patient transportation problem. The problem is formulated using as inspiration the CVRP, and it is solved using an improved version of the CS, in which a new category of cuckoos is introduced to improve the search capacity of the approach.

3.8 Vehicle Routing Problem and Imperialist Competitive Algorithm

ICA is a multi-population technique which was conceived in 2007 by Atashpaz-Gargari and Lucas. Main inspiration of ICA rests on the concept of imperialism, dividing first the complete population of solutions in independent empires, which fight to each other along the execution with the aim of conquering the weakest colonies of the rest of the empires [17]. This method has also been applied in many occasions to the VRP, being this adaptation to the focused scope of some remarkable works. We can find the first adaptation of the ICA to VRP problems in the work proposed by Wang et al. in [138]. In that work, the VRPTW is used as benchmarking problem. In [139], advanced hybrid method is proposed for the multi-source multi-product location routing inventory problem, which is rich variant of the VRP. The approach designed for solving this challenging problem explores the synergies between an ICA and the SA, using as key concept the embedding of the SA acceptance criterion into the ICA colony movement scheme. An additional remarkable work can be found in [140] for solving the often treated open VRP. Main crucial factors of the adaptation proposed in that research are the permutation encoding adopted and the use of movement operators such as swapping and insertion. Furthermore, a notable study has been recently published in [141], presenting a VRP in cross-docking networks with time windows and solving it through the multi-objective viewpoint. In that case, the main objectives to optimize are the total transportation cost and the total earliness and tardiness of visiting retailers. For efficiently tackling this complex instance of the VRP, a multi-objective ICA is developed and compared with the well-known non-dominated sorting genetic algorithm and Pareto archived evolution strategy. Results shown in that research certify the ICA is a promising approach for dealing with this problem. Interested readers on this algorithmic scheme are referred to the following works: [142, 143].

3.9 Vehicle Routing Problem and Bat Algorithm

Bat algorithm is a recently proposed method which has gained a lot of attention from the related community thanks to its fast executions times and good results. Several outstanding works have been recently proposed which gravitate in the adaptation of this successful method to different VRP variants. The first reported discretization to this family of routing problems can be found in [144] to solve the CVRP. Four different local search operators were used by this first solver: two single-point swapping, insertion, inversion, and single-point swapping with next point local search. All these operators are randomly chosen each time a bat perform a movement in the solution space. Concerning the experimentation, a benchmark composed of nine different datasets is employed in this preliminary research, comparing the obtained outcomes with the ones produced by a GA. The same problem is deeper dealt by Zhou et al. in [60]. In that paper, a hybrid BA is implemented based on the framework of the continuous BA. In order to augment the exploitation ability of the solver, the greedy randomized adaptive search procedure and path relinking are effectively integrated. Moreover, aimed at improving the performance of this hybrid metaheuristic, the random subsequences and single-point local search are also operated with certain probability. Cai et al. also deal with the CVRP in their recent work [145]. In that work, authors proposed a BA with new parameters and operations, which uses penalty functions for dealing with the constraint conditions. Additionally, a chaotic initialization mechanism is also used, based on local search strategies. Moreover, an interesting version of the BA is proposed in [146] for the VRP with Time Windows. One of the novelties of this work is the adoption of the well-known regret-2 heuristic as movement function. Furthermore, in order to enhance the exploration capacity, the destroy-and-repair paradigm of the large neighborhood search is embedded to the method. Same VRP variant is handled in more recent [147], in which novel random reinsertion operators for the BA are derived and tested to reach better results than naive BA schemes. The experimentation carried out on this study explored the outcomes obtained over the whole 56 datasets that composed the renowned 100 customers Solomons benchmark [22]. Finally, an improved version of the BA is presented in [148] for solving a real-world medical goods distribution problem with pharmacological waste collection. This high-complexity logistic situation is modeled as a RVRP with a large number of restrictions. Some of the most important aspects of this method are the use of the Hamming distance for measuring the bats' difference, and the adoption of the neighborhood adaption mechanisms also employed previous works such as [149, 150]. For the experimentation, the proposed approach is tested over 24 different real dataset and compared with three additional nature-inspired methods. The BA developed on that research emerges as a promising alternative for handling the designed RVRP.

3.10 Vehicle Routing Problem and Firefly Algorithm

Several interesting works focused on the FA have been also published in the recent years, proving that this method is an excellent alternative also in this context. The study published in [151], for example, proposes a hybrid bioinspired method namely HAFA. This solver combines the schemes of a FA and an ACO. The rationale behind this hybridization is the following one: on the one hand, the ACO provides the method the basic framework, while the FA is in charge of exploring the uncharted regions of the solution space. Besides that, the pheromone shaking process inherent to the ACO is used to assist the escaping from local optimum. The problems chosen for testing the implemented algorithm are the CVRP and the VRPTW, showing a great performance in comparison with other seven competitive heterogeneous approaches. The same VRPTW problem is also dealt in [152], proposing a similar FA method for its solving. The research issued in [153] also in 2018 presents a discrete FA for the heterogeneous fleet VRP. The cornerstones of that method are the adoption of the 2-opt as successor function and the firefly codification, which transforms the integer-based individuals into real-valued ones. An additional valuable research can be discovered in [154], in which a hybrid FA coined as Gaussian firefly algorithm is developed for solving a real-world rich VRP. The case study presented in that work is focused on a distribution company, established in Esfahan, Iran, and it has as main objective the reduction of fuel consumption. The problem modeled for the planned situation is a time-dependent VRP with multi-alternative graph. Finally, in [155], the performance of a discrete FA was compared with other bioinspired solvers for the high-complexity VRP problem with pickup and delivery deadlines, selectivity nodes, and multiple concurrent vehicles. An interesting point of that research is that the quality of the built routes is defined by the Pareto trade-off between a measure of fairness in the share of the revenues of the transport company and the profit gained by the delivery of goods along the routes. In [156], a noteworthy work is proposed in which a door-to-door service for farm machinery maintenance is modeled as a rich VRP. The specific problem designed is an asymmetric multi-depot vehicle routing problem, which authors solve through the adaptation of a FA with compound neighborhoods. Additional interesting examples of the FA applied to VRP variants can be found in [157, 158] or [159].

3.11 Vehicle Routing Problem and Other Nature-Inspired Metaheuristics

As has been deeply discussed in remarkable literature works such as [160], nature-inspired community experienced a great impact after the proposal of PSO and ACO metaheuristics. These groundbreaking methods decisively influenced in the creation of an abundant amount of techniques which inherits their philosophy and some of their mechanisms. All these newly created schemes were generated after the scru-

tinize of natural and biological processes by researchers and practitioners. Thus, in the same way that the ACO was influenced by the behavior of the ants, or the PSO by that of the birds, additional phenomena have been considered in last decades such as (1) social and political behaviors as hierarchical societies, (2) physical processes such as gravitational dynamics, the natural cycle of the water or the electromagnetic theory, or (3) the behavioral patterns of animals such as spiders or monkeys.

Having said this, the survey outlined along this section has gravitated toward some of the most well-reputed and recognized algorithmic schemes. Anyhow, we are aware that the literature is comprised of a wider number of additional successful methods. These additional techniques could be equally effective than the ones that have guided this section, but they are definitively less studied by combinatorial optimization community. Because of the almost unattainable nature that would suppose the labor of considering all the methods deemed along the history, we have focused our attention on the ten metaheuristics above described.

Seeking the completeness of this study and with the aim of partially satisfying possible readers who may logically think that certain method should deserve a mention, we show in Table 2 a summary of additional metaheuristics that have been adapted in recent years for solving any instance of the VRP. In this table, we not only depict the name of the method and some related works, but also the main inspiration of each solver.

Table 2 Summary of additional nature-inspired methods and their application to the VRP

Name	Inspiration	References
Flower pollination algorithm [161]	Pollination process of flowers	[63, 162]
Fireworks algorithm [163]	Fireworks explosion and location of sparks	[164–166]
Symbiotic organisms search [167]	Interaction of organisms to survive and propagate	[168, 169]
Harmony search [170]	Mimicking the improvisation of music players	[171–174]
Honey-bees mating optimization [175]	Honey-bees mating process	[176–178]
Golden ball metaheuristic [35]	Teams and players organization in soccer world	[179–184]
Brain storm optimization [185]	Human brainstorming process	[186–189]
Glowworm swarm optimization [190]	Luciferin-induced glow of a glowworm	[191–194]
Whale optimization algorithm [195]	Social behavior of humpback whales	[196–198]
Bacterial foraging optimization [199]	Social foraging behavior of *Escherichia coli*	[200–202]
Gravitational search algorithm [203]	Law of gravity and mass interactions	[204, 205]
Pigeon-inspired optimization [206]	Homing characteristics of pigeons	[207, 208]
Penguins search optimization [209]	Collaborative hunting strategy of penguins	[210]
Elephants herding optimization [211]	Natural herding behavior of elephants groups	[189]
Monarch butterfly optimization [212]	Migration of monarch butterflies	[213]

4 Challenges and Research Opportunities

As can be easily evidenced in the overview made in the previous Sect. 3, the VRP and all its countless variants still attract a the lot of attention from the related practitioners. At this point, and thanks to the current state of computation and the resources available in hands of researchers, it is time for the community to take a step forward and tackle new challenges. These open challenges directly lead to the existence of research opportunities, which can be contextualized along diverse axis. Accordingly, we outline in this section some of the most important research directions that should guide the research made in the upcoming years.

- To being with, the rapid advance of the technology has increased the amount and quality of the resources available by the researches for dealing with VRP instances. In this sense, this is an appropriate landscape for practitioners for considering the dealing of large size VRP instances. The great majority of the experiments conducted in current literature deal with controlled problem datasets of small–medium size (in terms of number of nodes). Novertheless, problems arisen in the real world are usually prone to have a higher magnitude or even features that evolve along short periods of time. These facts suppose a challenge for the well-established solving approaches. Indeed, large-scale instances not only jeopardize the efficiency of many of the conventional metaheuristic methods, but they also compromise the convergence of these methods. For tackling with this specific situation, the deeming of techniques framed in the referred as *large-scale global optimization* as emerged as a promising alternative. The application of methods such as the multiple offspring sampling [214] or SHADEILS [215] can unchain unprecedented benefits for this field.
- Related to the previous challenge, and has been cited, the good health that computing enjoys today ease the consideration of alternative solving philosophies. Thus, we highly encourage the related researchers to consider alternative solving strategies for the seeking of computational efficiency and the addressing of demanding VRP instances and variants. Two of these strategies could be the cooperative co-evolutionary algorithms [216] and self-adaptive solvers [217]. An additional research stream that has demonstrated to be promising for solving similar problems is the so-called transfer optimization [218]. One interesting characteristic of VRP datasets is that, in many contexts, they share some crucial structures and knowledge. For this reason, approaches such as evolutionary multitask optimization [219] and multifactorial evolution [220] should be considered as promising research streams. Another interesting knowledge stream that should be deemed by practitioners is the so-called federated optimization [221]. In these systems, different agents can share their logistics information, schedules, and plans to other actors or similar service providers with the objective of obtaining different benefits, such as the use of shared services, the allocation of transport units, or the hiring of third-party resources.
- Another important point that should be mentioned in this section is related to the great amount of method that can be found in the literature. Although the existence

of a plethora of well-reputed and contrasted metaheuristics, a remarkable excerpt of the community continues analyzing the natural world aiming at finding new biological phenomena for mimicking. Even today, it is easy to discover recent works in the literature proposing different solvers based on yet unexplored metaphors. Some examples are the works [222–224], which introduce methods inspired by the game of tug of war, food search and mating behavior of butterflies, and the rhino herd behavior, respectively. The continuous elaboration of these kind of novel methods contributes to the crowding of an already overcrowded literature, and adding metaheuristics not only offers a clear benefit for the community, but also augments the skepticism of critical researchers. This same problem was extensively discussed in recent studies such as [160, 225], which critically doubt about the need of novel methods apparently similar to already published ones. In this line, through this paper we call for a profound reflection around the challenge of urgently standstill the elaboration of additional methods. In this regard, the whole community should work in the same direction, trying to adapt the existing well-known and sophisticated solver to more complex formulations of the VRP. An additional research trend is related to the exploration of fruitful synergies between different existing mechanisms and approaches.

- Finally, the VRP has been used extensively in the last decades as a benchmarking problem, mainly because in it most basic version is still conceived as inadequate for being applied to real-world problems. Anyway, the VRP has a greater potential to be adapted to real situations in comparison to other combinatorial optimization problems such as the TSP. This is so thanks to the three features that comprise of its formulation (depot, clients, and a fleet of vehicles), which properly adapted can deal with complex restrictions and conditions. This specific trend has explored in the literature, grasping some attention from practitioners [226]. This activity has led to the adoption of a concept named as *multi-attribute* or *rich* VRP. As has been described, these instances of the VRP are attracting some activity due to their closer match to realistic situations. Notwithstanding, the research dedicated to these advanced variants of the VRP is still scarce in comparison to the studied using it as benchmarking problems. For this reason, we want to highlight in this study the necessity of modeling new complex formulations of the VRP, which can directly lead to the efficient addressing of complex transportation and logistic situations. Another interesting trend in this direction is the consideration of multiple objectives. Researchers should find heterogeneous and valuable objectives, such as fairness, distance, cost, or emissions for the formulation of demanding many-objective VRP instances. Currently, several libraries can be found in the literature which facilitate the solving of this complex VRP cases. Two examples are the well-known JMetal [227] or JMetalPy [228].

5 Conclusions

This research has gravitated on the well-known vehicle routing problem and its multiple variants. First, we have briefly introduced this famous problem and its basic formulation, accompanied with the description of some of its most studied and practical variant. Then, we have dedicated a whole section to systematically describe the current state and recent history of this problem, highlighting some of the most remarkable studies published in the last years. We have put our attention on both classical (SA, TS, GA …) and sophisticated advanced (BA, ICA, FA) solvers. Finally, we have concluded this manuscript by sharing our envisioned future of the related community. In this line, we have pinpointed several inspiring open opportunities and their inherent research challenges, which should gather most of the activities that will be carried out in the upcoming years. Among these future lines, we advocate the using of alternative solvers not yet deeply studied, the addressing of bigger and more applicable datasets, or the exploration of profitable synergies between the solvers proposed by the related experts along last decades.

As main conclusion, and using as irrefutable proof what has been described throughout this survey, it is prudent to affirm that the VRP community has an exciting and prolific future. For this reason, the road that this community has ahead is still far to coming to an end, being more alive and vibrant than ever.

Acknowledgements Eneko Osaba and Javier Del Ser would like to thank the Basque Government for its funding support through the EMAITEK program.

References

1. Lawler EL, Lenstra JK, Kan AR, Shmoys DB (1985) The traveling salesman problem: a guided tour of combinatorial optimization, vol 3. Wiley, New York
2. Christofides N (1976) The vehicle routing problem. RAIRO-Oper Res-Rech Opérationnelle 10(V1):55–70
3. Braekers K, Ramaekers K, Van Nieuwenhuyse I (2016) The vehicle routing problem: state of the art classification and review. Comput Ind Eng 99:300–313
4. Laporte G (1992) The traveling salesman problem: an overview of exact and approximate algorithms. Eur J Oper Res 59(2):231–247
5. Laporte G (1992) The vehicle routing problem: an overview of exact and approximate algorithms. Eur J Oper Res 59(3):345–358
6. Vaghela KN, Tanna PJ, Lathigara AM (2018) Job scheduling heuristics and simulation tools in cloud computing environment: a survey. Int J Adv Netw Appl 10(2):3782–3787
7. Pozna C, Precup RE, Tar JK, Škrjanc I, Preitl S (2010) New results in modelling derived from bayesian filtering. Knowl-Based Syst 23(2):182–194
8. Kirkpatrick S, Gellat C, Vecchi M (1983) Optimization by simulated annealing. Science 220(4598):671–680
9. Glover F (1989) Tabu search, part i. ORSA J Comput 1(3):190–206
10. Bell JE, McMullen PR (2004) Ant colony optimization techniques for the vehicle routing problem. Adv Eng Inf 18(1):41–48
11. Yu B, Yang ZZ, Yao B (2009) An improved ant colony optimization for vehicle routing problem. Eur J Oper Res 196(1):171–176

12. Goldberg D (1989) Genetic algorithms in search, optimization, and machine learning. Addison-Wesley Professional
13. De Jong K (1975) Analysis of the behavior of a class of genetic adaptive systems. PhD thesis, University of Michigan, Michigan
14. Kennedy J, Eberhart R et al (1995) Particle swarm optimization. In: Proceedings of IEEE international conference on neural networks, vol 4. Perth, Australia, pp. 1942–1948
15. Tang K, Li Z, Luo L, Liu B (2015) Multi-strategy adaptive particle swarm optimization for numerical optimization. Eng Appl Artif Intell 37:9–19
16. Yang XS (2009) Firefly algorithms for multimodal optimization. In: Stochastic algorithms: foundations and applications. Springer, pp 169–178
17. Atashpaz-Gargari E, Lucas C (2007) Imperialist competitive algorithm: an algorithm for optimization inspired by imperialistic competition. In: IEEE congress on evolutionary computation. IEEE, pp 4661–4667
18. Yang XS, Deb S (2009) Cuckoo search via lévy flights. In: World congress on nature & biologically inspired computing. NaBIC 2009. IEEE, pp 210–214
19. Karaboga D, Basturk B (2007) A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm. J Global Optim 39(3):459–471
20. Yang XS (2010) A new metaheuristic bat-inspired algorithm. In: Nature inspired cooperative strategies for optimization (NICSO 2010). Springer, pp 65–74
21. Cordeau J, Maischberger M (2012) A parallel iterated tabu search heuristic for vehicle routing problems. Comput Oper Res 39(9):2033–2050
22. Solomon MM (1987) Algorithms for the vehicle routing and scheduling problems with time window constraints. Oper Res 35(2):254–265
23. Desrochers M, Desrosiers J, Solomon M (1992) A new optimization algorithm for the vehicle routing problem with time windows. Oper Res 40(2):342–354
24. Gambardella LM, Taillard É, Agazzi G (1999) Macs-vrptw: a multiple colony system for vehicle routing problems with time windows. In: New ideas in optimization. Citeseer
25. Dabia S, Ropke S, Van Woensel T (2019) Cover inequalities for the vehicle routing problem with time windows and shifts. Transp Sci
26. Belhaiza S (2019) MHallah R, Brahim GB, Laporte G (2019) Three multi-start data-driven evolutionary heuristics for the vehicle routing problem with multiple time windows. J Heuristics 25(3):485–515
27. Cordeau JF, Desaulniers G, Desrosiers J, Solomon MM, Soumis F (2001) VRP with time windows. The Veh Routing Prob 9:157–193
28. Taillard É, Badeau P, Gendreau M, Guertin F, Potvin JY (1997) A tabu search heuristic for the vehicle routing problem with soft time windows. Transp Sci 31(2):170–186
29. Baldacci R, Battarra M, Vigo D (2008) Routing a heterogeneous fleet of vehicles. In: The vehicle routing problem: latest advances and new challenges. Springer, pp 3–27
30. Dethloff J (2001) Vehicle routing and reverse logistics: the vehicle routing problem with simultaneous delivery and pick-up. OR-Spektrum 23(1):79–96
31. Savelsbergh MW, Sol M (1995) The general pickup and delivery problem. Transp Sci 29(1):17–29
32. Goetschalckx M, Jacobs-Blecha C (1989) The vehicle routing problem with backhauls. Eur J Oper Res 42(1):39–51
33. Caceres-Cruz J, Arias P, Guimarans D, Riera D, Juan AA (2015) Rich vehicle routing problem: survey. ACM Comput Surv (CSUR) 47(2):32
34. Lahyani R, Khemakhem M, Semet F (2015) Rich vehicle routing problems: from a taxonomy to a definition. Eur J Oper Res 241(1):1–14
35. Osaba E, Diaz F, Onieva E (2013) A novel meta-heuristic based on soccer concepts to solve routing problems. In: Proceedings of the 15th annual conference companion on genetic and evolutionary computation. ACM, pp 1743–1744
36. Cordeau JF, Desaulniers G, Desrosiers J, Solomon MM, Soumis F (2002) The vehicle routing problem. VRP with time windows, pp 157–193

37. Van Breedam A (1995) Improvement heuristics for the vehicle routing problem based on simulated annealing. Eur J Oper Res 86(3):480–490
38. Homberger J, Gehring H (1999) Two evolutionary metaheuristics for the vehicle routing problem with time windows. INFOR: Inf Syst Oper Res 37(3):297–318
39. Russell RA (1995) Hybrid heuristics for the vehicle routing problem with time windows. Transp Sci 29(2):156–166
40. Christofides N, Eilon S (1969) An algorithm for the vehicle-dispatching problem. J Oper Res Soc 20(3):309–318
41. Fisher ML (1994) Optimal solution of vehicle routing problems using minimum k-trees. Oper Res 42(4):626–642
42. Golden BL, Wasil EA, Kelly JP, Chao IM (1998) The impact of metaheuristics on solving the vehicle routing problem: algorithms, problem sets, and computational results. In: Fleet management and logistics. Springer, pp 33–56
43. Gillett BE, Johnson JG (1976) Multi-terminal vehicle-dispatch algorithm. Omega 4(6):711–718
44. Cordeau JF, Gendreau M, Laporte G (1997) A tabu search heuristic for periodic and multi-depot vehicle routing problems. Netw Int J 30(2):105–119
45. Potvin JY, Bengio S (1996) The vehicle routing problem with time windows part ii: genetic search. INFORMS J Comput 8(2):165–172
46. Baker BM, Ayechew M (2003) A genetic algorithm for the vehicle routing problem. Comput Oper Res 30(5):787–800
47. Chiang WC, Russell RA (1996) Simulated annealing metaheuristics for the vehicle routing problem with time windows. Ann Oper Res 63(1):3–27
48. Potvin JY (1993) State-of-the-art survey the traveling salesman problem: a neural network perspective. ORSA J Comput 5(4):328–348
49. Gendreau M, Hertz A, Laporte G (1994) A tabu search heuristic for the vehicle routing problem. Manage Sci 40(10):1276–1290
50. Osman IH (1993) Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. Ann Oper Res 41(4):421–451
51. Thangiah SR, Osman IH, Sun T (1994) Hybrid genetic algorithm, simulated annealing and tabu search methods for vehicle routing problems with time windows. Computer Science Department, Slippery Rock University, Technical Report SRU CpSc-TR-94-27 69
52. Tavakkoli-Moghaddam R, Safaei N, Gholipour Y (2006) A hybrid simulated annealing for capacitated vehicle routing problems with the independent route length. Appl Math Comput 176(2):445–454
53. Precup RE, David RC (2019) Nature-inspired optimization algorithms for fuzzy controlled servo systems. Butterworth-Heinemann
54. Al Chen, Yang Gk Wu, Zm, (2006) Hybrid discrete particle swarm optimization algorithm for capacitated vehicle routing problem. J Zhejiang Univ-Sci A 7(4):607–614
55. Ai TJ, Kachitvichyanukul V (2009) A particle swarm optimization for the vehicle routing problem with simultaneous pickup and delivery. Comput Oper Res 36(5):1693–1702
56. Bullnheimer B, Hartl RF, Strauss C (1999) An improved ant system algorithm for the vehicle routing problem. Ann Operations research 89:319–328
57. Bräysy O (2003) A reactive variable neighborhood search for the vehicle-routing problem with time windows. INFORMS J Comput 15(4):347–368
58. Kytöjoki J, Nuortio T, Bräysy O, Gendreau M (2007) An efficient variable neighborhood search heuristic for very large scale vehicle routing problems. Comput Oper Res 34(9):2743–2757
59. Polacek M, Hartl RF, Doerner K, Reimann M (2004) A variable neighborhood search for the multi depot vehicle routing problem with time windows. J Heuristics 10(6):613–627
60. Zhou Y, Luo Q, Xie J, Zheng H (2016) A hybrid bat algorithm with path relinking for the capacitated vehicle routing problem. In: Metaheuristics and optimization in civil engineering. Springer, pp 255–276

61. Zheng HQ, Zhou Y, Luo Q (2013) A hybrid cuckoo search algorithm-grasp for vehicle routing problem. J Convergence Inf Technol 8(3):
62. Szeto WY, Wu Y, Ho SC (2011) An artificial bee colony algorithm for the capacitated vehicle routing problem. Eur J Oper Res 215(1):126–135
63. Zhou Y, Wang R (2016) An improved flower pollination algorithm for optimal unmanned undersea vehicle path planning problem. Int J Pattern Recogn Artif Intell 30(04):1659010
64. Lin C, Choy KL, Ho GT, Chung SH, Lam H (2014) Survey of green vehicle routing problem: past and future trends. Expert Syst Appl 41(4):1118–1138
65. Golden BL, Raghavan S, Wasil EA (2008) The vehicle routing problem: latest advances and new challenges, vol 43. Springer Science & Business Media
66. Laporte G (2009) Fifty years of vehicle routing. Transp Sci 43(4):408–416
67. Larrañaga P, Kuijpers CMH, Murga RH, Inza I, Dizdarevic S (1999) Genetic algorithms for the travelling salesman problem: a review of representations and operators. Artif Intell Rev 13(2):129–170
68. Cheng R, Gen M, Tsujimura Y (1996) A tutorial survey of job-shop scheduling problems using genetic algorithmsi representation. Comput Ind Eng 30(4):983–997
69. Cheng R, Gen M, Tsujimura Y (1999) A tutorial survey of job-shop scheduling problems using genetic algorithms, part ii: hybrid genetic search strategies. Comput Ind Eng 36(2):343–364
70. Ruiz E, Soto-Mendoza V, Barbosa AER, Reyes R (2019) Solving the open vehicle routing problem with capacity and distance constraints with a biased random key genetic algorithm. Comput Ind Eng 133:207–219
71. Saidi-Mehrabad M (2019) An optimization model for heterogeneous vehicle routing and scheduling problem with fixed cost and green reverse logistics network using genetic algorithm. J Optim Ind Eng
72. Baniamerian A, Bashiri M, Tavakkoli-Moghaddam R (2019) Modified variable neighborhood search and genetic algorithm for profitable heterogeneous vehicle routing problem with cross-docking. Appl Soft Comput 75:441–460
73. Long J, Sun Z, Pardalos PM, Hong Y, Zhang S, Li C (2019) A hybrid multi-objective genetic local search algorithm for the prize-collecting vehicle routing problem. Inf Sci 478:40–61
74. Koç Ç, Bektaş T, Jabali O, Laporte G (2015) A hybrid evolutionary algorithm for heterogeneous fleet vehicle routing problems with time windows. Comput Oper Res 64:11–27
75. Mohammed MA, Ghani MKA, Hamed RI, Mostafa SA, Ahmad MS, Ibrahim DA (2017) Solving vehicle routing problem by using improved genetic algorithm for optimal solution. J Comput Sci 21:255–262
76. Wang S, Lu Z, Wei L, Ji G, Yang J (2016) Fitness-scaling adaptive genetic algorithm with local search for solving the multiple depot vehicle routing problem. Simulation 92(7):601–616
77. Yang H, Yang S, Xu Y, Cao E, Lai M, Dong Z (2015) Electric vehicle route optimization considering time-of-use electricity price by learnable partheno-genetic algorithm. IEEE Trans Smart Grid 6(2):657–666
78. Shi Y, Boudouh T, Grunder O (2017) A hybrid genetic algorithm for a home health care routing problem with time window and fuzzy demand. Expert Syst Appl 72:160–176
79. Xiao Y, Konak A (2017) A genetic algorithm with exact dynamic programming for the green vehicle routing & scheduling problem. J Cleaner Prod 167:1450–1463
80. Lai DS, Demirag OC, Leung JM (2016) A tabu search heuristic for the heterogeneous vehicle routing problem on a multigraph. Transp Res Part E Logistics Transp Rev 86:32–52
81. Qiu M, Fu Z, Eglese R, Tang Q (2018) A tabu search algorithm for the vehicle routing problem with discrete split deliveries and pickups. Comput Oper Res 100:102–116
82. Sicilia JA, Quemada C, Royo B, Escuín D (2016) An optimization algorithm for solving the rich vehicle routing problem based on variable neighborhood search and tabu search metaheuristics. J Comput Appl Math 291:468–477
83. Silvestrin PV, Ritt M (2017) An iterated tabu search for the multi-compartment vehicle routing problem. Comput Oper Res 81:192–202
84. Niu Y, Yang Z, Chen P, Xiao J (2018) A hybrid tabu search algorithm for a real-world open vehicle routing problem involving fuel consumption constraints. Complexity

85. Molina JC, Eguia I, Racero J (2019) Reducing pollutant emissions in a waste collection vehicle routing problem using a variable neighborhood tabu search algorithm: a case study. TOP 27(2):253–287
86. Goeke D (2019) Granular tabu search for the pickup and delivery problem with time windows and electric vehicles. Eur J Oper Res 278(3):821–836
87. Bernal J, Escobar JW, Paz JC, Linfati R, Gatica G (2018) A probabilistic granular tabu search for the distance constrained capacitated vehicle routing problem. Int J Ind Syst Eng 29(4):453–477
88. Xia Y, Fu Z, Pan L, Duan F (2018) Tabu search algorithm for the distance-constrained vehicle routing problem with split deliveries by order. PloS one 13(5):e0195457
89. Vincent FY, Redi AP, Jewpanya P, Lathifah A, Maghfiroh MF, Masruroh NA (2019) A simulated annealing heuristic for the heterogeneous fleet pollution routing problem. In: Environmental sustainability in Asian logistics and supply chains. Springer, pp 171–204
90. Karagul, K., Sahin, Y., Aydemir, E., Oral, A.: A simulated annealing algorithm based solution method for a green vehicle routing problem with fuel consumption. In: Lean and green supply chain management. Springer, pp 161–187
91. Rezaei N, Ebrahimnejad S, Moosavi A, Nikfarjam A (2019) A green vehicle routing problem with time windows considering the heterogeneous fleet of vehicles: two metaheuristic algorithms. Eur J Ind Eng 13(4):507–535
92. Vincent FY, Redi AP, Hidayat YA, Wibowo OJ (2017) A simulated annealing heuristic for the hybrid vehicle routing problem. Appl Soft Comput 53:119–132
93. Normasari NME, Yu VF, Bachtiyar C et al (2019) A simulated annealing heuristic for the capacitated green vehicle routing problem. Math Prob Eng
94. Wei L, Zhang Z, Zhang D, Leung SC (2018) A simulated annealing algorithm for the capacitated vehicle routing problem with two-dimensional loading constraints. Eur J Oper Res 265(3):843–859
95. Vincent FY, Lin SY (2015) A simulated annealing heuristic for the open location-routing problem. Comput Oper Res 62:184–196
96. Mahmudy WF (2016) Improved simulated annealing for optimization of vehicle routing problem with time windows (vrptw). Kursor 7(3):
97. Wang C, Mu D, Zhao F, Sutherland JW (2015) A parallel simulated annealing method for the vehicle routing problem with simultaneous pickup-delivery and time windows. Comput Ind Eng 83:111–122
98. Afifi S, Dang DC, Moukrim A (2016) Heuristic solutions for the vehicle routing problem with time windows and synchronized visits. Optim Lett 10(3):511–525
99. Yu VF, Lin SY (2016) Solving the location-routing problem with simultaneous pickup and delivery by simulated annealing. Int J Prod Res 54(2):526–549
100. Mu D, Wang C, Zhao F, Sutherland JW (2016) Solving vehicle routing problem with simultaneous pickup and delivery using parallel simulated annealing algorithm. Int J Shipping Transp Logistics 8(1):81–106
101. Shaabani H, Kamalabadi IN (2016) An efficient population-based simulated annealing algorithm for the multi-product multi-retailer perishable inventory routing problem. Comput Ind Eng 99:189–201
102. Marinakis Y, Marinaki M (2010) A hybrid genetic-particle swarm optimization algorithm for the vehicle routing problem. Expert Syst Appl 37(2):1446–1455
103. Yao B, Yu B, Hu P, Gao J, Zhang M (2016) An improved particle swarm optimization for carton heterogeneous vehicle routing problem with a collection depot. Ann Oper Res 242(2):303–320
104. Kumar RS, Kondapaneni K, Dixit V, Goswami A, Thakur LS, Tiwari M (2016) Multi-objective modeling of production and pollution routing problem with time window: a self-learning particle swarm optimization approach. Comput Ind Eng 99:29–40
105. Kaiwartya O, Kumar S, Lobiyal D, Tiwari PK, Abdullah AH, Hassan AN (2015) Multiobjective dynamic vehicle routing problem and time seed based solution using particle swarm optimization. J Sens

106. Okulewicz M, Mandziuk J (2016) A particle swarm optimization hyper-heuristic for the dynamic vehicle routing problem. In: International conference on bioinspired optimization methods and their applications, pp 215–227
107. Chen RM, Shen YM, Hong WZ (2019) Neural-like encoding particle swarm optimization for periodic vehicle routing problems. Expert Syst Appl, 112833
108. Chen J, Shi J (2019) A multi-compartment vehicle routing problem with time windows for urban distribution-a comparison study on particle swarm optimization algorithms. Comput Ind Eng 133:95–106
109. Li Y, Lim MK, Tseng ML (2019) A green vehicle routing model based on modified particle swarm optimization for cold chain logistics. Ind Manage Data Syst 119(3):473–494
110. Norouzi N, Sadegh-Amalnick M, Alinaghiyan M (2015) Evaluating of the particle swarm optimization in a periodic vehicle routing problem. Measurement 62:162–169
111. Wei Q, Guo Z, Lau HC, He Z (2019) An artificial bee colony-based hybrid approach for waste collection problem with midway disposal pattern. Appl Soft Comput 76:629–637
112. Hao, D., CHENG, H.j., Xian, S.: Modified artificial bee colony algorithm for the capacitated vehicle routing problem. DEStech Trans Soc Sci Educ Hum Sci (AMSE)
113. George S, Binu S (2018) Vehicle route optimisation using artificial bees colony algorithm and cuckoo search algorithm-a comparative study. Int J Appl Eng Res 13(2):953–959
114. Ng K, Lee C, Zhang S, Wu K, Ho W (2017) A multiple colonies artificial bee colony algorithm for a capacitated vehicle routing problem and re-routing strategies under time-dependent traffic congestion. Comput Ind Eng 109:151–168
115. Yu S, Tai C, Liu Y, Gao L (2016) An improved artificial bee colony algorithm for vehicle routing problem with time windows: a real case in Dalian. Adv Mech Eng 8(8):1687814016665298
116. Yin PY, Chuang YL (2016) Adaptive memory artificial bee colony algorithm for green vehicle routing with cross-docking. Appl Math Modell 40(21–22):9302–9315
117. Alzaqebah M, Abdullah S, Jawarneh S (2016) Modified artificial bee colony for the vehicle routing problems with time windows. SpringerPlus 5(1):1298
118. Yao B, Yan Q, Zhang M, Yang Y (2017) Improved artificial bee colony algorithm for vehicle routing problem with time windows. PloS One 12(9):e0181275
119. Zhang S, Lee C (2015) An improved artificial bee colony algorithm for the capacitated vehicle routing problem. In: IEEE international conference on systems, man, and cybernetics. IEEE, pp 2124–2128
120. Gao S, Wang Y, Cheng J, Inazumi Y, Tang Z (2016) Ant colony optimization with clustering for solving the dynamic location routing problem. Appl Math Comput 285:149–173
121. Xu H, Pu P, Duan F (2018) Dynamic vehicle routing problems with enhanced ant colony optimization. Discrete Dyn Nat Soc
122. Wang X, Choi TM, Liu H, Yue X (2016) Novel ant colony optimization methods for simplifying solution construction in vehicle routing problems. IEEE Trans Intell Transp Syst 17(11):3132–3141
123. Kalayci CB, Kaya C (2016) An ant colony system empowered variable neighborhood search algorithm for the vehicle routing problem with simultaneous pickup and delivery. Expert Syst Appl 66:163–175
124. Li Y, Soleimani H, Zohal M (2019) An improved ant colony optimization algorithm for the multi-depot green vehicle routing problem with multiple objectives. J Cleaner Prod 227:1161–1172
125. Zhang S, Zhang W, Gajpal Y, Appadoo S (2019) Ant colony algorithm for routing alternate fuel vehicles in multi-depot vehicle routing problem. In: Decision science in action. Springer, pp 251–260
126. Jabir E, Panicker VV, Sridharan R (2017) Design and development of a hybrid ant colony-variable neighbourhood search algorithm for a multi-depot green vehicle routing problem. Transp Res Part D Transp Environ 57:422–457
127. Yao B, Chen C, Song X, Yang X (2019) Fresh seafood delivery routing problem using an improved ant colony optimization. Ann Oper Res 273(1–2):163–186

128. Tirkolaee EB, Alinaghian M, Hosseinabadi AAR, Sasi MB, Sangaiah AK (2018) An improved ant colony optimization for the multi-trip capacitated arc routing problem. Comput Electr Eng
129. Abdulkader MM, Gajpal Y, ElMekkawy TY (2015) Hybridized ant colony algorithm for the multi compartment vehicle routing problem. Appl Soft Comput 37:196–203
130. Kuo R, Wibowo B, Zulvia F (2016) Application of a fuzzy ant colony system to solve the dynamic vehicle routing problem with uncertain service time. Appl Math Modell 40(23–24):9990–10001
131. Pavlyukevich I (2007) Lévy flights, non-local search and simulated annealing. J Comput Phys 226(2):1830–1844
132. Teymourian E, Kayvanfar V, Komaki GM, Zandieh M (2016) Enhanced intelligent water drops and cuckoo search algorithms for solving the capacitated vehicle routing problem. Inf Sci 334:354–378
133. Santillan JH, Tapucar S, Manliguez C, Calag V (2018) Cuckoo search via lévy flights for the capacitated vehicle routing problem. Int J Ind Eng 14(2):293–304
134. Goli A, Aazami A, Jabbarzadeh A (2018) Accelerated cuckoo optimization algorithm for capacitated vehicle routing problem in competitive conditions. Int J Artif Intell 16(1):88–112
135. Alssager M, Othman ZA (2016) Taguchi-based parameter setting of cuckoo search algorithm for capacitated vehicle routing problem. In: Advances in machine learning and signal processing. Springer, pp 71–79
136. Chen X, Wang J (2016) A novel hybrid cuckoo search algorithm for optimizing vehicle routing problem in logistics distribution system. J Comput Theor Nanosci 13(1):114–119
137. Xiao L, Dridi M, Hajjam El Hassani A, Fei H, Lin W (2018) An improved cuckoo search for a patient transportation problem with consideration of reducing transport emissions. Sustainability 10(3):793
138. Gj Wang, Zhang YB, Chen JW (2011) A novel algorithm to solve the vehicle routing problem with time windows: imperialist competitive algorithm. Adv Inf Sci Serv Sci 3(5):108–116
139. Ghorbani A, Jokar MRA (2016) A hybrid imperialist competitive-simulated annealing algorithm for a multisource multi-product location-routing-inventory problem. Comput Ind Eng 101:116–127
140. Shamshirband S, Shojafar M, Hosseinabadi AAR, Abraham A (2015) Ovrp_ica: an imperialist-based optimization algorithm for the open vehicle routing problem. In: International conference on hybrid artificial intelligence systems. Springer, pp 221–233
141. Hasani Goodarzi A, Nahavandi N, Zegordi SH (2018) A multi-objective imperialist competitive algorithm for vehicle routing problem in cross-docking networks with time windows. J Ind Syst Eng 11(1):1–23
142. Golmohammadi A, Bonab S, Parishani A (2016) A multi-objective location routing problem using imperialist competitive algorithm. Int J Ind Eng Comput 7(3):481–488
143. Nekooghadirli N, Tavakkoli-Moghaddam R, Ghezavati V (2014) Efficiency of a multi-objective imperialist competitive algorithm: a bi-objective location-routing-inventory problem with probabilistic routes. J AI Data Min 2(2):105–112
144. Taha A, Hachimi M, Moudden A (2015) Adapted bat algorithm for capacitated vehicle routing problem. Int Rev Comput Softw 10(6):610–619
145. Cai Y, Qi Y, Cai H, Huang H, Chen H (2019) Chaotic discrete bat algorithm for capacitated vehicle routing problem. Int J Auton Adapt Commun Syst 12(2):91–108
146. Taha A, Hachimi M, Moudden A (2017) A discrete bat algorithm for the vehicle routing problem with time windows. In: 2017 International colloquium on logistics and supply chain management (LOGISTIQUA). IEEE, pp 65–70
147. Osaba E, Carballedo R, Yang XS, Fister Jr I, Lopez-Garcia, P., Del Ser, J.: On efficiently solving the vehicle routing problem with time windows using the bat algorithm with random reinsertion operators. In: Nature-inspired algorithms and applied optimization. Springer, pp 69–89
148. Osaba E, Yang XS, Fister I Jr, Del Ser J, Lopez-Garcia P, Vazquez-Pardavila AJ (2019) A discrete and improved bat algorithm for solving a medical goods distribution problem with pharmacological waste collection. Swarm Evol Comput 44:273–286

149. Osaba E, Yang XS, Diaz F, Lopez-Garcia P, Carballedo R (2016) An improved discrete bat algorithm for symmetric and asymmetric traveling salesman problems. Eng Appl Artif Intell 48:59–71
150. Osaba E, Del Ser J, Sadollah A, Bilbao MN, Camacho D (2018) A discrete water cycle algorithm for solving the symmetric and asymmetric traveling salesman problem. Appl Soft Comput 71:277–290
151. Goel R, Maini R (2018) A hybrid of ant colony and firefly algorithms (hafa) for solving vehicle routing problems. J Comput Sci 25:28–37
152. Aggarwal D, Kumar V (2018) An improved firefly algorithm for the vehicle routing problem with time windows. In: 2018 international conference on advances in computing, communications and informatics (ICACCI). IEEE, pp 222–229
153. Matthopoulos PP, Sofianopoulou S (2018) A firefly algorithm for the heterogeneous fixed fleet VRP. Int J Ind Syst Eng 33(1):
154. Alinaghian M, Naderipour M (2016) A novel comprehensive macroscopic model for time-dependent vehicle routing problem with multi-alternative graph to reduce fuel consumption: a case study. Comput Ind Eng 99:210–222
155. Del Ser J, Torre-Bastida AI, Lana I, Bilbao MN, Perfecto C (2017) Nature-inspired heuristics for the multiple-vehicle selective pickup and delivery problem under maximum profit and incentive fairness criteria. In: IEEE congress on evolutionary computation, IEEE, pp 480–487
156. Li J, Li T, Yu Y, Zhang Z, Pardalos PM, Zhang Y, Ma Y (2019) Discrete firefly algorithm with compound neighborhoods for asymmetric multi-depot vehicle routing problem in the maintenance of farm machinery. Appl Soft Comput 81:105460
157. Osaba E, Yang XS, Diaz F, Onieva E, Masegosa AD, Perallos A (2017) A discrete firefly algorithm to solve a rich vehicle routing problem modelling a newspaper distribution system with recycling policy. Soft Comput 21(18):5295–5308
158. Shankar RBS, Reddy KD, Venkataramaiah P (2018) Solution to a capacitated vehicle routing problem using heuristics and firefly algorithm. Int J Appl Eng Res 13(21):15247–15254
159. Osaba E, Carballedo R, Yang XS, Diaz F (2016) An evolutionary discrete firefly algorithm with novel operators for solving the vehicle routing problem with time windows. In: Nature-inspired computation in engineering. Springer, pp 21–41
160. Del Ser J, Osaba E, Molina D, Yang XS, Salcedo-Sanz S, Camacho D, Das S, Suganthan PN, Coello CAC, Herrera F (2019) Bio-inspired computation: where we stand and what's next. Swarm Evol Comput 48:220–250
161. Yang XS (2012) Flower pollination algorithm for global optimization. In: International conference on unconventional computing and natural computation. Springer, pp 240–249
162. Hu W (2019) An improved flower pollination algorithm for optimization of intelligent logistics distribution center. Adv Prod Eng Manage 14(2):177–188
163. Tan Y, Zhu Y (2010) Fireworks algorithm for optimization. In: International conference in swarm intelligence. Springer, pp 355–364
164. Yang W, Ke L (2019) An improved fireworks algorithm for the capacitated vehicle routing problem. Front Comput Sci 13(3):552–564
165. Cai Y, Qi Y, Chen H, Cai H, Hejlesen O (2018) Quantum fireworks evolutionary algorithm for vehicle routing problem in supply chain with multiple time windows. In: 2nd IEEE advanced information management, communicates, electronic and automation control conference (IMCEC). IEEE, pp 383–388
166. Alihodzic A (2016) Fireworks algorithm with new feasibility-rules in solving uav path planning. In: 2016 3rd international conference on soft computing & machine intelligence (ISCMI). IEEE, pp 53–57
167. Cheng MY, Prayogo D (2014) Symbiotic organisms search: a new metaheuristic optimization algorithm. Comput Struct 139:98–112
168. Vincent FY, Redi AP, Yang CL, Ruskartina E, Santosa B (2017) Symbiotic organisms search and two solution representations for solving the capacitated vehicle routing problem. Appl Soft Comput 52:657–672

169. Eki R, Vincent FY, Budi S, Redi AP (2015) Symbiotic organism search (sos) for solving the capacitated vehicle routing problem. World Acad Sci Eng Technol Int J Mech Aerosp Ind Mechatron Manuf Eng 9(5):850–854
170. Geem ZW, Kim JH, Loganathan GV (200) A new heuristic optimization algorithm: harmony search. Simulation 76(2):60–68
171. Del Ser J, Bilbao MN, Perfecto C, Salcedo-Sanz S (2016) A harmony search approach for the selective pick-up and delivery problem with delayed drop-off. In: Harmony search algorithm. Springer, pp 121–131
172. Wang Z, Lu Y, Zhao L, Cao N (2018) Improved harmony search algorithm for truck scheduling problem in multiple-door cross-docking systems. Discrete Dyn Nat Soc
173. Yassen ET, Ayob M, Nazri MZA, Sabar NR (2015) Meta-harmony search algorithm for the vehicle routing problem with time windows. Inf Sci 325:140–158
174. Chen S, Chen R, Gao J (2017) A modified harmony search algorithm for solving the dynamic vehicle routing problem with time windows. Sci Program
175. Haddad OB, Afshar A, Mariño MA (2006) Honey-bees mating optimization (hbmo) algorithm: a new heuristic approach for water resources optimization. Water Resour Manag 20(5):661–680
176. Dorigo M et al (2018) A honey bees mating optimization algorithm with path relinking for the vehicle routing problem with stochastic demands. In: Swarm intelligence: 11th international conference, ANTS 2018, Rome, Italy, Oct 29–31, Proceedings, vol 11172. Springer, pp 423
177. Fatnassi E, Chebbi O, Chaouachi J (2016) Discrete honeybee mating optimization algorithm for the routing of battery-operated automated guidance electric vehicles in personal rapid transit systems. Swarm Evol Comput 26:35–49
178. Marinakis Y, Marinaki M (2015) Combinatorial neighborhood topology bumble bees mating optimization for the vehicle routing problem with stochastic demands. Soft Comput 19(2):353–373
179. Ruttanateerawichien K, Kurutach W, Pichpibul T (2016) A new efficient and effective golden-ball-based technique for the capacitated vehicle routing problem. In: 2016 IEEE/ACIS 15th international conference on computer and information science (ICIS). IEEE, pp 1–5
180. Osaba E, Díaz F (2016) Design and implementation of a combinatorial optimization multi-population meta-heuristic for solving vehicle routing problems. Int J Interact Multimedia Artif Intell 4(2):89–90
181. Ruttanateerawichien K, Kurutach W (2018) An improved golden ball algorithm for the vehicle routing problem with simultaneous pickup and delivery. In: 2018 IEEE/ACIS 17th international conference on computer and information science (ICIS). IEEE, pp 258–262
182. Osaba E, Diaz F, Onieva E (2014) Golden ball: a novel meta-heuristic to solve combinatorial optimization problems based on soccer concepts. Appl Intell 41(1):145–166
183. Guezouli L, Bensakhria M, Abdelhamid S (2018) Efficient golden-ball algorithm based clustering to solve the multi-depot VRP with time windows. Int J Appl Evol Comput (IJAEC) 9(1):1–16
184. Osaba E, Díaz F, Carballedo R, Onieva E, Perallos A (2014) Focusing on the golden ball metaheuristic: an extended study on a wider set of problems. Sci World J
185. Shi Y (2011) Brain storm optimization algorithm. In: International conference in swarm intelligence. Springer, pp 303–309
186. Yan X, Hao Z, Huang H, Li G (2016) Human-computer cooperative brain storm optimization algorithm for the two-echelon vehicle routing problem. In: IEEE congress on evolutionary computation (CEC). IEEE, pp 2676–2681
187. Ke L (2018) A brain storm optimization approach for the cumulative capacitated vehicle routing problem. Memetic Comput 10(4):411–421
188. Wu L, He Z, Chen Y, Wu D, Cui J (2019) Brainstorming-based ant colony optimization for vehicle routing with soft time windows. IEEE Access 7:19643–19652
189. Dolicanin E, Fetahovic I, Tuba E, Capor-Hrosik R, Tuba M (2018) Unmanned combat aerial vehicle path planning by brain storm optimization algorithm. Stud Inf Control 27(1):15–24

190. Krishnanand K, Ghose D (2009) Glowworm swarm optimisation: a new method for optimising multi-modal functions. Int J Comput Intell Stud 1(1):93–119
191. Marinaki M, Marinakis Y (2016) A glowworm swarm optimization algorithm for the vehicle routing problem with stochastic demands. Expert Syst Appl 46:145–163
192. Yarinezhad R, Sarabi A (2019) A new routing algorithm for vehicular ad-hoc networks based on glowworm swarm optimization algorithm. J AI Data Min 7(1):69–76
193. Chen X, Zhou Y, Tang Z, Luo Q (2017) A hybrid algorithm combining glowworm swarm optimization and complete 2-opt algorithm for spherical travelling salesman problems. Appl Soft Comput 58:104–114
194. Dong W, Zhou K, Zhang G, Chao HC (2018) Modified discrete glowworm swarm optimization algorithm based on time window division for multi-objective vrptw. J Internet Technol 19(1):001–013
195. Mirjalili S, Lewis A (2016) The whale optimization algorithm. Adv Eng Softw 95:51–67
196. Horng MF, Dao TK, Shieh CS et al (2017) A multi-objective optimal vehicle fuel consumption based on whale optimization algorithm. In: Advances in intelligent information hiding and multimedia signal processing. Springer, pp 371–380
197. Mofid-Nakhaee E, Barzinpour F (2019) A multi-compartment capacitated arc routing problem with intermediate facilities for solid waste collection using hybrid adaptive large neighborhood search and whale algorithm. Waste Manage Res 37(1):38–47
198. Yadav H, Lithore U, Agrawal N (2017) An enhancement of whale optimization algorithm using ann for routing optimization in ad-hoc network. Int J Adv Technol Eng Explor 4(36):161–167
199. Passino KM (2002) Biomimicry of bacterial foraging for distributed optimization and control. IEEE Control Syst Mag 22(3):52–67
200. Gan X, Liu L, Niu B, Tan L, Zhang F, Liu J (2015) Srbfos for solving the heterogeneous fixed fleet vehicle routing problem. In: International conference on intelligent computing. Springer, pp 725–732
201. Tan L, Lin F, Wang H (2015) Adaptive comprehensive learning bacterial foraging optimization and its application on vehicle routing problem with time windows. Neurocomputing 151:1208–1215
202. Niu B, Wang H, Tan LJ, Li L, Wang JW (2012) Vehicle routing problem with time windows based on adaptive bacterial foraging optimization. In: International conference on intelligent computing. Springer, pp 672–679
203. Rashedi E, Nezamabadi-Pour H, Saryazdi S (2009) Gsa: a gravitational search algorithm. Inf Sci 179(13):2232–2248
204. MirHassani S, Mohammadyari S (2014) Reduction of carbon emissions in VRP by gravitational search algorithm. Manage Environ Qual Int J 25(6):766–782
205. Hosseinabadi AAR, Kardgar M, Shojafar M, Shamshirband S, Abraham A (2016) Gravitational search algorithm to solve open vehicle routing problem. In: Innovations in bio-inspired computing and applications. Springer, pp 93–103
206. Duan H, Qiao P (2014) Pigeon-inspired optimization: a new swarm intelligence optimizer for air robot path planning. Int J Intell Comput Cybern 7(1):24–37
207. Hu C, Xia Y, Zhang J (2019) Adaptive operator quantum-behaved pigeon-inspired optimization algorithm with application to uav path planning. Algorithms 12(1):3
208. Zhang D, Duan H (2018) Social-class pigeon-inspired optimization and time stamp segmentation for multi-uav cooperative path planning. Neurocomputing 313:229–246
209. Gheraibia Y, Moussaoui A (2013) Penguins search optimization algorithm (pesoa). In: International conference on industrial, engineering and other applications of applied intelligent systems. Springer, 222–231
210. Ammi M, Chikhi S (2016) Cooperative parallel metaheuristics based penguin optimization search for solving the vehicle routing problem. Int J Appl Metaheuristic Comput (IJAMC) 7(1):1–18
211. Wang GG, Deb S, Coelho LdS (2015) Elephant herding optimization. In: 3rd international symposium on computational and business intelligence (ISCBI). IEEE, pp 1–5

212. Wang GG, Deb S, Cui Z (2019) Monarch butterfly optimization. Neural Comput Appl 31(7):1995–2014
213. Chen S, Chen R, Gao J (2017) A monarch butterfly optimization for the dynamic vehicle routing problem. Algorithms 10(3):107
214. LaTorre A, Muelas S, Peña JM (2012) Multiple offspring sampling in large scale global optimization. In: IEEE congress on evolutionary computation. IEEE, pp 1–8
215. Molina D, LaTorre A, Herrera F (2018) Shade with iterative local search for large-scale global optimization. In IEEE congress on evolutionary computation (CEC). IEEE, pp 1–8
216. Ma X, Li X, Zhang Q, Tang K, Liang Z, Xie W, Zhu Z (2018) A survey on cooperative co-evolutionary algorithms. IEEE Trans Evolut Comput
217. Kramer O (2008) Self-adaptive heuristics for evolutionary computation, vol 147. Springer
218. Gupta A, Ong YS, Feng L (2017) Insights on transfer optimization: because experience is the best teacher. IEEE Trans Emerg Top Comput Intell 2(1):51–64
219. Ong YS, Gupta A (2016) Evolutionary multitasking: a computer science view of cognitive multitasking. Cogn Comput 8(2):125–142
220. Gupta A, Ong YS, Feng L (2015) Multifactorial evolution: toward evolutionary multitasking. IEEE Trans Evol Comput 20(3):343–357
221. Konečnỳ J, McMahan HB, Ramage D, Richtárik P (2016) Federated optimization: distributed machine learning for on-device intelligence. arXiv preprint arXiv:1610.02527
222. Kaveh A, Zolghadr A (2016) A novel meta-heuristic algorithm: tug of war optimization. Iran Univ Sci Technol 6(4):469–492
223. Arora S, Singh S (2019) Butterfly optimization algorithm: a novel approach for global optimization. Soft Comput 23(3):715–734
224. Wang GG, Gao XZ, Zenger K, Coelho LdS (2018) A novel metaheuristic algorithm inspired by rhino herd behavior. In: Proceedings of The 9th EUROSIM congress on modelling and simulation. Number 142, Linköping University Electronic Press, pp 1026–1033
225. Sörensen K (2015) Metaheuristics the metaphor exposed. Int Trans Oper Res 22(1):3–18
226. Del Ser J, Osaba E, Sanchez-Medina JJ, Fister I (2019) Bioinspired computational intelligence and transportation systems: a long road ahead. IEEE Trans Intell Transp Syst
227. Durillo JJ, Nebro AJ (2011) jmetal: a java framework for multi-objective optimization. Adv Eng Softw 42(10):760–771
228. Benitez-Hidalgo A, Nebro AJ, Garcia-Nieto J, Oregi I, Del Ser J (2019) jmetalpy: a python framework for multi-objective optimization with metaheuristics. arXiv preprint arXiv:1903.02915

Chapter 4
Review of Tour Generation for Solving Traveling Salesman Problems

Aziz Ouaarab

1 Introduction

Optimization is the process of starting with a solution and trying iteratively to improve it otherwise seeking another better, until finding the optimum. Generally, optimization problems are not solvable with an exact method. Most of them are very difficult, and no algorithm is actually able to solve this class of problems in a polynomial time on a deterministic machine [1]. On the other hand, the representation of the solution space structure is related significantly to the nature of the problem and the way we see it, so the way the problem is modeled and then solved. Optimization problems are either continuous or discrete. Designing a model to a continuous optimization problem is not the same in terms of complexity while solving a combinatorial optimization problem.

It is obvious that seeking solution in the continuous space does not impose real constraints. To move between continuous solutions (change their components), we only need to modify the current values of its coordinates. This is the case in most continuous optimization problems, which can be considered as an advantage that avoids many technical obstacles related to how the coordinates are represented in the combinatorial space (in our case TSP solution space). As a blocking constraint to consider in TSP is that for a given solution, the coordinates of the visited cities are fixed. So, changing the visiting order of the cities is the only way to move from a solution to a new one. However, this change is not correlated to the solution fitness.

For solving TSP, a great number of approximate approaches are proposed. They are figured in one or all layers of the described architecture in this chapter. The first layer contains all methods that generate or construct an initial solution. This solution is generated randomly, without exigencies about its fitness [2, 3], or by using a

A. Ouaarab (✉)
Higher School of Technology, Essaouira,
Cadi Ayyad University of Marrakesh, Marrakesh, Morocco
e-mail: a.ouaarab@uca.ma

X.-S. Yang and Y.-X. Zhao (eds.), *Nature-Inspired Computation in Navigation and Routing Problems*, Springer Tracts in Nature-Inspired Computing,
https://doi.org/10.1007/978-981-15-1842-3_4

method to construct a "good" initial solution such as ant colony optimization [4]. In the second layer, the current solution is changed (perturbed) to produce a new improved one. Here, methods move in the solution space with the aim to find out the optimal solution in the neighborhood of the current solution, while the method is not trapped in a local optimum. Therefore, third layer accommodates techniques that make escaping from local optimum, possible. However, avoiding the local optimum is not the final objective of optimization process. Finding the global optimum is much more important and difficult. This is why in the last layer, methods that implement advanced techniques relatively to those of the other layers are grouped. Therefore, the reason of this chapter is to describe the important parts of approximately solving TSP procedures considering a model of layers.

In this chapter, we will show, by using TSP, how most of the proposed methods solve combinatorial optimization problems (COPs), find their place in the presented model. Because of its simplicity and reduced number of constraints, TSP represents a pertinent choice to highlight the layers of the shown model. Also, it can be transformed easily, to a great number of COPs [5, 6].

The remainder of this chapter is organized as follows: Sect. 2 introduces the traveling salesman problem by a historical context and its formal definition. Section 3 presents TSP as a combinatorial optimization problem solved by tour construction and improvement methods. Section 4 gives a description of the TSP combinatorial search space components. Section 5 first briefly describes the third and the fourth layer methods and then discusses the performance of each one. An example of how the layers are figured in one algorithm is shown in Sect. 6, and the chapter is concluded in Sect. 7.

2 Traveling Salesman Problem (TSP)

2.1 History

Suppose a commercial agent with a prescribed list of cities to be visited and closing the cycle by returning to the first visited city. To reduce the cost of the trip in terms of distance, he insists on some rules like: (1) Each city of his list is visited once and only once. (2) For a given pair of cities, the distance between them is known. These rules will define a problem that we will give it the name of "*Traveling Salesman Problem*" as mentioned by Lawler et al. in [7] and at another version of Tucker in 1983 in [42].

According to the two, previously, fixed rules, the traveling salesman problem (TSP) has a goal to find the shortest route or minimum trip cost to visit all the n cities exactly once and return back to the departure city. If n represents the number of the visited cities, the cardinal of tour possibilities or achievable TSP solutions is given as $n!$.

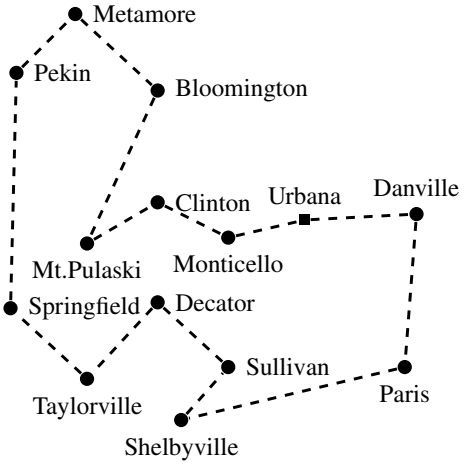


Fig. 1 Lincoln of Illinois tour in 1850 [8]

A fine historical example of the TSP is Lincoln's tour in 1850. Lincoln worked as a traveling lawyer with a circuit court. At the time, a circuit was a trip through several cities, whereas in each city, he stopped and treated local cases, and he spent a few days there. The visit that the traveling tribunal (and Lincoln) took in 1850 is illustrated in Fig. 1 (they started from Urbana and returned to it). He is very close to the optimum. So, a shorter solution was found, but it is not clear that it was shorter as much as the travel time in 1850 [8].

2.2 *Definitions*

Davendra defines formally TSP in [5] as follows:

Let $C = \{c_1, \ldots, c_n\}$ be the set of distinct cities in space, $E = \{(c_i, c_j) : i, j \in \{1, \ldots, n\}\}$ is the set of arcs between each pair of cities, and $d_{c_i c_j}$ the cost associated with each arc $(c_i, c_j) \in E$. The TSP consists of finding the minimum length of the closed tour that visits each city once and only once. Otherwise, an ordering π of cities such that the total time for the salesman is minimized. The cities $c_i \in C$ are represented by their coordinates (c_{ix}, c_{iy}), and $d_{c_i c_j} = \sqrt{(c_{ix} - c_{jx})^2 + (c_{iy} - c_{jy})^2}$ is the Euclidean distance between c_i and c_j. A tour can be represented as a circular permutation $\pi = (\pi(1), \pi(2), \ldots, \pi(n))$ cities from 1 to n if $\pi(i)$ is interpreted to be the city visited in the step i, $i = 1, \ldots, n$. The cost of a permutation (tour) is defined as

$$f(\pi) = \sum_{i=1}^{n-1} d_{\pi(i)\pi(i+1)} + d_{\pi(n)\pi(1)} \tag{1}$$

If $d_{c_i c_j} = d_{c_j c_i}$, we are talking about the symmetric Euclidean TSP, and if $d_{c_i c_j} \neq d_{c_j c_i}$ for at least one arc (c_i, c_j), the TSP is called asymmetric problem which is generally more difficult to solve. In what follows the TSP refers to the symmetric TSP.

The TSP can also be modeled as a weighted graph $G = (C, E)$. The set of n vertices C of the graph corresponds to the cities, and E is the set of arcs or edges which corresponds to the connections between the cities. These arcs are weighted, and this weight between its two cities represents the distance in the case of TSP. Let: (M_{ij}) be a distance (or cost) matrix associated with E. TSP consists of determining the minimum circuit distance passing through each vertex once and only once. In several applications, M can also be interpreted as a cost or travel time matrix. M is said to satisfy the triangle inequality if and only if $c_{ij} + c_{jk} \geq c_{iK}$ for all $i, j, k \in 1, 2, \ldots, n$. This occurs in Euclidean problems, i.e., when C is a set of points in $\mathbb{R}^2$, and $d_{c_i c_j}$ is the straight-line distance between c_i and c_j [9]. A TSP tour is a Hamiltonian cycle, and the optimal tour is the shortest Hamiltonian cycle.

2.3 Applications

TSP has mere appearance and statement that does not require much mathematical knowledge to understand it and a high level of thinking to get a solution. However, it is actually very important, either in its theoretical side as an experimental base for studying various optimization techniques or practical domain as a model for real-world problems. Lenstra et al. and Reinelt [6, 10] have shown various industrial and technological applications of TSP, such as overhauling gas turbine engines, X-ray crystallography, computer wiring, order-picking problem in warehouses, vehicle routing and mask plotting in PCBs production. Each of these problems can be formulated as a traveling salesman problem from, where most of them were not immediately recognized as TSP. Moreover, not only are the special cases of the TSP, but with the formulation as a TSP, it is essentially the simplest way to solve them [6, 11, 12].

3 Best Tour Generation

The traveling salesman problem belongs to a class of NP-hard problems [13] and is still a big challenge for researchers in this field. Papadimitriou [14] has shown that the complexity of Euclidean TSP is NP-complete. Even the problem of simply verifying whether a given solution is optimal is also intractable. No algorithm can obtain an exact solution or a solution with a predefined accuracy in a polynomial time. Its complexity of computation exponentially increases with the number of cities.

The hardness of TSP comes from its definition. Because, solving TSP is finding exactly the optimum, which is not yet possible in a polynomial time. To treat approx-

imately this situation, we can find a great variety of techniques [12]. Considering how these techniques generate the optimal solution for TSP, we will group them under two principle classes: tour construction in a short computation time even if its quality is moderate and iteratively improvement (of the built tour) to reach the optimal tour in a reasonable computation time.

3.1 Tour Construction

Related to our model, tour construction methods compose the first layer. As the first treatment on the problem instances, solutions are built in a reduced runtime. However, its quality is moderate. After building a solution, the method is stopped, and it never makes any improvement on the constructed solution. The idea is to produce a solution by an iterative addition of its components to the existing sub-solution (partial solution at a given iteration).

The most known construction methods are the greedy methods [15]. Among its variants, we cite the nearest neighbor algorithm. In the case of TSP, a salesman traveler searches iteratively the nearest city that remains outside the built tour. In another variant, the traveling salesman feeds the incomplete solution by the iterative selection of the shortest arcs and adds them to the current solution without closing the tour, as he did not reach arcs or increase the node degree more than 2.

We can also talk about insertion methods that start initially with an arc or a closed sub-solution of three arcs, and in the following, the insertion of the rest is done by some heuristics [16, 17].

These examples (and other construction methods are cited by Davendra [5]) show that the main objective of these methods is not seeking for the best solution, but the reduced calculation even if the solution quality is moderate. Despite their generated-solution quality, construction algorithms remain widely used but as auxiliaries in other methods. They are called to, initially, construct solutions or within their procedures.

3.2 Tour Improvement

To make improvement, algorithms begin with an initial solution, possibly generated by a construction method. They proceed to an iterative improvement of the solution until obtaining a not improvable (block situation) solution. Generally, they manage intelligently simple changes made to the current solution. Consider this, every change on the current solution leads to new one nearby in the search space. They, then, seek to improve solutions by making local moves in the neighborhood, while the solution is improvable. In the case of minimization, this disturbance is performed only if the new tour is shorter than the current tour. So, this process is repeated as long as there is a possible improvement or other stopping criteria not met.

Tour construction algorithm considers TSP as a set of cities to be stacked to form a cycle; however, tour improvement algorithms see TSP as a set of tours to be perturbed to find the optimum configuration.

From the second layer, TSP is shown as a space of feasible solutions or permutations. So, before passing to the rest of layers, we will give a detailed description of the TSP space and how all techniques of these layers are performed while seeing TSP as combinatorial optimization problem.

4 TSP Solution Space

Most of combinatorial optimization problems from practical or theoretical fields, engineering or computer science, are designed to guide research toward a solution, combination or an optimal configuration of a set of variables in the context of some constraints. The aim of any COP is to find an object from a finite set or infinitely countable. This object is typically an integer, subset or a graph structure [18].

When we talk about COPs, designing a preference model for the solutions is not enough to deal with how to find the optimum because the set of all alternatives (feasible solutions) is defined in understanding. And its size prevents any explicit enumeration. Therefore, simple approaches like dynamic programming and greedy algorithms are not extended naturally with the presence of complex preferences. The greater part of the studied optimization problems in combinatorial context is NP-complete. While resolving such problems [19], there exist no effective algorithms. However, search algorithms try to overcome the constraint of complexity with more leniently concerning achieving optimality. They intelligently look for good solutions even if they are not optimal but found in an acceptable runtime [20]. Therefore, designing a tuned model for the solution space is a crucial part, for helping search algorithms to make significant moves. It decides how solutions will be placed and connected and how the search process will be performed.

4.1 Search Space Model

Most of combinatorial optimization problems like traveling salesman [7] and others [21–26] consider their feasible solutions as integer (or binary) series when they are solved approximately by a search algorithm. The main characteristic of any solution of the search space is its position (combination of discrete variables) which is associated to a value reflecting its fitness. All the neighbor solutions have connections between each other. These connections are the operation needed to change position from the current solution to a neighbor one. So, our search space is shown in Fig. 2 as a graph, and the connections between solutions are the arcs of the graph.

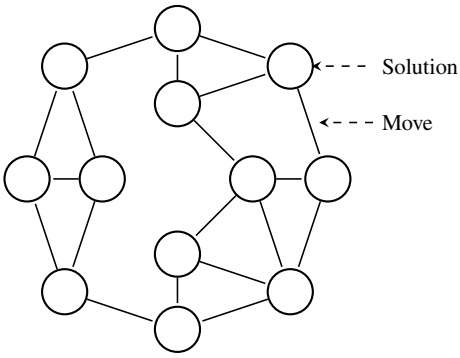


Fig. 2 Combinatorial solution space

Moving from a given solution to another one is the result of changing the coordinates (the cities order) of this solution to generate a new one. To show more description about how the combinatorial search space is designed, we will discuss its main components which are solution (or position), move and neighborhood.

4.1.1 Solution

In the search space, solutions are considered as the main component which is composed of variables (coordinates) related to the appropriate model of the treated problem (TSP in our case) [27]. While TSP search space is combinatorial, the coordinate variables of each solution are discrete (Fig. 3). The solution fitness is controlled by an objective function and affected by changes over the coordinate variables [28, 29].

In the TSP combinatorial space, Fig. 3 shows a solution that is composed of cities c_i ordered with respect to a permutation π. Consequently, two solutions are different if the visit order of their cities is different, regardless if the departure cities are the same or not.

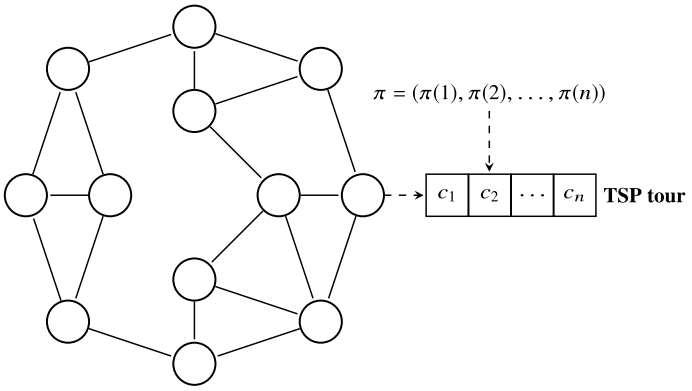


Fig. 3 Solution in the search space

4.1.2 Moves

In TSP, the coordinates of a given solution are changed according to their characteristics. Generally, changing the solution position is done by a new combination or permutation controlled by a group of techniques called moves [30]. They are all based on edge-exchange or node-insertion [31].

In this section, we will list the most known families from the simplest to advanced moves. The discussion about their performance and how they are used is developed in the following.

Two adjacent cities exchange
Exchanging two adjacent cities in the current tour is among the simplest moves in TSP. This operator (Fig. 4) is characterized by generating a relatively reduced number of new possible solutions (size of the neighborhood) that is equal to N [32].

Two adjacent cities exchange is a poor operator in terms of searching efficiency. The perturbation is on just two adjacent cities, so the search is very slow, and it cannot find solutions out of its small search region. Replacing two adjacent cities (N exchanges) by any pair of cities (N^2 exchanges) is turning short configurations into long ones but still have no effective searches [33].

2-opt move
Croes [34] has introduced, first 2-opt move. Otherwise, Flood [35] has before suggested its basic move."2-optimal" or 2-opt is the most famous change/move, for TSP solutions [33]. Two arcs are removed in the tour, reversing one of the resulting sub-tours and reconnects the two newly created paths differently, as shown in Fig. 5. The worst-case complexity for searching with one move is $O(n^2)$.

3-opt move
In 3-opt [36, 37] (Fig. 6), the exchange replaces up to three edges of the current tour. The three resulting paths are put together in a new way, possibly reversing one or more of them. The complexity for searching the neighborhood defined by 3-opt is $O(n^3)$. 3-opt is also considered as the special case of 1-shift [38].

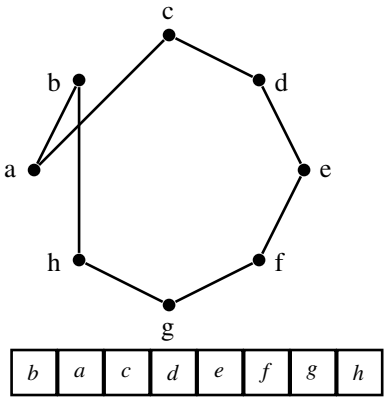


Fig. 4 Two adjacent cities exchange

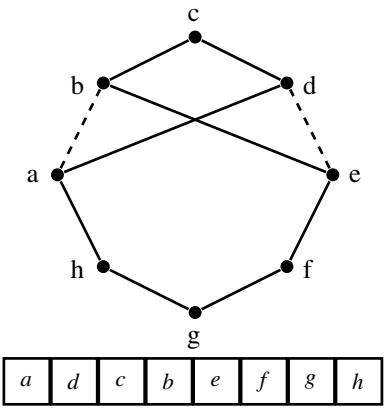


Fig. 5 2-optimal move

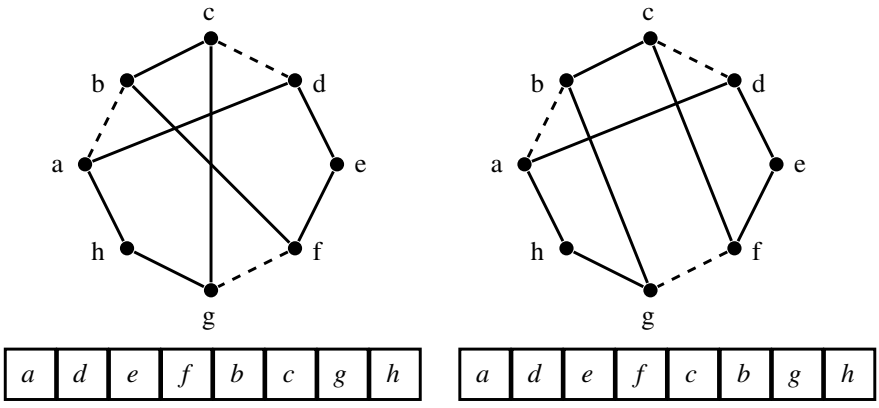


Fig. 6 Example of 3-optimal possible moves

2.5-opt move

A variant between 2-opt and 3-opt moves proposed by Bentley [39] to reduce the complexity of 3-opt procedure is called 2.5-opt. It expands 2-opt neighborhood by including a part from 3-opt moves. 2.5-opt move changes the position of one city to its new position elsewhere between two neighbor cities in the tour. In Fig. 6, the part used from 3-opt moves is when cities *b* and *c* are considered as one city [32] (Fig. 7).

Double-bridge move

4-opt is a move where up to four edges are changed in the tour. Another variant of the 4-opt move, that is considered as a simplified move thanks to its non-sequential nature, is the double-bridge move as shown by Lin and Kernighan [40] (Fig. 8).

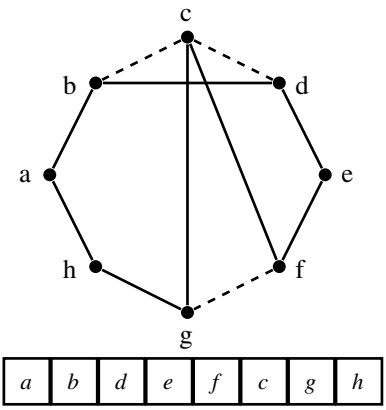


Fig. 7 2.5-optimal move

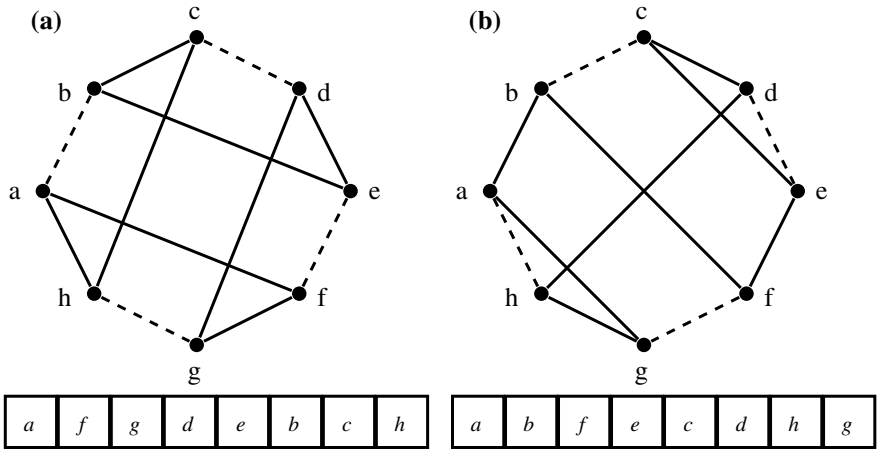


Fig. 8 **a** Double-bridge and **b** 4-optimal move

Other moves

k-opt moves for $k > 2$ are a generalized case of 2-opt move. It can be performed by a finite sequence of 2-opt moves [12]. A step is the distance between two solutions. It is related to the space topology and the neighborhood notion. Steps are generally classified according to their lengths which highly depend on how the perturbation is applied and its iterated application on the initial solution position.

Crossover and mutation moves are performed as an uncontrolled random jump that diverse the search, which is not adapted to intensify the search to promising regions. Mutation, as a unary operator, is depending on the chosen neighborhood of the current solution. The new explored neighborhood is reached by replacing a

predefined number of edges in the current solution in a way the mutation avoids returning to the past position by the subsequent local search.

While crossover operator is binary, it has been developed to perform special "jumps" within the search space of local minima. Crossover considers the solution space of TSP as a "globally convex" or "big valley", such that it tries to find the optimal solution near to the center of the valley of near-optimum solutions [41]. So, the neighborhood structure or any information about the incorporated topology might be helpful [42].

4.1.3 Neighborhood

The search space is composed of connections or arcs between solutions. This connection is the only condition to define the neighborhood notion in a given space topology. The notion of neighborhood in combinatorial spaces requires that a neighbor solution is the one that is generated by one canonical change (small steps) on the current solution. Regions are then specified by a group of solutions that are distanced with one canonical move in the same space topology (or neighborhood).

When approching the optimal solution, considering neighborhood moves is important to keep searching in this promising region. All search methods have to control and balance intensification and diversification [43] described later. So, to perform this balance, we need to differentiate neighbor solutions from distant solutions.

To stay and exploit in a given region, intensification uses small steps to move from one position to another. It has to keep searching in the neighborhood of the current solution. To do this, we have to control the step size and the width of the neighborhood.

Some of the proposed k-exchanges are used to represent a small step and small neighborhood and some others for big jumps and large neighborhood.

For example, 3-opt explores the space much more than 2-opt. However, the size of the neighborhood is larger and consumes more time to search. Also, "non-sequential" exchanges for double-bridge move are considered as big jump out of the neighborhood of the current solution [30].

Neighborhood is also measuring distance between solutions in the space. A neighbor solution is a solution which are not far from the current solution. The distance between two TSP tours can be simply in terms of their differences in structure. If tours T and T′ have differences in k links, then the distance is k. That is related to k-opt moves [44].

The cardinality of neighborhood or its size is not the one thing that matters while studying neighborhoods. Reachability of the solutions is also an important parameter. Some tours of the TSP are not reachable from each other if we consider some neighborhood structures [12]. The aim of designing neighborhoods is to be able to balance between intensify the search in the neighborhood of the current solution and a diversification out of this neighborhood by producing a big jump. This jump can be performed by a sequence of small steps, with no improvement, or by changing the structure of the neighborhood (topology) by changing the move.

4.2 Constraints

One of the important reasons behind studying neighborhood and distances between solutions is to adapt the steps to the variation of the objective function values. The minimum change on a solution must produce a small change on the solution quality. We can consider a good neighborhood structure as the one that avoids containing the optimum and a bad solution in the same region.

The main constraint while solving combinatorial optimization problems is the relation between objective function and topology of the search space. A small move from one current solution in its neighborhood does not imply that the difference, in terms of fitness, between those solutions will be necessarily small.

5 Search Methods

Local search methods for combinatorial optimization proceed by performing a sequence of local changes in an initial solution which improve each time its fitness until a local optimum is found. But, evaluating the neighborhood of a solution can slow the search process if all possible moves in this neighborhood are visited. Among many techniques, local search methods choose the first improving move to reach the local optimum faster. To make the improving move, some methods evaluate solution by solution sequentially or a limited number of evaluations on solutions chosen randomly [45].

The previously described class of methods can be grouped in the second layer. However, the process of optimization is not just iteratively improving the current solution, with local search methods, until finding the optimal solution. It is also having the ability of escaping the local optimum which is the distinguishing characteristic of the third layer. Here, the relation between layers is complementary. Methods of layer i serve those of $i + 1$, and methods of layer $i + 1$ complete those of i.

Search methods are typically incomplete when the local search process is stopped even if the solution found is not optimal. A good search method is not only the one that can find the local minimum. It has to be able to change the neighborhood when a local optimum is found to reach the best solution of the search space.

To complete the search process of a local search method, a variable-k neighborhood search can be performed. It solves TSP by iteratively improving a solution with k-change moves, while k is 2 or 3. In this example, the local search can be guided by 2 or 3-opt moves and "double-bridges" for a big jump [46–48]. This kind of methods is viewed as variable depth methods, since the type of moves carried out at each iteration is dynamically determined, and varies from one iteration to the next. Changing the moves here is to search in a different space topology to avoid reversing the search and so returning to the previously visited regions.

Methods such as Taboo Search (TS) [49], in order to overcome this local minimum problem, implement in many ways a taboo list. To avoid recycling, it changes its

current position to a new neighbor solution whatever its solution nearby fitness, which decreases the quality of the current solution. On the other hand, simulated annealing (SA) [50] chooses randomly a neighbor solution. Each new solution is accepted according to its fitness and the temperature parameter which decreases the new solution acceptance probability.

There are other single-solution algorithms that have partial differences in the way they follow to escape the local minimum, such as Iterated Local Search (ILS) [51], Variable Local Search (VLS) [52], Guided Local Search (GLS) [53] and many others. Layer three contains methods that escape the local minimum with the help of two principal techniques. The first is changing the topology of the search space by performing more than one move. And the second generally keeps the same move but displaces even if the new found solution is not better than the current one.

Population-based methods considered also as another way to handle the problem of local optimum. They improve, during iterations, a population of solutions. The main characteristic of these methods is to move to another level of research with more diversity and cooperation between individuals or across generations in the population.

In order to find a high-quality solution in a reduced runtime, different approximate methods are proposed. Among these methods, the most effectives are metaheuristics [54–56]. Indeed, metaheuristics have proven their robustness to solve several optimization problems. Most of metaheuristics begin with an initial solution (or population) and try to reach the optimum solution by an improvement process. At each step, they change the combination of the current solution to generate an improved one and move to the new generated solution. It is a strategic research to more effectively explore the search space, with often focus on promising regions.

As the main method of the fourth layer, to find the optimum, metaheuristics are in principle based on the notion of intensification and diversification, in order to select the best possible solutions that are around the current solution, or randomize the search to regions far from the last visited solution. Metaheuristics need to keep exploring new regions while encouraging investment in promising regions over a limited period of time. Generally, intensification and diversification must be balanced intelligently and dynamically [57].

Whatever simple improvement algorithm or an intelligent metaheuristic, they look at TSP as a space of solutions. They do not care how the solution is built as much as how to iteratively move from the current position (solution) to a new one much better. Therefore, every information about this space and its topology enhance the chance to move effectively and find out the optimum as quickly as possible. An example of this information is to differentiate between exploring and exploiting moves and, in a specific topology, between a small and big steps.

In the case of population-based methods, how individuals communicate, share position of good solutions is an important characteristic of fourth layer methods. Sharing information and experiments is not just between individuals of the same population, and it can pass to individuals of the next generation.

6 Example: Discrete Cuckoo Search

To illustrate the layer model, we take as example, discrete cuckoo search (DCS) for the traveling salesman problem developed by Ouaarab et al. [47]. It shows how the solution passes from the first layer to the fourth one and how each layer complements the others.

6.1 First Layer: Construct a Solution

In this layer, DCS generates a solution randomly. It begins from an initial city and randomly adds the next one to the tour until the last city. The result is a permutation as follows (Fig. 9).

6.2 Second Layer:Improving the Solution

DCS has used 2-opt move in its improvement method. The idea is to repeat this move, while the improvement is possible. There exist several ways to implement this technique. Among them, we cite the first improvement or the best improvement method. In DCS, they performed the best improvement after m move evaluations [58] (Fig. 10).

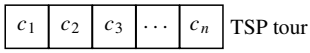


Fig. 9 Random TSP solution

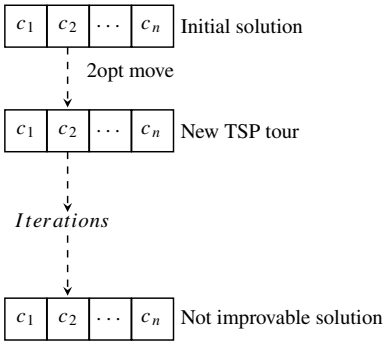


Fig. 10 2-opt improvement method

6.3 Third Layer: Local Optimum Escaping Methods

In this important layer, DCS has implemented two techniques to escape from the local optimum. The first is double-bridge move Fig. 8 which is a perturbation move characterized by its big jump in the 2-opt neighborhood topology. And the second is a sequential applications of 2-opt moves on the current solution without testing improvement and accept the generated solution. When the improvement process is blocked in a local optimum, DCS alternates between double-bridge and sequential 2-opt moves regarding the value generated by Lévy flight distribution [58].

6.4 Fourth Layer: Discrete Cuckoo Search

The fourth layer in DCS is how it controls the other layer techniques. It uses a population of initial solutions created thanks to the first layer. Each population individual improves its fitness by implementing 2-opt improvement method. Then, the third layer is used by DCS to help individuals to escape from the local optimum. The population structure [47] is performed also to select the best individuals in order to share their experiences with others, and the worsts are removed and replaced by new ones.

In this layer, we can see how DCS searches intelligently for the optimum by using the main technique of metaheuristics which is intensification and diversification. It mainly exploits by using improvement method and sharing experience between population individuals and explores by escaping from the local optimum and replacing the worst individuals.

7 Conclusion

Among operational research problems, TSP is one of the most studied, transformed and solved. The main two discussed ideas in the present chapter are to highlight the principal components of TSP space of solutions and in the other side, show up how methods dealing with TSP approximately form four groups that can be hierarchically presented as layers.

For the first layer methods, the search space components are the cities or edges between cities. They try iteratively to search in this space the next best component to build the solution. After having the built solution of the first layer, the second layer presents the search space as a graph of solutions and moves. The third layer has the permission to change the topology (type of move) of the solution search space. The final layer methods search in the space much more intelligently and can detect effectively promising regions.

This analytic study has chosen TSP because of its reduced number of constraints which relatively facilitate the representation of the search space and the moves between the feasible solutions.

This model can be useful to design generic methods that try to solve TSP dynamically by simply generating a combination of four layer methods. They will, at each layer, take the well-adapted method to the treated instance and complete the process by passing to the next layer until the final one. Following this perspective, it is possible to extend the proposed concept to solve the variants of TSP and problems that can be transformed to it, with a minimum number of generic methods.

References

1. Johnson DS, Garey MR (1979) Computers and intractability: a guide to the theory of NP-completeness. WH Freeman
2. Pardalos LY, Resende MGC (1994) A greedy randomized adaptive search procedure for the quadratic assignment problem. Quad Assign Relat Problem DIMACS Seri Discr Math Theoret Comput Sci 16:237–261
3. Applegate DL, Bixby RE, Chvatal V, Cook WJ (2006) The traveling salesman problem: a computational study. Princeton University Press
4. Stützle T, Dorigo M et al (1999) ACO algorithms for the traveling salesman problem. In: Evolutionary algorithms in engineering and computer science, pp 163–183
5. Donald D (2010) Traveling salesman problem, theory and applications. INTECH Open Access Publisher
6. Lenstra JK, Kan AHGR (1975) Some simple applications of the travelling salesman problem. Oper Res Quart, 717–733
7. Lawler EL, Lenstra JK, Kan AHGR, Shmoys DB (1985) The traveling salesman problem: a guided tour of combinatorial optimization, vol 3. Wiley, Chichester
8. Har-Peled S (2010) Approximating the euclidean traveling salesman problem (TSP) (Chapter 10)
9. Laporte G (1992) The traveling salesman problem: an overview of exact and approximate algorithms. Eur J Oper Res 59(2):231–247
10. Gerhard R (1994) The traveling salesman: computational solutions for TSP applications. Springer
11. Larranaga P, Kuijpers CMH, Murga RH, Inza I, Dizdarevic S (1999) Genetic algorithms for the travelling salesman problem: a review of representations and operators. Artifi Intell Rev 13(2):129–170
12. Gutin G, Punnen AP (2002) The traveling salesman problem and its variations, vol 12. Springer
13. Garey MR, Johnson D (1979) Computers and intractability: a guide to the theory of NP-completeness. In: Computer animation conference
14. Papadimitriou CH (1977) The euclidean travelling salesman problem is NP-complete. Theoret Comput Sci 4(3):237–244
15. Zhang Z, Schwartz S, Wagner L, Miller W (2000) A greedy algorithm for aligning DNA sequences. J Comput Biol 7(1–2):203–214
16. Rosenkrantz DJ, Stearns RE, Lewis PM II (1977) An analysis of several heuristics for the traveling salesman problem. SIAM J Comput 6(3):563–581
17. Johnson DS (1990) Local optimization and the traveling salesman problem. In: Automata, languages and programming. Springer, pp 446–461
18. Papadimitriou CH, Steiglitz K (1998) 6.1 the max-flow, min-cut theorem. Combinatorial optimization: algorithms and complexity, Dover, pp 120–128

19. Sanjeev A, Boaz B (2009) Computational complexity: a modern approach. Cambridge University Press
20. Parker RG, Rardin RL (2014) Discrete optimization. Elsevier
21. Lawler EL (1963) The quadratic assignment problem. Manage Sci 9(4):586–599
22. Martello S (1990). Knapsack problems: algorithms and computer implementations. Wiley-interscience series in discrete mathematics and optimization
23. Pinedo ML (2012) Scheduling: theory, algorithms, and systems. Springer Science & Business Media
24. Laporte G (1992) The vehicle routing problem: an overview of exact and approximate algorithms. Eur J Oper Res 59(3):345–358
25. Wolsey LA, Nemhauser GL (2014) Integer and combinatorial optimization. Wiley
26. Schrijver A (2003) Combinatorial optimization: polyhedra and efficiency, vol 24. Springer Science & Business Media
27. Fredman ML, Johnson DS, McGeoch LA, Ostheimer G (1995) Data structures for traveling salesmen. J Algorithm 18(3):432–479
28. Ouaarab A, Yang X-S (2016) Cuckoo search: from cuckoo reproduction strategy to combinatorial optimization. In: Nature-inspired computation in engineering. Springer, pp 91–110
29. Ouaarab A, Ahiod B, Yang X-S, Abbad M (2015) Random-key cuckoo search for the quadratic assignment problem. Nat Comput (submitted)
30. Voudouris C, Tsang E (1999) Guided local search and its application to the traveling salesman problem. Eur J Oper Res 113(2):469–499
31. Babin G, Deneault S, Laporte G (2007) Improvements to the or-opt heuristic for the symmetric travelling salesman problem. J Oper Res Soc 58(3):402–407
32. Johnson DS, McGeoch LA (1997) The traveling salesman problem: a case study in local optimization. Local Search Combin Opt 1(1):215–310
33. Kirkpatrick S, Toulouse G (1985) Configuration space analysis of travelling salesman problems. J Phys 46(8):1277–1292
34. Croes GA (1958) A method for solving traveling-salesman problems. Oper Res 6(6):791–812
35. Flood MM (1956) The traveling-salesman problem. Oper Res 4(1):61–75
36. Bock F (1958) An algorithm for solving travelling-salesman and related network optimization problems. Oper Res 6:897–897 (Inst operations research management sciences 901 Elkridge Landing RD, STE)
37. Lin S (1965) Computer solutions of the traveling salesman problem. Bell Syst Techn J 44(10):2245–2269
38. Bianchi L, Knowles J, Bowler N (2005) Local search for the probabilistic traveling salesman problem: correction to the 2-p-opt and 1-shift algorithms. Eur J Oper Res 162(1):206–219
39. Bentley JJ (1992) Fast algorithms for geometric traveling salesman problems. ORSA J Comput 4(4):387–411
40. Lin S, Kernighan BW (1973) An effective heuristic algorithm for the traveling-salesman problem. Oper Res 21(2):498–516
41. Freisleben B, Merz P (1996) A genetic local search algorithm for solving symmetric and asymmetric traveling salesman problems. In: Proceedings of IEEE international conference on evolutionary computation. IEEE, pp 616–621
42. Merz P, Freisleben B (1997) Genetic local search for the TSP: new results. In: IEEE International Conference on Evolutionary Computation (Icec'97), pp 159–164, IEEE
43. Lozano M, García-Martínez C (2010) Hybrid metaheuristics with evolutionary algorithms specializing in intensification and diversification: overview and progress report. Comput Oper Res 37(3):481–497
44. Mak K-T, Morton AJ (1995) Distances between traveling salesman tours. Discr Appl Math 58(3):281–291
45. Kernighan BW, Lin S (1970) An efficient heuristic procedure for partitioning graphs. Bell System Techn J 49(2):291–307
46. Martin O, Otto SW, Felten EW (1992) Large-step markov chains for the tsp incorporating local search heuristics. Oper Res Lett 11(4):219–224

47. Ouaarab A, Ahiod B, Yang X-S (2014) Discrete cuckoo search algorithm for the travelling salesman problem. Neural Comput Appl 24(7–8):1659–1669
48. Boese KD (1995) Cost versus distance in the traveling salesman problem. UCLA Computer Science Department Los Angeles
49. Fred G, Manuel L (1997) Tabu search, 1997. Kluwer Academic Publishers
50. Kirkpatrick S, Gelatt CD, Vecchi MP (1983) Optimization by simulated annealing. Science 220(4598):671
51. Lourenço HR, Martin OC, Stutzle T (2001) Iterated local search. arXiv preprint math/0102188
52. Branke J (2001) Evolutionary approaches to dynamic optimization problems-updated survey. In: GECCO workshop on evolutionary algorithms for dynamic optimization problems, pp 27–30
53. Balas E, Vazacopoulos A (1998) Guided local search with shifting bottleneck for job shop scheduling. Manage Sci 44(2):262–275
54. Blum C, Roli A (2003) Metaheuristics in combinatorial optimization: overview and conceptual comparison. ACM Comput Surv (CSUR) 35(3):268–308
55. Glover F, Kochenberger GA (2003) Handbook of metaheuristics. Springer
56. El-Ghazali T (2009) Metaheuristics: from design to implementation, vol 74. Wiley
57. Yang X-S (2012) Swarm-based metaheuristic algorithms and no-free-lunch theorems. INTECH Open Access Publisher
58. Ouaarab A, Ahiod B, Yang X-S. Improved and discrete cuckoo search for solving the travelling salesman problem. In: Cuckoo search and firefly algorithm. Springer, pp 63–84

Chapter 5
Flow Shop Scheduling By Nature-Inspired Algorithms

M. K. Marichelvam, Ömür Tosun and M. Geetha

1 Introduction

Scheduling problems were addressed by several researchers for the past many decades based on their theoretical and practical impacts. Scheduling is one of the critical decision-making tasks in our daily life. In scheduling, limited resources are effectively allocated to meet certain objective functions over time [4, 53]. Most of the scheduling problems were known to be non-deterministic polynomial time hard (NP-hard) type combinatorial optimization problems. As it is difficult to solve and find an optimal solution in a reasonable time to these problems, several nature-inspired algorithms were proposed to find near-optimal solutions to them. These algorithms were developed by aping the behavior of various animals, birds, insects, etc. In this chapter, ant colony optimization, African wild dog algorithm, artificial bee colony (ABC), bacterial foraging optimization algorithm (BFOA), bat algorithm (BA), cuckoo search (CS), crow search algorithm, firefly algorithm, flower pollination algorithm, fruit fly optimization algorithm, gray wolf optimization algorithm, invasive weed optimization algorithm, migrating birds optimization algorithm, monkey search algorithm, particle swarm optimization algorithm, rhinoceros search algorithm, sheep flock heredity algorithm, shuffled frog leaping algorithm, water wave optimization algorithm and whale optimization algorithm are considered in this

M. K. Marichelvam (✉)
Department of Mechanical Engineering, Mepco Schlenk Engineering College, Sivakasi, Tamil Nadu 626005, India
e-mail: mkmarichelvamme@gmail.com

Ö. Tosun
Department of Management Information Systems, Faculty of Applied Sciences, Akdeniz University, Antalya, Turkey

M. Geetha
Department of Mathematics, Kamaraj College of Engineering and Technology, Virudhunagar, Tamil Nadu 626001, India

X.-S. Yang and Y.-X. Zhao (eds.), *Nature-Inspired Computation in Navigation and Routing Problems*, Springer Tracts in Nature-Inspired Computing,
https://doi.org/10.1007/978-981-15-1842-3_5

chapter. The next section will present the flow shop scheduling problem. Section 3 will illustrate the literature review on the various algorithms to solve the flow shop scheduling problems. Then, the future scope will be addressed in Sect. 4. Finally, Sect. 5 will conclude the section.

2 Problem Definition

Flow shop environment can be seen in different production environments like automotive, electronics, furniture and others. A flow shop comprises a group of *m* machineries aligned in sequences, and *n* jobs are to be ordered [23]. Every job should be worked on the existing machines in a pre-determined specific job order. A job is first handled to machine 1, following to machine 2 and at last finished on the final machine *m*. The working period of the jobs is known before which are fixed and nonnegative. It is also estimated that the jobs are accessible at time zero. Each of the machines can process only one job at a given time, and each of the jobs can only be processed on one specific machine at a given time. The processing activity of each job cannot be disturbed, which is, preemption is not permitted. Finally, it is also known that the machines are existing through the complete scheduling time period (known as no machine breakdown). A simple flow shop scheduling environment is given in Fig. 1.

The permutation flow shop scheduling (PFSS) problem, which can be characterized as $\alpha/\beta/\gamma$, and in this notation, the first term refers to the machine environment (flow shop), the second term is the processing restrictions or constraints (permutation), and the third term is the objective to be minimized (makespan or total flow time).

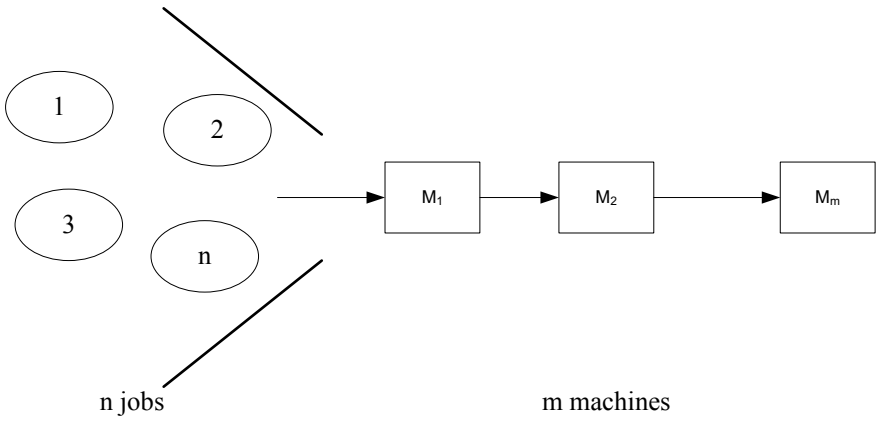


Fig. 1 Layout of flow shop scheduling environment

In the flow shop problem, there are n jobs are to be processed on m-machines. All jobs must be processed on every machine in the same given sequence. For the makespan or total flow time approach, the aim is to find best the permutation of jobs to be valid for each machine that minimizes the given objective function.

For F_m/prmu/$C_{\max}$ problem, $C(j_i, k)$ denotes the completion time of job j_i on machine k, $\pi = (j_1, j_2, \ldots, j_n)$ denotes a permutation of all jobs, and $p_{i,j}$ is the process time of job i on machine j. The completion time for an n-job, m-machine flow shop problem is calculated as follows [41]:

$$C(j_1, 1) = p_{j_1,1} \tag{1}$$

$$C(j_i, 1) = C(j_{i-1}, 1) + p_{j_1,1}, \quad i = 2, \ldots, n \tag{2}$$

$$C(j_1, k) = C(j_1, k-1) + p_{j_1,k}, \quad k = 2, \ldots, m \tag{3}$$

$$\begin{aligned} C(j_i, k) = {} & \max\{C(j_{i-1}, k), C(j_i, k-1)\} \\ & + p_{j_i,k}, \quad i = 2, \ldots, n; \quad k = 2, \ldots, m \end{aligned} \tag{4}$$

$C_{\max}$ is the makespan that can be defined as the completion time of the last job in the manufacturing system. Makespan can be calculated as

$$C_{\max}(\pi) = C(j_n, m) \tag{5}$$

So, the PFSS problem with the makespan criterion is to find a permutation π^* in the set of all possible permutations Π such that

$$C_{\max}(\pi^*) \leq C(j_n, m) \quad \forall \pi \in \Pi \tag{6}$$

Let $F(j_i)$ be the flow time of job j_i and is equal to the completion time $C(j_n, m)$ of job j_i on machine m since ready times are zero. Total flow time of a permutation π can be found by summing or the flow times or completion times of the jobs (TFT(π)). Therefore, total flow time can be found as

$$\mathrm{TFT}(\pi) = \sum_{i=1}^{n} F(j_i) = \sum_{i=1}^{n} C(j_i, m) \tag{7}$$

So, the PFSS problem with total flow time criterion is to find a permutation π^* in the set of all permutations Π such that

$$\mathrm{TFT}(\pi^*) \leq \mathrm{TFT}(\pi) \quad \forall \pi \in \Pi \tag{8}$$

For earliness or tardiness objectives, due date of a job j is given by d_j. It is assumed that all jobs are ready at the starting point, and job preemption is not possible. Also

to prevent the processing from interruptions, there is enough buffer space between each successive machines. In the basic approach, there is no machine blocking and waiting time bottleneck between the following machines.

This kind of problem is given as $F_m/d_j/\sum E_j + T_j$. E_j is the earliness of job j and it is defined as the time by which the job j is finished before the due date d_j. T_j is the tardiness of the job j and it is defined as the time by which job j is finished later than the given due date d_j.

Objective of this kind problems is to find the best schedule that minimizes the $\sum_{j=1}^{n}\left(E_j + T_j\right)$, which is the sum of earliness and tardiness. Here, $E_j = \max\left\{0, d_j - C_j\right\}$ and $T_j = \max\left\{0, C_j - d_j\right\}$.

3 Literature Review

This section presents a detailed literature review on various nature-inspired algorithms and their variants to tackle the flow shop scheduling problems with a variety of objective functions.

3.1 Ant Colony Optimization (ACO)

The main idea of the ACO was the pheromone trail used by real ants as a basis for communication and feedback between each of them while searching for a food source [12]. Simple agents as ants are used in this population-based algorithm. The construction of solutions is guided by (artificial) pheromone trails and problem-specific heuristic information. The algorithm starts with an individual ant and its pheromone trail. If more ants used the same way, the intensity of the pheromone (chance of selection of that food source (or solution)) increases. If a food source is selected by less ants, its pheromone is evaporated (meaning the probability of that solution's selection is decreased). Usually, different local search-based improvements are added to the structure of the algorithm. Rajendran and Ziegler [59] developed two different ACO for PFSP with total flow time. Comparing their algorithm against another ACO (which is said to have given the best solutions up to that time), their improvements give better solutions for the benchmark problems. Gajpal et al. [15] proposed an ant colony algorithm to minimize the makespan in SDST flow shop scheduling problems. A new ACO algorithm is developed, and its performance is compared against an existing ACO and two other heuristics from the literature. It is seen that the new algorithm gives better solutions against the others. Rajendran and Ziegler [58] have handled the problem of scheduling in permutation flow shops by utilizing ACO algorithms using the objective of minimizing machine total flow time and makespan. The efficiency of the proposed ant colony optimization algorithm has been evaluated by considering the benchmark problems. Rajendran and Ziegler [57]

proposed multi-objective ACO to produce non-dominated solution with the objective of minimizing makespan and total flow time in permutation flow shop scheduling. Yagmahan and Yenisey [84, 85] applied multi-objective ant colony system algorithm to minimize the both objectives of makespan and total flow time in flow shop scheduling. Reported results are better in terms of the performance solution when compared to other heuristics.

3.2 African Wild Dog Algorithm

This metaheuristics is based on the communal (or social) hunting behavior of African wild dogs which live in packs of up to 20 adults and their dependent young. By moving as a group instead of individuals, the chance of getting a hunt (or prey) increases. For an optimization problem, each dog is accepted as a solution of a problem, whereas prey is the solution of the related problem. If a wild dog moves near its prey, the solution improved, which is mostly known as local search [74]. Marichelvam and Geetha [40] solved the PFSP with makespan using African wild dog algorithm. It is stated the algorithm is a very simple which only have two control parameters, whereas most of the metaheuristics have several. Using difference benchmark problems, algorithm is compared against heuristic approaches and gives better results than them.

3.3 Artificial Bee Colony (ABC)

The artificial bee colony (ABC) algorithm consists of three kinds of bees called employed bees, onlooker bees and scout bees which simulates the foraging actions of a real bee colony. In the basic ABC algorithm, candidate solution of an optimization problem is represented with the position of a food source, and the nectar amount of a food source corresponds to the profitability (fitness) of the related solution. Employed bees are assigned for exploiting the nectar sources discovered before and giving this information to the waiting bees (onlooker bees) in the hive. This information is about the quality of the food source positions which they have exploited. Scout bees either randomly search the environment in order to find a new food source depending on an internal motivation or based on possible external clues. Each food source is harnessed by only one employed bee. So, in the ABC algorithm, the number of employed bees is equal to the number of food sources existing around the hive.

There are three basic tasks in the ABC algorithm repeated until a termination criterion is met:

1. Each employed bee is assigned to a food source, and their nectar amounts are measured.

2. Employed bees return to their hive and share the pre-gained knowledge about food sources (quality of the solution) with onlooker bees. According to this information, onlooker bees select a food source and explore around it.
3. If a food source is not improved for a pre-determined number of trials, abandon that source, and its employed bee becomes a scout bee. Produce a random food source for the scout bee.

Tasgetiren et al. [78] applied a simple artificial bee colony algorithm in permutation flow shops with the objective of the total flow time. ABC is hybridized with different variants of iterated greedy algorithm. Also, a hybrid differential evolution algorithm based on local search is given. Tasgetiren et al. [76] applied discrete artificial bee colony (DABC) algorithm to minimize total tardiness for no-idle permutation flow shop scheduling with different due date. They tested DABC algorithm with the benchmark problems. They reported that the performance of DABC algorithm is better than GA. Liu and Liu [34] developed a hybrid discrete artificial bee colony algorithm to minimize the makespan in permutation flow shop scheduling problems. In this method, initially greedy randomized adaptive search procedure (GRASP) based on heuristics used to generate the initial population, and then, new solution produced from the distinct operators and algorithm such as insert, swap, path relinking and GRASP for the employed bees. Also, this hybrid algorithm was tested with benchmark problems. Ribas et al. [61] introduced a high-performing model for the blocking flow shop problem under total flow time criterion. Different food searching strategies are introduced in the study. By using a design of experiment, best strategies are selected. Using this new algorithm gives better results in the test problems against the other comparing algorithms.

Han et al. [19] hybridized ABC with differential evolution (DE) to solve flow shop scheduling problem with blocking. Using test instances, the solutions show that the proposed methodology is superior to the compared algorithms for minimizing the makespan criterion. Gong et al. [17] developed an ABC algorithm for blocking lot-streaming flow shop scheduling problem with a multi-objective approach: solving for makespan and earliness time simultaneously. The modification to the basic ABC is justified using a variety of test problems, and the performances of the proposed algorithm are evaluated and compared with four different algorithms on 18 test subsets. The experimental results justify the significance of the approach. The results also show that the proposed algorithm significantly outperforms the other algorithms in terms of convergence and diversity of non-dominated solutions.

3.4 Bacterial Foraging Optimization Algorithm (BFOA)

As a nature-inspired optimization algorithm, BFOA is based on a of group foraging strategy of a swarm of E. coli bacteria. To survive in a harsh environment, bacteria tend to search for nutrients in a manner to maximize energy obtained per unit time. Individual bacterium shares their knowledge between them by communicating with

others by sending signals. A bacterium takes foraging decisions after evaluating two previous factors. The process, in which a bacterium moves by taking small steps while searching for better nutrients, is called chemo taxis, and key idea of BFOA is mimicking chemotactic movement of virtual bacteria in the problem search space [52]. Shivakumar and Amudha [72] developed enhanced bacterial foraging algorithm for solving permutation flow shop scheduling problems with the objective of makespan. They tested the algorithm with benchmark problems and reported that the improvement of hybrid bacterial swarming algorithm. Zhao et al. [92] adopted a chaotic local search based bacterial foraging optimization (CLSBFO) to minimize makespan in permutation flow shop scheduling problem. In order to improve its effectiveness, they used DE operator and the chaotic search operator into the chemotactic step of the basic BFO. Also, they compared the results of CLSBFO with based bacterial optimization and particle swarm optimization.

3.5 Bat Algorithm (BA)

The basics of bat algorithm are based on the echolocation or bio-sonar characteristics of Microbats [86]. Microbats detect their preys, avoid obstacles and locate their roosting places in the dark using a type of sonar called echolocation. In this process, bats emit a very loud sound pulse and listen for its echo that bounces back from the surrounding objects. Their pulses vary in properties and can be correlated with their hunting strategies, depending on the species. Most bats use short, frequency-modulated signals to sweep through about an octave, while others more often use constant-frequency signals for echolocation [80]. Tosun and Marichelvam [80] developed a hybrid bat algorithm to minimize the makespan in permutation flow shop scheduling problems. They tested the hybrid algorithm with Reeves benchmark problems. They compared the performance solutions of the hybrid algorithm with other metaheuristics algorithms. They conducted the statistical analysis to validate the results. Xie et al. [83] presented a differential Lévy-flights bat algorithm to solve the permutation flow shop with makespan objective function.

3.6 Cuckoo Search (CS)

CSA is developed from the obligate brood parasitic behavior of some cuckoo species in combination with the Levy flight behavior of some birds and fruit flies in nature [88]. The following three idealized rules are considered for describing the CS.

1. Cuckoo eggs correspond to the solution of the problem. Each cuckoo lays one egg at a time and dumps its egg in a randomly chosen nest.
2. The best nests with high-quality eggs (solutions) will carry out to the next generation.

3. The egg laid by a cuckoo can be discovered by the host bird with a probability, and a nest will then be built. This helps the chance of creating a new unknown solution alternative to the problem, and the better solutions are replacing the worse solutions.

Marichelvam [42] improved the basic CS algorithm by including NEH heuristic to the initial solution generation step. Using benchmark problems for PFSP problem with makespan criterion, developed approach gives better results than the ACO algorithm. Li and Yin [30] developed a CS approach to the PFSP. First, they used largest-ranked-value rule to convert the continuous nature of the CS to discrete optimization. To further improve the performance, NEH heuristic and a fast local search techniques are integrated with the basic CS. Makespan criterion is used, and performance comparison gives the superiority of the proposed approach. Dasgupta and Das [10] proposed an inter-species cuckoo search (ISCS) algorithm for hybrid flow shop scheduling (HFS) and permutation flow shop sequencing problems. Smallest position value (SPV) heuristic [5] is used to convert the continuous nature of the cuckoo search algorithm to discrete optimization. Both makespan and mean flow time objectives are used. In general, proposed algorithm has better solution quality against other metaheuristics given in the study. The proposed CS algorithm gives better results than the rest. Wang et al. [82] developed a new cuckoo search (NCS) algorithm for solving flow shop scheduling problems. They used four strategies. The first strategy was changing continuous nature of the algorithm into discrete job permutations using smallest position value (SPV) rule. The second strategy was to generate initial solutions generated with high quality using NEH heuristic. The third strategy was to modify generalized opposition-based learning to improve the convergence speed. In the fourth strategy, local search was used to enhance the exploitation. They tested NCS algorithm with the benchmark problems. They reported NCS algorithm produced better results than standard Cuckoo search.

In Komaki et al. [26], two different CS algorithms are given. Three-stage assembly flow shop scheduling problem based on makespan minimization is handled. Both models are compared against SA, VNS and different dispatching rules. Wang et al. [81] developed a CS algorithm and measure its performance with Taillard test instances. In the approach, smallest position value (SPV) is preferred for changing the continuous solutions to discrete variables. Like the other studies, heuristics is also used to enhance the initial solution quality. To further improve the proposed approach, generalized opposition-based learning and a local search strategy are utilized. Results showed that performance obtained is better than the traditional CS and other meta-heuristic algorithms selected for comparison. Sun and Gu [75] hybridized CS with the estimation of distribution algorithm (EDA). No-idle PFSP is handled under total tardiness criterion. Like most of the previous studies, NEH heuristics [42] is used to increase the efficiency of the initial solution. Some crossover and local search-based strategies are also included in the hybrid model. Using benchmark problems, algorithm is compared against GA, CS and EDA, in which it gives better results than each of them.

3.7 Crow Search Algorithm (CSA)

CSA is a population-based nature-inspired algorithm which copies the behavior of crow birds and their social interaction [3]. Living in group, crows are intelligent birds which hide their foods in hiding places and also able to memorize their places and retrieve the hidden food even after several months. Crows can do thievery by following the other crows by watching their food's hiding places and steal the hidden food. When a crow feels that another one is following it, it moves to another place far away from the food's hiding place in order to confuse the thief crow. Flow shop scheduling environment is analyzed by Marichelvam and Geetha [38]. Basic CSA is hybridized with dispatching rules to improve the efficiency of the solutions. Minimizing the sum of makespan and total flow time is the objective of the given study. Random test instances are generated, and solutions are compared against other algorithms. In all the situations, the proposed methodology gives better solutions.

3.8 Firefly Algorithm (FA)

FA is a swarm intelligent-based metaheuristics based on the characteristic behaviors of fireflies. Fireflies use the flashing light for finding mates, attracting their potential prey and protecting themselves from their predators. The swarm of fireflies will move to brighter and more attractive locations by the flashing light intensity that associated with the objective function of problem considered in order to obtain better optimal solutions [89]. The development of FA is based on three rules:

(i) Artificial fireflies are assumed to be unisex so that gender is not an issue for attraction.
(ii) Attractiveness is proportional to flashing brightness which decreases as the distance from the other firefly increases due to the fact that the air absorbs light. As the attractiveness increases, the brightest firefly becomes the most attractive firefly in the swarm, to which it convinces neighbors moving toward. In case of no brighter one, it freely moves any direction
(iii) The brightness of the flashing light can be considered as objective function of a problem that needs to be optimized.

Sayadi et al. [67] proposed discrete firefly algorithm for minimizing the makespan in permutation flow shop scheduling problems. They formulated the problem as a mixed-integer programming. They compared the results of discrete firefly algorithm with ant colony optimization algorithm and then reported that discrete firefly algorithm produced better results when compared to the ant colony for some well-known benchmark problems. Chakaravarthy et al. [8] compare firefly algorithm and artificial immune system algorithm for the lot-streaming flow shop scheduling problems. Using test instances, they measured algorithms' efficiency using makespan. Both algorithms showed great performance but according to the study FA is better than

other algorithms. Lo et al. [35] improved the performance of FA by using the logistic map which shows a chaotic behavior for initializing the population. PFSP with makespan criterion is solved, and performance of the chaos-based FA is compared with basic FA, PSO and GA. Marichelvam et al. [36] solved multi-objective FSSP with FA algorithm. Objective of the scheduling problem is the weighted sum of makespan, mean flow time, mean tardiness and number of tardy jobs. In the study, smallest position value is used to convert the continuous nature of the traditional FA to discrete nature. Using randomly generated test problems, algorithm is compared against SA, ACO, GA, CS and PSO. FA gives better results than the corresponding metaheuristics. Marichelvam and Geetha [39] proposed a FA to solve flow shop scheduling problem to minimize the total flow time. The basic FA is improved with different heuristic approaches. Comparing its results against GA, ACO and SA, FA has produced far better solutions.

3.9 Flower Pollination Algorithm (FPA)

FPA is a swarm intelligence-based metaheuristics proposed by Yang [87]. Its essential is behavior of flowers' pollination. Basic idea behind the algorithm is

- Each flower is handled as an individual in a population
- Cross-pollination is conducted based on a probability of $p \in [0, 1]$. Here, cross-pollination is based on the animals (like bees or butterflies) pollination among different kinds of flowers using the Levy flight mode.
- Self-pollination, on the other hand, is the close distance pollination among the same species of the flowers. In the algorithm, it has a probability of $(1 - p)$ to be conducted.

Qu et al. [54] improved the cross-pollination mechanism of the basic FPA with hormone modulation mechanism and used this algorithm to solve the no-wait FSSP under makespan. Comparing their approach against other metaheuristics from the literature showed that proposed methodology gives better solutions against them.

3.10 Fruit Fly Optimization Algorithm (FFO)

The fruit fly optimization algorithm (FOA) mimics the food-finding behavior of the fruit fly. The fruit fly is better to other species in sensing and perception, especially in osphresis and vision. The osphresis organs of fruit flies can find all kinds of scents floating in the air; it can even smell food source from more than 40 km away. Then, after it gets close to the food location, it can also use its sensitive vision to find food and the company's flocking location and fly toward that direction too [50].

Marichelvam et al. [37] solved multi-objective FSSP with FFO algorithm. Objective of the scheduling problem is the weighted sum of makespan, mean flow time,

earliness and tardiness. Authors include the constructive heuristics and a dispatching rule to improve the solution quality of the basic FFO. Performance of the model is compared with GA, ACO, SA, PSO and basic FFO. Proposed hybrid FFO gives better results for each test instance. FA is a swarm intelligent-based metaheuristics based on the characteristic behaviors of fireflies. Fireflies use the flashing light for finding mates, attracting their potential prey and protecting themselves from their predators. The swarm of fireflies will move to brighter and more attractive locations by the flashing light intensity that associated with the objective function of problem considered in order to obtain better optimal solutions. The development of FA is based on three rules:

3.11 Gray Wolf Optimization (GWO) Algorithm

Gray wolf optimization is a nature-based swarm intelligent method which mimics the leadership hierarchy of wolves that are well known for their group hunting. In their natural environment, gray wolfs tend to live in a pack. They have a strict social dominant structure in which their leader is called alpha. The alpha is generally responsible for decision making. The orders of the dominant wolf should be followed by the pack. The betas are subordinate wolves which help the alpha in decision making. The beta is an advisor to alpha and discipliner for the pack. The lower-ranking gray wolf is omega which has to submit all other dominant wolves. If a wolf is neither an alpha or beta nor omega, then it is called delta. Delta wolves dominate omega and report to alpha and beta [47]. Jeet [22] solved the two-machine flow shop scheduling problem under makespan and idle time criteria. In the proposed multi-objective approach, the processing time and setup times are based on fuzzy environment. Performance of the proposed approach is compared against multi-objective GA, PSO and NSGA-II. Better solutions are obtained with the new approach. Komaki and Kayvanfar [25] solved the two-stage flow shop scheduling problem with release time for minimizing the makespan. GWO algorithm is developed, and performance is measured. Also, the effect of different dispatching rules on the problem sets is measured. Yang and Liu [90] developed a hybrid multi-objective gray wolf optimization algorithm to solve the fuzzy blocking flow shop scheduling problem.

3.12 Invasive Weed Optimization (IWO) Algorithm

Invasive weed optimization is a bio-inspired stochastic optimization algorithm that basically simulates natural behavior of weeds in colonizing and finding suitable place for growth and reproduction [43]. Some of the distinctive properties of IWO in comparison with other evolutionary algorithms are the way of reproduction, spatial dispersal and competitive exclusion [95]. Zhou et al. [95] proposed an IWO algorithm for PFSS problem under makespan. They compared their findings against PSO and

hybrid GA, and the simulation results show that the best scheduling sequence can be obtained effectively by IWO which showed a strong adaptive and robustness. Sang and Pan [62] solved the FSSP with intermediate buffers under makespan. NEH heuristic, an improved search mechanism and a local search strategy, is integrated into the basic IWO. Comparing their results against other metaheuristics showed the effectiveness of their methodology. Sang et al. [63] improved the IWO with NEH heuristic and local search techniques for the no-wait lot-streaming FSSP using test problems. Comparing their results against different metaheuristics from the literature proved their algorithms efficiency. Sang et al. [64] developed a two-stage IWO for distributed assembly PSFP with total flow time criterion. Both product permutations and job sequences are optimized in the study. Heuristic-based initializing, job-based and product-based local search strategies are used to have better solution quality by the IWO. Solutions are compared against two different memetic algorithm-based metaheuristics [29, 33]. Improved IWO gives better results against each comparing algorithm.

Jafarzadeh et al. [20] designed a multi-objective IWO for no-wait two-stage flexible flow shop scheduling problem under makespan and average lateness time. First, a Taguchi experiment is given to find the optimal control parameters. Three other multi-objective algorithms based on PSO, GA and evolutionary strategy are used to compare the proposed model. Test results indicated the better performance of the multi-objective IWO. Sang et al. [65] used IWO for the lot-streaming FSSP with SDST with makespan. NEH heuristic is used to increase the quality of the initial solutions, and different local search and global search-based improvements are proposed. A comparative performance measurement is studied with using different well-performing metaheuristics including randomly generated 280 test problems. The experimental simulations show that the proposed IWO approach gives exceptionally better results than the competing algorithms. In Shao et al. [70], a multi-objective IWO is used for the blocking FSSP for simultaneously minimizing the makespan and total tardiness. Using two different heuristics, quality of the initial population is improved, also to further enhance the solution quality a reference line-based reproduction, a sliding insertion-based spatial dispersal and local search strategies in included to basic IWO. Comparing the performance against different metaheuristics from the literature indicated that proposed approach produced better results. Shao et al. [68] improved IWO using a random insertion-based searching approach and shuffle-based local search strategy. Blocking flow FSSP with makespan is used to measure the efficiency of the approach. Shao et al. [69] also proposed the IWO to solve the blocking flow shop scheduling problems with multiple objectives. Sang et al. [66] presented three different IWO algorithms for distributed assembly PFSP with total flow time criterion. Different neighborhood operators and local search strategies are integrated into the basic IWO. In those three IWO-based methodologies, the one with hybrid search operators gives better solution quality. Comparing it to other metaheuristics also confirms its performance superiority.

3.13 Migrating Birds Optimization (MBO) Algorithm

Duman et al. [13] propose a new nature-inspired metaheuristics algorithm technique based on the V flight formation of the migrating birds which is proven to be an effective formation in energy saving. Algorithm is started with initial solutions. Afterward, these solutions are tried to improve on each step. The MBO uses a neighborhood structure to improve solutions. Also, it has a benefit mechanism that unused neighbor solutions shared with other solutions. Further, the leader bird is replaced with following bird. This is applied to left side of the flock firstly. The leader bird is gone to end of left side, and following bird is assigned as new leader. The flock is continued same formation as far as flapping parameter. Then, the leader bird is replaced with right side of the flock. Gao et al. [16] used MBO for solving the no-wait flow shop scheduling with total flow time criterion. To improve the efficiency of the algorithm, three heuristics are used for population initialization, and an effective neighborhood structure is integrated. Proposed algorithm gives better solutions against some well-known metaheuristics used for comparison. Tongur and Ülker [79] solved flow shop sequencing problem for minimizing the makespan. Using some basic benchmark problems, performance of the algorithm is measured. Meng et al. [44] designed an enhanced MBO for the lot-streaming flow shop scheduling problem with setup times, in which job-splitting and job scheduling are considered simultaneously. The objective of the study is to minimize the makespan. In the study, a two-stage vector is employed to represent solutions in the swarm. Also, a special neighbor structure is designed to further improve the local search. Finally, to prevent to be stuck around to local minimum, a new solution update scheme is included. Comparing the results against different metaheuristics showed that the enhanced MBO has better solution quality.

Benkalai et al. [6] proposed two different MBO algorithms for the permutation flow shop scheduling with SDST under makespan. First one is the basic MBO, and the second is improved MBO based on or-opt neighborhood approach and also using different heuristics for the initial solutions. The results showed that the improved MBO is superior to the basic one. Meng et al. [45] improved the basic MBO for solving the lot-streaming FSSP. In their approach, lot-splitting and job scheduling are optimized simultaneously. Aim of the study is minimizing the makespan. To improve the basic MBO, a harmony search-based schema is used to develop neighborhood solutions. Another improvement is based on a leaping mechanism to prevent the algorithm trapped in a local optimum. Different metaheuristics from the literature are used to compare the proposed approach's performance. Sioud and Gagne [73] proposed both a heuristic and MBO-based solution approach for PFSP with sequence-dependent setup times (SDST) under makespan. MBO is improved using tabu list, a local search procedure and a probabilistic leader bird selection strategy. Both the new heuristic and the improved MBO give better results than the comparing algorithms. Han et al. [18] improved the basic MBO algorithm to solve the blocking lot-streaming flow shop problem under makespan. Their proposed algorithm uses multiple neighborhoods based on insert and swap operators, estimation of distribution algorithm (EDA)

and also a local search based on the insert neighborhood strategy. They compared their results with the existing discrete invasive weed optimization, EDA and MBO algorithms based on different benchmark problems. Results showed the superiority of the given algorithm.

3.14 Monkey Search Algorithm

Monkey search algorithm is emerged from the idea of the monkeys and their mountain-climbing process. In the algorithm, the climb process used is a kind of recursive optimization technique. The proposed climbing process is used for improving objective function value (or position of a monkey). During the climbing process, each monkey arrives its own mountain top. Then, each monkey will start looking for other higher points than its current position. If a higher mountain is found (a better solution to the problem), the monkey could jump there by using its eyesight property. The eyesight of a monkey is known as the maximum distance that a monkey could watch. The current position of a monkey is updated. In the third step, the new searching domains can be found by the monkeys based on their current positions as a starting point. This stage is referred as somersault process and will yield a new position for the monkeys [93]. Marichelvam et al. [41] developed hybrid monkey search algorithm for solving the flow shop scheduling problem with the objective of minimizing makespan and total flow time. They tested the algorithm with benchmark problems. They reported hybrid monkey search algorithm produced better results when compared to other methods.

3.15 Particle Swarm Optimization (PSO)

PSO was developed by Kennedy and Eberhart [24], inspired by the behavior of social living organisms in groups, such as group of birds and fish or ant colonies. This algorithm emulates the interaction between members to share information. PSO performs its search of the optimal solution through agents, known as particles, whose trajectories are adjusted by a stochastic and a deterministic component. Each particle is influenced by its 'best' achieved position and the group 'best' position but tends to move randomly. A particle is defined by its position vector and its velocity vector. Position of each particle represents its fitness function value. For every iteration, each particle changes its position according to the new velocity based on the history of both its own current position and best location (fitness) with those of the swarm with some random perturbation. Rahimi-Vahed and Mirghorbani [56] developed multi-objective particle swarm optimization for minimizing the weighted mean completion time and weighted mean tardiness in flow shop scheduling. They concluded that for large-sized problem, the proposed PSO performed better when compared to GA. Tasgetiren et al. [77] proposed PSO algorithm to solve permutation flow shop sequencing problem

with the objective of makespan and total flow time. They applied smallest position value (SPV) rule to PSO algorithm to both Watson's benchmarks for makespan and Taillard benchmark instances for total flow time. They reported that the performance of proposed method produced better solution for most of the sequencing problems.

Jarboui et al. [21] applied Combinatorial Particle Swam Optimization (CPSO) for solving permutation flow shop problem with the objective of minimizing the makespan and the total flow time. The authors compare the performance of CPSO to the smallest position value of PSO (PSOspv), GA and Hybrid CPSO (H-CPSO) for makespan. Similarly, they compare the performance of H-CPSO algorithm to PACO, M-MMAS and Variable Neighborhood Search PSO (PSOVNS) for total flow time. Finally, they report the performance of H-CPSO better for total time up to 20 jobs and machine varying from 5 to 20. PSOVNS produces better result for jobs 50, and machine varies from 5 to 20. Lian et al. [31] developed novel particle swarm optimization (NPSO) algorithm to solve permutation flow shop scheduling problem with the objective of minimizing makespan. They tested algorithm with Taillard benchmark problems. They compared the results of NPSO with standard genetic algorithm, and then, they concluded NPSO performed well. Chen et al. [9] proposed revised discrete particle swarm optimization (RDPSO) to solve the permutation flow shop scheduling problem with the objective of minimizing the makespan. They developed a new filtered local search to filter the solution regions and guide the search to new solution regions in order to avoid premature convergence. The RDPSO algorithm was tested with the benchmark problem sets. They reported RDPSO algorithm performed better than existing PSO algorithms. Lian et al. [32] proposed an improved PSO based on crossover and mutation operators for the FSSP to minimize makespan. Solutions compared against GA show that the improved methodology has a better performance on the test instances.

3.16 Rhinoceros Search Algorithm (RSA)

The RSA is based on the daily life routines of rhinoceros' groups foraging and other natural activities. Global exploration and local intensification of a search algorithm is represented with different genders in a group of rhinoceros. Deb et al. [11] proposed some assumptions for the algorithm to work efficiently:

- All rhinoceros are doing Levy flight in their search procedure.
- Male rhinoceros have bounce mechanism, while female rhinoceros do not have this property.
- In each iteration, with a probability of 0.05, any rhinoceros would die. Therefore, a reborn mechanism is developed to ensure the stability of the population.

Deb et al. [11] proposed a novel approach based on the rhinoceros' natural behaviors. PFSP with makespan and total flow time are used with different benchmark problems are solved with RSA. Algorithm is compared against PSO. For makespan criterion, RSA gives better solution than the PSO, whereas for total flow time criterion, RSA becomes better with the increasing problem dimension.

3.17 Sheep Flock Heredity Algorithm (SFHA)

Basic idea of the SFHA is the behaviors of sheep within two different flocks. Normally, sheep are living together with their own flock under the supervision of their shepherds. Based on this closed society, the genetic heritage can only occur inside the flock, in other words, some special attributes in one flock develop only within this flock by heredity, and the sheep with high fitness properties (objective function of a problem) to their environment breed in the flock [48]. When two different sheep flocks were encountered and mixed together, shepherd of the corresponding flocks run into the mixed flock, and try to separate their own sheep as before. However, shepherds cannot distinguish between their original sheep they owned because of the same appearance of each sheep. Therefore, it becomes possible that several sheep from one of the flocks can inevitably mix with the other flocks. This random occurrence can enhance property of each flock. The characteristics of the sheep in the neighboring flocks can be inherent to the sheep in other flocks with this process. After each flock is separated in the field, the flock of the sheep having better fitness characteristics to the field environment breeds most (improves the chance of having a better solution quality) [60]. Anandaraman [2] proposed an improved sheep flock heredity algorithm (ISFHA) to solve the job shop and flow shop scheduling problems. Using different benchmark problems, makespan criterion is used for performance measurement. ISFHA shows better results in both problem types. Chakaravarthy et al. [7] proposed a SFHA and ABC algorithm for the lot-streaming flow shop problem under makespan and total time criteria. They have improved the basic SFHA using pairwise mutation process and a robust-replace heuristic for the neighborhood search process. Both algorithms are compared against PSO and DEA. It is given that improved SFHA gives better quality results for makespan and total flow time, compared with ABC. Chakaravarthy et al. [7] compared an improved SFHA against ABC algorithm for lot streaming in m-machine flow shop scheduling. Two different objectives as total flow time and makespan are used in the study. To improve the basic SFHA, a single mutation process is introduced instead of pairwise mutation process, and robust-replace heuristic is used to enrich the neighborhood search process. Proposed approach showed better results than the ABC algorithm.

3.18 Shuffled Frog Leaping Algorithm (SFLA)

Inspired by the leaping process of frog groups in a swamp, the swamp has a number of stones (possible solutions of a problem) at discrete locations. Each frog wants to find the best stone (the one with the maximum amount of food) with their leaping ability. As social beings, the frogs can allow to communicate with each other so that they can improve their memes (a unit of cultural evolution) using another frogs' information. Improvement in a meme results changing an individual frog's position to be optimum by altering the leaping steps of each frog. In the proposed approach, the changing of a leaping step is only allowed to be a discrete value by correspondence with the frogs and their discrete positions. To improve the search quality, it is required that frogs with better positions contribute more to the development of new ideas than frogs with poor positions [14]. Pan et al. [49] used SFLA for solving the lot-streaming flow shop scheduling problem under makespan criterion. Two different situations known as no-idling and idling cases are used. To have a better solution, quality NEH heuristic is used for the initialization schema, and a local search strategy is included to improve the algorithm's efficiency. In both problem types, proposed algorithm shows better solution quality. Rahimi-Vahed and Mirzaei [55] solved multi-objective (weighted mean completion time and weighted mean tardiness) PFSS problem. A multi-objective SFLA is proposed, and its performance is compared against other three multi-objective metaheuristics based on GA. The computational results show that the proposed SFLA performs better than the genetic algorithms, especially for the large-sized problems.

Pan et al. [51] converted the basic SFLA to solve the multi-objective no-wait flow shop scheduling in which makespan, maximum tardiness and total flow time are the criteria used. A NEH-based initialization, two-point crossover-based procedure and a local search techniques are combined to improve the basic algorithm. Better solutions are gained against the comparing metaheuristics. Lei and Guo [28] used SFLA for the HFSP with two agents. The selected objective is the total tardiness of the both agents. Random test instances are generated, and the performance of the proposed approach is measured against GA and SA. SFLA gives better solutions quality against the selected algorithms. In Lei and Tan [27], order acceptance and scheduling problem are considered in a flow shop where the objective is to decide on the orders to accept and then schedule the accepted orders in order to maximize total net revenue. A SFLA approach based on tournament selection is developed for the problem. Using randomly generated test instances, algorithm's performance is compared against ABC and SA. It is given that the proposed SFLA performs better than the ABC and SA.

3.19 Water Wave Optimization (WWO) Algorithm

WWO method is inspired by shallow water wave models borrows ideas from wave motions controlled by the wave-current-bottom interactions for solving optimization problems. Wave propagation, breaking and refraction phase are the three steps of the algorithm. Each of the alternative solution is represented as a wave with height and wave length. In wave propagation, the wave is propagated to a random position. If it reached to a lower sea depth (a better fitness value), it breaks into solitary waves which are formed in the breaking phase. This phase is the exploitation step of the algorithm by generating random solitary waves around the current best position. On the other hand, in refraction phase, algorithm searches for any other best solutions to avoid from the local optimal [91, 94]. In Shao et al. [70], *n*-job *m*-machines blocking FSSP with SDST is considered. Both heuristic and WWO-based metaheuristics solution techniques are given in the study. Makespan is the objective function used, performance of the proposed heuristics is compared against 14 heuristics, and metaheuristics approach is compared against 23 different metaheuristics from the literature. Statistical analysis showed that the both solution techniques developed gives better solution quality against the comparing algorithms. Shao et al. [71] developed a multi-objective WWO algorithm for the multi-objective blocking FSSP to simultaneously minimize the makespan and total flow time. Basic WWO is redesigned to cope with a multi-objective problem. These improvements are a decomposition-based initializing strategy, a ranking-based propagation operator for solution improvement and some local search techniques. Performance of the algorithm is evaluated against 14 different metaheuristics which showed that the multi-objective approach outperforms each of them.

3.20 Whale Optimization Algorithm (WOA)

WOA is inspired by hunting mechanism humpback whales. Algorithm consists three phases: encircling prey (solution initializing), bubble-net attacking method (exploitation phase) and search for prey (exploration phase) [46].

Abdel-Basset et al. [1] proposed a WOA hybridized with a local search strategy for solving PFSS problems with makespan. Different benchmark instances are used to evaluate the performance of the proposed algorithm.

4 Future Direction of Research

From the above literature review papers, it is evident that a wide variety of flow shop scheduling problems was solved by different objective functions. However, there is a wide scope for expanding this research in several ways. First, most of the

researchers dealt with only theoretical problems by considering the flow benchmark problem instances. It would be interesting to consider the scheduling problems of real industries and solve them by these algorithms. As the industries ought to deliver their products in time, this would be useful to the companies. The second important development of this research is to apply the above nature-inspired algorithms to solve multi-objective and many objective scheduling problems. As of now, most of the researchers considered only single objective function only. The third scope may be applying these nature-inspired algorithms to solve the flow shop scheduling problems with uncertainties. Parameter optimization would be other interesting scope of this research. As the performance of the nature-inspired algorithms would depend on the parameters, optimization of these parameters is an important one. Hybridization of the proposed algorithms with other suitable algorithms to obtain the optimal or near-optimal solutions with less computational time is another vital scope of this research work.

5 Conclusions

In this chapter, a comprehensive review is presented to focus the nature-inspired algorithms proposed to solve the flow shop scheduling problems which was developed during 1950s. The algorithms and their variants are discussed in detail in this study. The future scope of the research on the nature-inspired algorithms is also addressed. It is observed that a wide variety of nature-inspired algorithms is addressed in the literature. Though researchers have made many diversifications in the problem settings, they made numerous assumptions to simplify the problems. However, in real production scheduling problems, one cannot make such assumptions. Another important conclusion from the current study is that most of the researchers used some heuristics and or dispatching rules to generate some of the initial solutions to obtain the enhanced results. Also, they used the smallest position value rule to apply the algorithms to solve the discrete scheduling problems. Taillard's benchmark problems were widely addressed in the literature to validate the performance of the proposed algorithms. Very few researchers provided the computational time comparison. Though excellent results were obtained by some of the algorithms, it cannot be concluded that one specific algorithm is better than other algorithms as the algorithms depend on randomness. Hence, the research can be extended in many directions as discussed above.

References

1. Abdel-Basset M, Manogaran G, El-Shahat D, Mirjalili S (2018) A hybrid whale optimization algorithm based on local search strategy for the permutation flow shop scheduling problem. Future Gener Comput Syst 85:129–145
2. Anandaraman C (2011) An improved sheep flock heredity algorithm for job shop scheduling and flow shop scheduling problems. Int J Ind Eng Computations 2(4):749–764
3. Askarzadeh A (2016) A novel metaheuristic method for solving constrained engineering optimization problems: Crow search algorithm. Comput Struct 169:1–12
4. Baker KR (1974) Introduction to sequencing and scheduling. Wiley, New York
5. Bean JC (1994) Genetic algorithms and random keys for sequencing and optimization. ORSA J Comput 6(2):154–160
6. Benkalai I, Rebaine D, Gagné C, Baptiste P (2017) Improving the migrating birds optimization metaheuristic for the permutation flow shop with sequence-dependent set-up times. Int J Prod Res 55(20):6145–6157
7. Chakaravarthy GV, Marimuthu S, Ponnambalam SG, Kanagaraj G (2014) Improved sheep flock heredity algorithm and artificial bee colony algorithm for scheduling m-machine flow shops lot streaming with equal size sub-lot problems. Int J Prod Res 52(5):1509–1527
8. Chakaravarthy GV, Marimuthu S, Sait AN (2012) Comparison of firefly algorithm and artificial immune system algorithm for lot streaming in m-machine flow shop scheduling. Int J Comput Intell Syst 5(6):1184–1199
9. Chen CL, Huang SY, Tzeng YR, Chen CL (2014) A revised discrete particle swarm optimization algorithm for permutation flow-shop scheduling problem. Soft Comput 18(11):2271–2282
10. Dasgupta P, Das S (2015) A discrete inter-species cuckoo search for flowshop scheduling problems. Comput Oper Res 60:111–120
11. Deb S, Tian Z, Fong S, Tang R, Wong R, Dey N (2018) Solving permutation flow-shop scheduling problem by rhinoceros search algorithm. Soft Comput 22(18):6025–6034
12. Dorigo M, Di Caro G (1999) Ant colony optimization: a new meta-heuristic. In: Proceedings of the 1999 congress on evolutionary computation-CEC99 (Cat. No. 99TH8406), vol 2, pp 1470–1477. IEEE
13. Duman E, Uysal M, Alkaya AF (2011) Migrating birds optimization: a new meta-heuristic approach and its application to the quadratic assignment problem. In: European conference on the applications of evolutionary computation. Springer, Berlin, Heidelberg, pp 254–263
14. Eusuff MM, Lansey KE (2003) Optimization of water distribution network design using the shuffled frog leaping algorithm. J Water Resour Plan Manage 129(3):210–225
15. Gajpal Y, Rajendran C, Ziegler H (2006) An ant colony algorithm for scheduling in flow shops with sequence dependent setup of jobs. Int J Adv Manuf Technol 30(5):416–442
16. Gao KZ, Suganthan PN, Chua TJ (2013) An enhanced migrating birds optimization algorithm for no-wait flow shop scheduling problem. In: 2013 IEEE symposium on computational intelligence in scheduling (CISched). IEEE, pp 9–13
17. Gong D, Han Y, Sun J (2018) A novel hybrid multi-objective artificial bee colony algorithm for blocking lot-streaming flow shop scheduling problems. Knowl-Based Syst 148:115–130
18. Han Y, Li J, Sang H, Tian T, Bao Y, Sun Q (2018) An improved discrete migrating birds optimization for lot-streaming flow shop scheduling problem with blocking. In: International conference on intelligent computing. Springer, Cham, pp 780–791
19. Han YY, Gong D, Sun X (2015) A discrete artificial bee colony algorithm incorporating differential evolution for the flow-shop scheduling problem with blocking. Eng Optim 47(7):927–946
20. Jafarzadeh H, Moradinasab N, Gerami A (2017) Solving no-wait two-stage flexible flow shop scheduling problem with unrelated parallel machines and rework time by the adjusted discrete multi objective invasive weed optimization and fuzzy dominance approach. J Ind Eng Manage 10(5):887–918
21. Jarboui B, Ibrahim S, Siarry P, Rebai A (2008) A combinatorial particle swarm optimization for solving permutation flow shop problems. Comput Ind Eng 54(3):526–538

22. Jeet K (2017) Fuzzy flow shop scheduling using grey wolf optimization algorithm. Indian J Sci Res 7(2):167–171
23. Johnson SM (1954) Optimal two and three stage production schedules with setup times included. Naval Res Logistics Q 1(1):61–68
24. Kennedy J, Eberhart RC (1995) Particle swarm optimization. In: Proceedings of the IEEE international conference on neural networks. IEEE Service Center, Piscataway, NJ, pp 1942–1948
25. Komaki GM, Kayvanfar V (2015) Grey wolf optimizer algorithm for the two-stage assembly flow shop scheduling problem with release time. J Comput Sci 8:109–120
26. Komaki GM, Teymourian E, Kayvanfar V, Booyavi Z (2017) Improved discrete cuckoo optimization algorithm for the three-stage assembly flowshop scheduling problem. Comput Ind Eng 105:158–173
27. Lei D, Tan X (2016) Shuffled frog-leaping algorithm for order acceptance and scheduling in flow shop. In: 2016 35th Chinese control conference (CCC). IEEE, pp 9445–9450
28. Lei D, Guo X (2015) A shuffled frog-leaping algorithm for hybrid flow shop scheduling with two agents. Expert Syst Appl 42(23):9333–9339
29. Li X, Ma S (2016) Multi-objective memetic search algorithm for multi-objective permutation flow shop scheduling problem. IEEE Access 4:2154–2165
30. Li X, Yin M (2013) A hybrid cuckoo search via Lévy flights for the permutation flow shop scheduling problem. Int J Prod Res 51(16):4732–4754
31. Lian Z, Gu X, Jiao B (2008) A novel particle swarm optimization algorithm for permutation flow-shop scheduling to minimize makespan. Chaos Solitons Fractals 35(5):851–861
32. Lian Z, Gu X, Jiao B (2006) A similar particle swarm optimization algorithm for permutation flowshop scheduling to minimize makespan. Appl Math Comput 175(1):773–785
33. Ling-Fang C, Ling W, Jing-jing W (2018) A two-stage memetic algorithm for distributed no-idle permutation flowshop scheduling problem. In: 2018 37th Chinese control conference (CCC). IEEE, pp 2278–2283
34. Liu YF, Liu SY (2013) A hybrid discrete artificial bee colony algorithm for permutation flowshop scheduling problem. Appl Soft Comput 13(3):1459–1463
35. Lo HL, Fong S, Zhuang Y, Wang X, Hanne T (2015) Applying a chaos-based firefly algorithm to the permutation flow shop scheduling problem. In: 2015 3rd international symposium on computational and business intelligence (ISCBI). IEEE, pp 51–57
36. Marichelvam MK, Prabaharan T, Geetha M (2015) Firefly algorithm for flow shop optimization. In: Recent advances in swarm intelligence and evolutionary computation. Springer, Cham, pp 225–243
37. Marichelvam MK, Azhagurajan A, Geetha M (2017) A hybrid fruit fly optimisation algorithm to solve the flow shop scheduling problems with multi-objectives. Int J Adv Intell Paradigms 9(2–3):164–185
38. Marichelvam MK, Geetha M (2018) A hybrid crow search algorithm to minimise the weighted sum of makespan and total flow time in a flow shop environment. Int J Comput Aided Eng Technol 10(6):636–649
39. Marichelvam MK, Geetha M (2016) A hybrid discrete firefly algorithm to solve flow shop scheduling problems to minimise total flow time. Int J Bio-Inspired Comput 8(5):318–325
40. Marichelvam MK, Geetha M (2013) Solving flowshop scheduling problems using a discrete African wild dog algorithm. ICTACT J Soft Comput 3(3):555–559
41. Marichelvam MK, Tosun Ö, Geetha M (2017) Hybrid monkey search algorithm for flow shop scheduling problem under makespan and total flow time. Appl Soft Comput 55:82–92
42. Marichelvam MK (2012) An improved hybrid Cuckoo Search (IHCS) metaheuristics algorithm for permutation flow shop scheduling problems. Int J Bio-Inspired Comput 4(4):200–205
43. Mehrabian AR, Lucas C (2006) A novel numerical optimization algorithm inspired from weed colonization. Ecol Inf 1(4):355–366
44. Meng T, Duan JH, Pan QK (2017) An enhanced migrating birds optimization for a lot-streaming flow shop scheduling problem. In: 2017 29th Chinese control and decision conference (CCDC). IEEE, pp 4687–4691

45. Meng T, Pan QK, Li JQ, Sang HY (2018) An improved migrating birds optimization for an integrated lot-streaming flow shop scheduling problem. Swarm Evol Comput 38:64–78
46. Mirjalili S, Lewis A (2016) The whale optimization algorithm. Adv Eng Softw 95:51–67
47. Mirjalili S, Mirjalili SM, Lewis A (2014) Grey wolf optimizer. Adv Eng Softw 69:46–61
48. Nara K, Takeyama T, Kim H (1999) A new evolutionary algorithm based on sheep flocks heredity model and its application to scheduling problem. In: IEEE SMC'99 conference proceedings. 1999 ieee international conference on systems, man, and cybernetics (Cat. No. 99CH37028), vol 6. IEEE, pp 503–508
49. Pan QK, Wang L, Gao L, Li J (2011) An effective shuffled frog-leaping algorithm for lot-streaming flow shop scheduling problem. Int J Adv Manuf Technol 52(5–8):699–713
50. Pan WT (2011) Fruit fly optimization algorithm. Tsang Hai Book Publishing Co., Taipei
51. Pan YX, Pan QK, Li JQ (2011) Shuffled frog-leaping algorithm for multi-objective no-wait flow-shop scheduling. Control Theor Appl 28(11):1363–1370
52. Passino KM (2002) Biomimicry of bacterial foraging for distributed optimization and control. IEEE Control Syst Mag 22(3):52–67
53. Pinedo M (2002) Scheduling: theory, algorithms, and systems. Prentice-Hall, New Jersey
54. Qu C, Fu Y, Yi Z, Tan J (2018) Solutions to no-wait flow shop scheduling problem using the flower pollination algorithm based on the hormone modulation mechanism. Complexity
55. Rahimi-Vahed A, Dangchi M, Rafiei H, Salimi E (2009) A novel hybrid multi-objective shuffled frog-leaping algorithm for a bi-criteria permutation flow shop scheduling problem. Int J Adv Manuf Technol 41(11–12):1227–1239
56. Rahimi-Vahed AR, Mirghorbani SM (2007) A multi-objective particle swarm for a flow shop scheduling problem. J Comb Optim 13(1):79–102
57. Rajendran C, Ziegler H (2009) A multi-objective ant-colony algorithm for permutation flow shop scheduling to minimize the makespan and total flow time of jobs, chapter: computational intelligence in flow shop and job shop scheduling, vol 230. Springer, Berlin, pp 53–99
58. Rajendran C, Ziegler H (2004) Ant-colony algorithms for permutation flowshop scheduling to minimize makespan/total flowtime of jobs. Eur J Oper Res 155(2):426–438
59. Rajendran C, Ziegler H (2005) Two ant colony algorithms for minimizing total flow time in permutation flow shops. Comput Ind Eng 48(4):789–797
60. Ramya G, Chandrasekaran M (2014) An evolutionary sheep flock heredity model algorithm for minimizing manufacturing cost in job shop scheduling. Int J Adv Mech Automobile Eng 1(1):16–20
61. Ribas I, Companys R, Martorell XT (2015) An efficient discrete artificial bee colony algorithm for the blocking flow shop problem with total flow time minimization. Expert Syst Appl 42(15):6155–6167
62. Sang HY, Pan QK (2013) An effective invasive weed optimization algorithm for the flow shop scheduling with intermediate buffers. In: 2013 25th Chinese control and decision conference (CCDC). IEEE, pp 861–864
63. Sang HY, Duan PY, Li JQ (2016) A discrete invasive weed optimization algorithm for the no-wait lot-streaming flow shop scheduling problems. In: International conference on intelligent computing. Springer, Cham, pp 517–526
64. Sang HY, Pan QK, Duan PY, Li JQ, Duan P (2017) A two-stage invasive weed optimization algorithm for distributed assembly permutation flowshop problem. In: 2017 Chinese automation congress (CAC). IEEE, pp 7051–7056
65. Sang HY, Pan QK, Duan PY, Li JQ (2018) An effective discrete invasive weed optimization algorithm for lot-streaming flowshop scheduling problems. J Intell Manuf 29(6):1337–1349
66. Sang HY, Pan QK, Li JQ, Wang P, Han YY, Gao KZ, Duan P (2019) Effective invasive weed optimization algorithms for distributed assembly permutation flowshop problem with total flowtime criterion. Swarm Evol Comput 44:64–73
67. Sayadi M, Ramezanian R, Ghaffari-Nasab N (2010) A discrete firefly meta-heuristic with local search for makespan minimization in permutation flow shop scheduling problems. Int J Ind Eng Computat 1(1):1–10

68. Shao Z, Pi D, Shao W, Yuan P (2019) An efficient discrete invasive weed optimization for blocking flow-shop scheduling problem. Eng Appl Artif Intell 78:124–141
69. Shao Z, Pi D, Shao W (2018) A multi-objective discrete invasive weed optimization for multi-objective blocking flow-shop scheduling problem. Expert Syst Appl 113:77–99
70. Shao Z, Pi D, Shao W (2018) A novel discrete water wave optimization algorithm for blocking flow-shop scheduling problem with sequence-dependent setup times. Swarm Evol Comput 40:53–75
71. Shao Z, Pi D, Shao W (2019) A novel multi-objective discrete water wave optimization for solving multi-objective blocking flow-shop scheduling problem. Knowl Based Syst 165:110–131
72. Shivakumar BL, Amudha T (2013) Enhanced bacterial foraging algorithm for permutation flow shop scheduling problems. ARPN J Eng Appl Sci 8:129–136
73. Sioud A, Gagné C (2018) Enhanced migrating birds optimization algorithm for the permutation flow shop problem with sequence dependent setup times. Eur J Oper Res 264(1):66–73
74. Subramanian C, Sekar ASS, Subramanian K (2013) A new engineering optimization method: African wild dog algorithm. Int J Soft Comput 8(3):163–170
75. Sun Z, Gu X (2017) Hybrid algorithm based on an estimation of distribution algorithm and cuckoo search for the no idle permutation flow shop scheduling problem with the total tardiness criterion minimization. Sustainability 9(6):953
76. Tasgetiren MF, Pan QK, Suganthan PN, Oner A (2013) A discrete artificial bee colony algorithm for the no-idle permutation flowshop scheduling problem with the total tardiness criterion. Appl Math Model 37(10–11):6758–6779
77. Tasgetiren MF, Liang YC, Sevkli M, Gencyilmaz G (2007) A particle swarm optimization algorithm for makespan and total flowtime minimization in the permutation flowshop sequencing problem. Eur J Oper Res 177(3):1930–1947
78. Tasgetiren MF, Pan QK, Suganthan PN, Chen AH (2011) A discrete artificial bee colony algorithm for the total flowtime minimization in permutation flow shops. Inf Sci 181(16):3459–3475
79. Tongur V, Ülker E (2014) Migrating birds optimization for flow shop sequencing problem. J Comput Commun 2(04):142
80. Tosun Ö, Marichelvam MK (2016) Hybrid bat algorithm for flow shop scheduling problems. Int J Math Oper Res 9(1):125–138
81. Wang H, Wang W, Sun H, Cui Z, Rahnamayan S, Zeng S (2017) A new cuckoo search algorithm with hybrid strategies for flow shop scheduling problems. Soft Comput 21(15):4297–4307
82. Wang J, Wang L, Shen J (2016) A hybrid discrete cuckoo search for distributed permutation flowshop scheduling problem. In: 2016 IEEE congress on evolutionary computation (CEC). IEEE, pp 2240–2246
83. Xie J, Zhou Y, Tang Z (2013) Differential lévy-flights bat algorithm for minimization makespan in permutation flow shops. In: International conference on intelligent computing. Springer, Berlin, Heidelberg, pp 179–188
84. Yagmahan B, Yenisey M (2010) A multi-objective ant colony system algorithm for flow shop scheduling problem. Expert Syst Appl 37(2):1361–1368
85. Yagmahan B, Yenisey MM (2008) Ant colony optimization for multi-objective flow shop scheduling problem. Comput Ind Eng 54(3):411–420
86. Yang XS (2010) A new metaheuristic bat-inspired algorithm. Nature inspired cooperative strategies for optimization (NICSO 2010). Springer, Berlin, Heidelberg, pp 65–74
87. Yang XS (2012) Flower pollination algorithm for global optimization. In: International conference on unconventional computing and natural computation. Springer, Berlin, Heidelberg, pp 240–249
88. Yang XS, Deb S (2009) Cuckoo search via Lévy flights. In: 2009 world congress on nature & biologically inspired computing (NaBIC). IEEE, pp 210–214
89. Yang XS (2008) Nature-inspired metaheuristic algorithms. Luniver Press, UK
90. Yang Z, Liu C (2018) A hybrid multi-objective gray wolf optimization algorithm for a fuzzy blocking flow shop scheduling problem. Adv Mech Eng 10(3):1687814018765535

91. Yun X, Feng X, Lyu X, Wang S, Liu B (2016) A novel water wave optimization based memetic algorithm for flow-shop scheduling. In: 2016 IEEE congress on evolutionary computation (CEC). IEEE, pp 1971–1976
92. Zhao F, Liu Y, Shao Z, Jiang X, Zhang C, Wang J (2016) A chaotic local search based bacterial foraging algorithm and its application to a permutation flow-shop scheduling problem. Int J Comput Integr Manuf 29(9):962–981
93. Zhao RQ, Tang WS (2008) Monkey algorithm for global numerical optimization. J Uncertain Syst 2(3):165–176
94. Zheng YJ (2015) Water wave optimization: a new nature-inspired metaheuristic. Comput Oper Res 55:1–11
95. Zhou Y, Chen H, Zhou G (2014) Invasive weed optimization algorithm for optimization no-idle flow shop scheduling problem. Neurocomputing 137:285–292

Chapter 6
Mobile Robot Path Planning Using a Flower Pollination Algorithm-Based Approach

Atul Mishra and Sankha Deb

1 Introduction

In today's era of modern robotics, mobile robots have been made as an intelligent vehicle which can perceive their work environment using modern sensors. These sensors can be used to provide the position and orientation and other data of the robot, which can be used to propose the path efficiently. From the sensor's data, the robot controller should be able to generate a safe and feasible path if it has to move from point A to point B. These sensors help make the robots autonomous which are nowadays widely used in several fields such as agriculture, manufacturing industries and transport. These mobile robots are equipped with several capabilities like perception, location, planning and navigation. Figure 1 displays the flow chart for the mobile robot navigation. Here, an attempt has been made to optimally plan the safe and feasible path for the mobile robot using a soft computing-based approach. In this problem, the robot has to navigate that path starting from point A (start point) to point B (end point) and also avoid the obstacles. These obstacles, for example, in a typical environment, could be machines, parts, pillar, etc. Several criteria like energy, time, distance, safety, etc. can be considered to make the path optimal [23]. Robot path planning problem can be categories mainly into two classes, local path planning and global path planning. The global path planning deals with the generation of a path when the working environment is static, and on the other hand, the local planning deals with the generation of a dynamic path considering the path obtained by the global path and the data obtained by the sensors.

From the literature study, it is learnt that the path planning approaches have been categorised into three, namely deliberative approaches, reactive approaches and

A. Mishra (✉) · S. Deb
Department of Mechanical Engineering, Indian Institute of Technology Kharagpur, Kharagpur 721302, India
e-mail: atulmishra.chdi@gmail.com

X.-S. Yang and Y.-X. Zhao (eds.), *Nature-Inspired Computation in Navigation and Routing Problems*, Springer Tracts in Nature-Inspired Computing,
https://doi.org/10.1007/978-981-15-1842-3_6

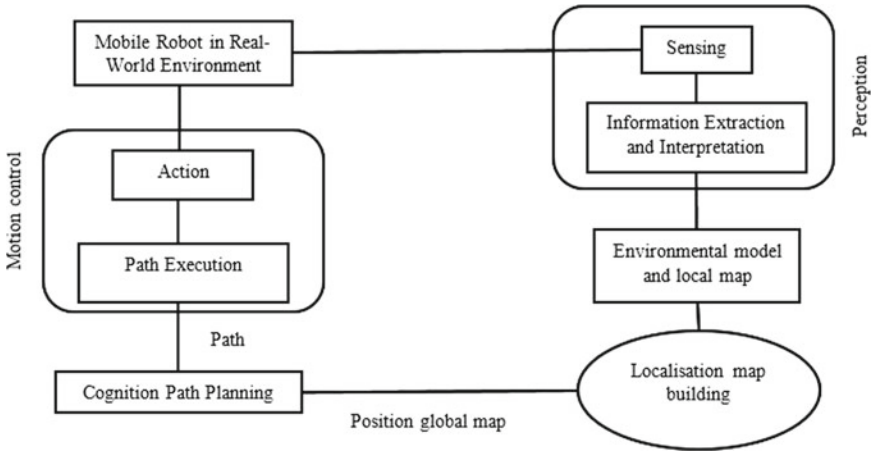


Fig. 1 Mobile robot navigation flow chart [46]

hybrid approaches. The deliberative approach for controlling the robot is computationally expensive as it involves the planning considering symbolic representations and world models. On the other hand, reactive control approach, as the name suggests, uses various sensors to perform actions and usually does not require intervening reasoning [20]. The mix of the above-mentioned approaches has also been developed previously where the control is both deliberative and reactive. The controller's part, which takes care of the deliberative approach, concerns with the high-level issues on a longer time scale, e.g. global path planning, while the part of the controller, which takes care of reactive approach, concerns with low-level control issues, e.g. local collision avoidance. These two parts of the controllers are often connected with the third component of the hybrid control system. Though the use of hybrid control in the mobile robot navigation system is advantageous, the implementation of this control makes the system quite complex and challenging. It is also worth noting that the reactive-based approaches are robust, hence, common, and have been used to improve the classical approaches [46].

Several approaches like potential field-based approach, probabilistic roadmap approach, grid-based approach, virtual impedance-based approach, divide-and-conquer-based approach, cell decomposition, etc. fall under the class of classic approaches. As per the work by Gemeinder and Gerke [21], it is learnt that, nowadays, several soft computing-based approaches from the domain of artificial intelligence have also been used to determine the collision-free trajectories in robot working environments. Example of few soft computing-based algorithms that have been applied in this field is neural network, particle swarm optimisation-based algorithms, firefly algorithm-based algorithms, genetic algorithm-based algorithms, cuckoo search based, artificial bee colony-based algorithms, etc.

1.1 Multi-robot Path Planning Approach

The case of multi-robot path planning becomes more interesting, where the robots move from their respective start points to their respective end points and while moving between these points, they have to avoid the obstacles as well as avoid colliding with each other. Hence, the coordination among the robots also becomes an important consideration, which should work robustly in real time. A simple idea to solve the multi-robot path planning problem is to embed every possibility of all the robots as a consolidated problem which can be later optimised using an evolutionary method. It is to be noted that those robots, which have their optimal robots away from the collision, do not affect each other [29]. Algorithms developed by Potter and de Jong [47, 48] based on coevolution, Sánchez-Ante and Latombe [57] and Kavraki et al. [32] are considered to be good choices for path planning of multi-robot. However, these algorithms were computationally rich which led to the development of other reactive-based approaches. Some of the work, in this field, include Khatib [34], Pradhan et al. [49], Baxter et al. [6], Selekwa et al. [62], Kala [29]. Various algorithms based on soft computing approaches have also been developed to solve this problem as discussed by Chakraborty et al. [11], Bhattacharjee et al. [8], Rakshit et al. [52], Kala [28, 29], Das et al. [15, 16], Abbas and Abdulsaheb [2], Faridi et al. [19], Thabit and Mohades [65] and Alotaibi and Al-Rawi [4].

1.2 Soft Computing-Based Approaches

As mentioned above, several researchers have applied various soft computing algorithms to not only improve the performance of the existing algorithms but also consider various cases in different environments such as aerial, underwater, land with various terrains and hazardous and industrial environments. The next sections give brief details of the algorithms used for robot path planning problem.

1.2.1 Genetic Algorithm

Genetic algorithm (GA) was proposed by Holland [26], which works on the biological evolution and genetics principle. Three main operators, used in GA, are the selection (to select the quality individuals out of the mating pool), the crossover (to mix the properties of parent individuals to generate offspring) and the mutation (to mutate the individuals so that they do not fall into local minima/maxima). Flow chart of the basic GA is given in Fig. 2. It has widely been used for both single and multiple robot path planning problems as found by Shing et al. [59], Xiao et al. [68], Kang et al. [30], Shi et al. [58], Kala [29], Liu et al. [39], Yang et al. [70], Hong et al. [25], Jianjum et al. [27], Patle et al. [45]. To tackle the problems of infeasible path generation, variable path length, etc., GA operators have been modified. Various

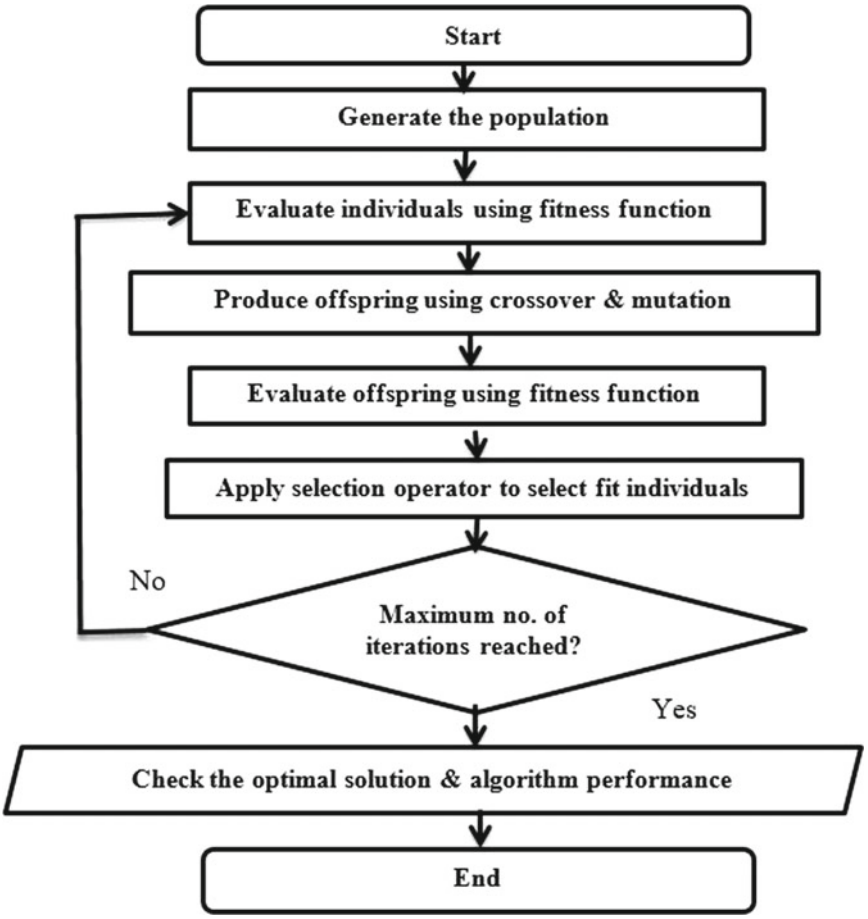


Fig. 2 Flow chart of the basic genetic algorithm

crossovers like partially mapped crossover (PMX), cycle crossover, order crossover, same point crossover (SP) and same adjacency crossover have been developed that contain different logics depending upon the specific problem.

1.2.2 Firefly Algorithm

Yang [71] had proposed the firefly algorithm (FA) which is motivated by flashing characteristics of fireflies. The flashes by the fireflies are used to appeal the coupling mate (used for the communication) and to draw the potential prey. It also serves as a mechanism for the protective warning for the fireflies. The pseudocode for the basic firefly algorithm is given in Fig. 3. Due to its success in various engineering fields, FA has also been used in mobile robot path planning. The application of FA was found in

```
Objective function f(i), i = (i1, ..., id)'
Initialise the population of N fireflies ij (j = 1, 2, ..., N)
Determine the Light intensity Ii at ii by f(ii).
Initialise the coefficient of light absorption, γ
while (T < MaxNumberOfGeneration)
for ii = 1: N
        for jj = 1: N
        if (Iii < Ijj)
                Move solution/firefly ii towards jj.
        end if
                Change the attractiveness with distance r
                Evaluate the new fireflies and replace the light intensity
        end for jj
end for ii
Sort the solutions/fireflies and determine the global best b*.
end while
Postprocess and visualise the results
```

Fig. 3 Pseudocode for the basic firefly algorithm [71]

the papers by Hidalgo-Paniagua et al. [24], Brand et al. [9], Sutantyon and Levi [61], Christensen et al. [12] and Wang et al. [67]. A system for mobile robot path planning was developed in the presence of stationary objects (obstacles) by Hidalgo-Paniagua et al. [24] considering the length of the path, smoothness and the safety criteria. A single mobile robot was considered in the work by Brand et al. [9] in a simulated environment. The FA has also been applied for robot planning in underwater by Sutantyon and Levi [60], Sutantyon and Levi [61] and Wang et al. [67] for aerial navigation. There have also been attempts to fuse FA with other approaches such as FA-ABC hybrid approach [1], FA-Q learning approach [55], and FA-vision-based approach [43].

1.2.3 Particle Swarm Optimisation

Particle swarm optimisation (PSO) had been innovated by Kennedy and Eberhart [33] which takes inspiration from the social behavioural patterns of organisms that live and intercommunicate within huge groups. The algorithm includes swarming behaviours observed in schools of fish, flocks of birds or swarms of bees and has also been observed in human social behaviour. The 'swarm intelligence' paradigm has developed by taking inspiration from these behaviours. The pseudocode for the basic PSO algorithm is given in Fig. 4. Few of the robot navigation planning works include Atyabi et al. [5], Tang and Eberhand [63], Xuan et al. [69], Couceiro et al.

Objective function $f(i)$, $i = (i_1, ..., i_d)'$
Initialize the locations i_j and the velocity v_j of N particles.
Determine the global best $b*$ from $\min\{f(i_1), ..., f(i_N)\}$ (when $T = 0$)
while ($T <$ MaxNumberOfGeneration)
    for loop over all d dimensions & all N particles
        Initialise new velocity v_j^{T+1}
        Compute new locations $i_j^{T+1} = i_j^{T} + v_j^{T+1}$
        Compute the objective functions at new locations i_j^{T+1}
        For each particle, determine the current best i_j*
    end for
    Find the current global best $b*$
    Increment the iteration counter T by one
end while
Postprocess and visualise the results

Fig. 4 Pseudocode for the basic PSO algorithm [71]

[14], Tang et al. [64] and Li et al. [37]. The navigational challenges embedded in an unknown environment are addressed in the work by Tang and Eberhand [63]. As mentioned before about the need of stable convergence for the soft computing-based algorithms, Xuan et al. [69] developed an algorithm by hybridizing the PSO with another algorithm called as mesh adaptive direct search. To address the cooperative motion path planning in complicated environments, Tang et al. [64] developed PSO with multi-body system dynamics that considered fault tolerance. For the problem of multi-robot path planning, Couceiro et al. [14] combined the PSO with DPSO to avoid the collision and issues related to mutual communication. Li et al. [37] used a self-adaptive PSO for robot navigation in challenging and complex environment while simultaneously considering several constraints.

1.2.4 Ant Colony Optimisation

The ant colony optimisation (ACO) algorithm had been presented by Dorigo [17] which follows the imitation of cooperation of ants when searching for food. While making their paths from nest to food source, a chemical substance which is known as 'pheromone' is deposited by these ants, which helps them communicate with other ants and make their path shortest over the time. The algorithm can easily be modelled on a graph, hence can be applied in various areas especially combinatorically explosive problems, and however, the generation of the graph is difficult, which further makes it computationally expensive. Some of the robot navigation planning work using ACO are: Liu et al. [41], Guan-Zheng et al. [22], Purian et al. [50], Castillo et al. [10], Liu et al. [40], Rajput and Kumari [54] and Kumar et al. [35]. In the work

by Liu et al. [39], a collision avoidance strategy was employed to solve the multi-robot navigation in the presence of stationary obstacles. Another interesting work was done by Guan-Zheng et al. [22] where the ACO algorithm was proposed to plan path in real time for the mobile robots. In this work, several advantages like convergence speed, computational time, etc., of the proposed ACO were found which were better than the GA-based approach. For unknown dynamic environments, Purian and Sadeghian [50] proposed the ACO that could select and optimise the fuzzy rules. Another hybridised version of ACO with fuzzy was demonstrated by Castillo et al. [10] for the path planning problem in the presence of stationary objects. Liu et al. [40] detailed the issues present in the ACO and proposed the improvement of the ACO by integrating the pheromone diffusion and local optimisation strategy. This work was further improved in terms of faster convergence by Kumar et al. [35] by further updating the strategies.

1.2.5 Cuckoo Search Algorithm

Cuckoo search (CS) algorithm had been proposed in the work by Yang and Deb [73] which imitates the egg-laying behaviour of cuckoos in the nests of other species. In the paper by Yang and Deb [73], few rules were laid down that mimic this phenomenon into the proposed algorithm. This algorithm has successfully been used in various engineering problems and has been found to give satisfactory results. The pseudocode for basic cuckoo search algorithm is shown in Fig. 5. List of the work where the problem of mobile robot path planning was

```
Objective function f(i), i = (i1, ..., id)'
Initialise the population of N host nests ij
while (T < MaxNumberOfGeneration) or (the stopping criteria)
    Select a cuckoo arbitrarily
    Generate a solution using Levy flights
    Compute its quality using objective function
    Select an individual nest among N (say, k) arbitrarily
        if (fj < fk),
            Store the new nest j in place of nest k
        end
    Abandon a fraction of worse nests
    Generate the new nests
    Retain the best nests
    Sort the population as per fitness value and determine the current best nest
    Increment the iteration counter T by one
end while
Postprocess and visualise the results
```

Fig. 5 Pseudocode for the basic cuckoo search algorithm [71]

solved using CS algorithms was: Wang et al. [66], Mohanty and Parhi [44]. In the work by Wang et al. [66], a hybrid approach by combining the differential evolution with cuckoo search was developed which worked for unknown environments and was found to be having better convergence speed in the aerial navigation system. While in the work by Mohanty and Parhi [44], it was found that the CS algorithm worked when combined with other navigation algorithms, especially in the case of complex and partially unknown environments. This was achieved by hybridizing the CS algorithm with adaptive neuro-fuzzy inference system (ANFIS).

1.2.6 Artificial Bee Colony Optimisation

Artificial bee colony algorithm (ABC) had been presented by Karaboga [31], where the bees present within a colony were categorised into the employed group, forager bees group, onlooker/observer bees and scouts group. For each food source, there is only one employed bee, and the number of food sources is assumed to be equal to the number of employed bees. The employed bee of a discarded food source is forced to become a scout bee who then starts to search for new food sources arbitrarily. The employed bees pass the information to the onlooker bees so that the onlooker bees can start searching for a new food source. As opposed to the honeybee algorithm, where there are two groups of bees (the observer bees and the forager bees), bees of ABC are more specialised. The intake efficiency, at a particular food source, can be calculated by the following equation

$$\text{Intake efficiency} = N/T,$$

where

N the amount of nectar present and
T the amount of time spent at food site.

At a given number of explorations, if a food source is found without improvement, then the food source is vacated, and the bee starts moving randomly to further explore new locations, from the present location. ABC algorithm has been used in various areas such as combinatorial optimisation, job scheduling and engineering design optimisation in the previous few years. Some of the robot path planning work where the ABC algorithm has been applied are: Saffari and Mahjoob [56], Ma et al. [42], Bhattacharjee et al. [8], Contreras-Cruz [13], Liang et al. [38] and Bhagade and Puranik [7].

In the work by Saffari and Mahjoob [56], ABC algorithm was developed when the obstacles were present in static condition. Ma et al. [42] demonstrated a hybridised version of ABC algorithm which could successfully plan the robot path in the real time for the dynamic environments. In order to tackle the problem of multiple mobile robot navigation, Bhattacharjee et al. [8] developed an ABC algorithm which could successfully work in stationary environments. The work demonstrated by Contreras-Cruz [13] showed the working of an ABC algorithm for the stationary environment

where the local search in addition to the evolutionary algorithm was used to get the optimal path. The ABC algorithm has also been applied in underwater navigation, aerial navigation and also in problems related to vehicle routine.

1.3 Challenges in the Application of Artificial Intelligent Approaches

The general problem in robot navigation is searching for the best path out of several possible paths so that a single or multiple robot(s) moves efficiently and safely in the given environment with the least time or energy incurred. However, the problem gets further challenging as the search space becomes bigger, resulting in a huge number of feasible paths. This problem is also present in case of soft computing based on artificial intelligent approaches. But due to their population-based search strategy, they are less affected and can yield the expected result in polynomial time [74]. In addition to general challenges like convergence speed, run time and parameter tuning of the algorithm, another challenge with soft computing-based algorithms is to generate a methodology to cope up with different numbers of intermediate points in a path through which the robot has to pass. A fixed-length solution in path planning does not work for a complex environment and can increase the time required to evolve a solution because of its inefficiency [53]. Researchers have innovated different modelling scheme to cope up with this issue, such as in the work by Elshamli et al. [18] discussed a methodology to represent the variable path length. Qiongbing and Lixin [51] studied the variable-length chromosome of their GA used by previous researchers and then devised a new crossover named as 'Same Adjacency' to take care of variable-length issue. In the work by Lamini et al. [36], they used the concept of variable-length chromosome, and accordingly, a novel crossover operator has also been proposed. While developing a soft computing-based algorithm, to solve the problem of mobile robot navigation, researchers have to consider the model of the robot's working environment, especially complex shaped obstacles (i.e. irregular shaped objects), consideration of the physical size of the robot (which, mostly, is assumed to be a point/negligible volume when planning), no repeated points/coordinates in the generated optimal path, generation of infeasible path during the working of the algorithm, etc. The last two challenges mentioned can be overcome by applying the sound repair strategies for the generated solutions.

In this work, a global path planning approach for the mobile robots have been developed where information about the robot's surrounding is known beforehand and the path is planned before the robot starts the navigation. An FPA-based approach has been proposed here to generate the feasible as well as optimal path for the mobile robot. The next section describes the basic principle of the FPA, followed by the proposed approach for path planning, results and discussions and finally conclusions.

2 Flower Pollination Algorithm

2.1 Basic Principle

This section briefs the working of flower pollination algorithm (FPA) or flower algorithm, developed by Yang [72]. It is known that the flowering plants have been evolving through the process of evolution for over 125 million years. A flower performs the reproduction via pollination. The transfer of pollens causes this pollination process which is transferred by the pollinators such as birds, insects and bat. There could be two types of pollination phenomenon in nature, namely abiotic and biotic. Almost 90% of the pollination happens in the form of biotic pollination in the flowering plants, where pollens are transferred mainly by animals and insects. On the other hand, almost 10% of the pollination is performed by the abiotic pollination which requires no pollinators. The abiotic pollination is performed by wind and diffusion in water, and a good example of it is grass. Another good pollinator is honeybees. A mechanism known as flower constancy provides evolutionary advantages as it can increase the transfer of flower pollen to the same plants and thus maximising the reproduction and also the survival of the same flower species.

In addition to the above, pollination has also been categorised into the self-pollination and the cross-pollination. The cross-pollination, also known as allogamy, is a form of pollination that occurs from the pollen of a flower of a different plant, and on the other hand, self-pollination is the pollination of one flower, from the pollen of the same flower or different flowers of the same plant. It is known that the cross and biotic pollination usually happens at long distances due to pollinators, namely bats, birds and bees, which fly for a long distance considering it as the global pollination. The birds and bees behave like the Levy flight behaviour, where their jump or fly distance steps follow the Levy distribution. Yang [72] has laid the following rules to model the flower pollination algorithm:

1. Biotic-cross-pollination is the global pollination where the pollinators carry the pollens using Levy flights.
2. Local pollination is considered to be a form of abiotic-self-pollination.
3. Flower constancy is assumed to be the reproduction probability.
4. To control the amount of pollination between local and global, a switch probability is proposed.

The global pollination can be represented mathematically as Eq. 1.

$$x_i^{t+1} = x_i^t + \gamma\, L\left(g^* - x_i^t\right) \tag{1}$$

where

x_i^t pollen i or individual vector x_i at iteration t,
g^* the current best individual present in the population at the iteration t,
γ a scaling factor,

L a step-size parameter based on Levy flights that measures the pollination strength.

This L is calculated using the Levy distribution function as shown in Eq. 2.

$$L = \frac{\lambda \Gamma(\lambda) \sin(\pi \lambda / 2)}{\pi} * \frac{1}{s^{1+\lambda}} \tag{2}$$

The gamma function Γ, given in the above equation, depends on λ. For $\lambda = 1.5$, $\Gamma = 0.88$ [3].

The local pollination is mathematically written as Eq. (3).

$$x_i^{t+1} = x_i^t + \in \left(x_j^t - x_k^t\right) \tag{3}$$

where

x_i^t pollen i or individual vector x_i at iteration t,
$\in$ a random number/vector from uniform distribution within 0–1,
x_i^t pollen j or individual vector x_j at iteration t,
x_i^t pollen k or individual vector x_k at iteration t.

The pseudocode for the basic FPA is given in Fig. 6.

```
Objective function f(i), i = (i1, ..., id)'
Generate N random flowers (solutions) as a population
Select the current best flower b* in initial population
Set the switch probability sp, ∈ [0, 1]
while (T < MaxNumberOfGeneration)
      for ii=1:N (all N solutions)
      if random_number < sp
          Generate a vector that follows Levy distribution
          Apply the Global pollination operator
      else
          Generate ∈ from a uniform distribution in the range of [0,1]
          Arbitrarily select j and k among all flowers
          Apply the Local pollination operator
       end if
       Evaluate new flowers using fitness function
       If new flowers are better, store them in the population and discard poorer ones
       end for
Find the current best solution b*
Increment the iteration counter T by one
end while
Postprocess and visualise the results
```

Fig. 6 Pseudocode of the basic flower pollination algorithm [72]

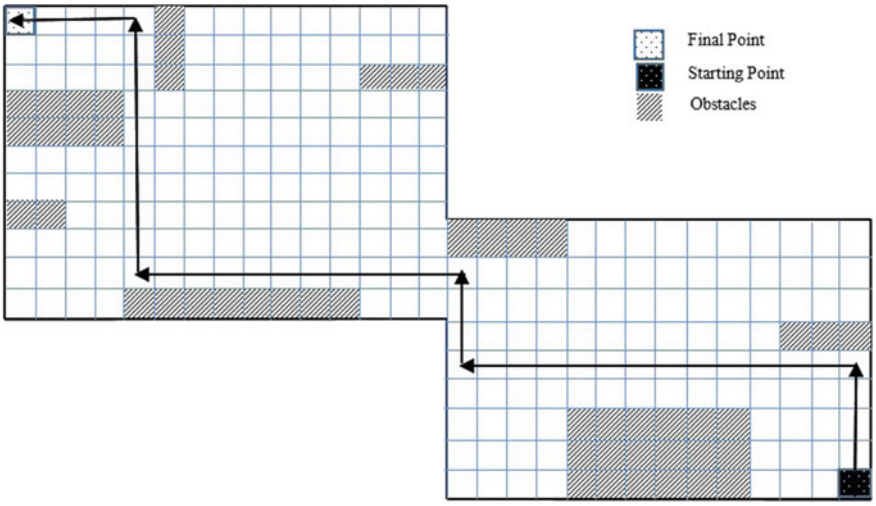


Fig. 7 Environment showing a feasible path for robot navigation

2.2 Proposed Approach for Robot Path Planning

The following sections detail the representation scheme, initial population generation, global pollination rule, local pollination rule and the fitness function used in the flower pollination algorithm.

2.2.1 Representation Scheme

For an algorithm to successfully work and yield the optimal results, it is very important to model the problem's data in which the information related to the problem is represented. It also affects the efficiency of the algorithm. In the case of path planning of mobile robots, previous researchers have used different modelling schemes to represent the information. In the present work, the working environment where the robot path has to be determined is divided into the small-sized square cell. A sample of this is shown in Fig. 7. For the recognition of each cell and location of obstacles and the robots, each cell has been recognised using an index number attached to it as shown in Fig. 8 (with one-level padding) and one of the possible paths.

2.2.2 Population Generation

Most of the soft computing algorithm work on the principle of the population-based approach, where several population candidates are generated followed by a few operators that are used intelligently to yield better population individuals. Several

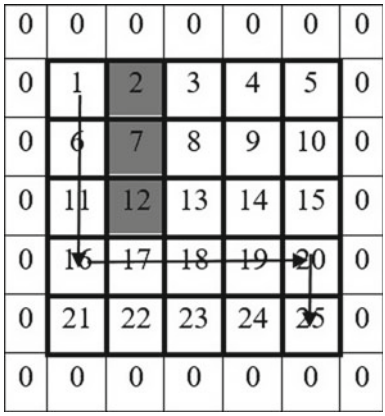


Fig. 8 Map showing the start point (point number 1) and end point (25), the static obstacles (shown in grey colour) with one of the possible paths

researchers have used classic algorithms to generate initial paths. In this work, a novel strategy has been used to generate the initial population. This has been shown below using an example. First, the user inputs the size of the cell where the robot has to move on the path as well as the size of the cell and population size and start and end points. The algorithm starts from the start point then selects four points surrounding the start point (i.e. left, right, up, down). Then, a random number is generated which is then compared with an optimal path probability (OPP) which is set by the user. If the random number is greater than this OPP value, the next point is selected out of four points which have the least distance from the end point. However, if the generated random number is found to be lower than the set OPP, then a random number is selected out of the surrounding numbers. This strategy gives the advantage of generating various paths between the start point and the end point. However, it is to be worth noting that this can make the cell visited by the robot more than once, which usually does not happen in case of *A** algorithm. The operators of the FPA are designed to take care of this so that none of the cells is repeated more than once. In order to avoid the problem of boundary elements, the padding of zeros has also been added to one level and two levels.

2.2.3 Global Pollination Rule

Global pollination rule as mentioned in the previous section is used to move the population individual to the current best individual. Unlike assembly sequence planning, the length of the population individuals may vary in robot path planning problem, and hence, the pollination operators have to be modified to take into account the length of the individual. Following Eq. 1, the length of the flowers (i.e. flower in consideration and the best flower found so far) is determined, and the flower with smaller length is appended with the zeros. This process makes the two flowers equal in length. Then, the elementwise difference between the two flowers is calculated, and if elements

g*	1	6	11	16	21	22	23	24	25	0	0
x_i^t	1	2	7	6	11	16	21	22	23	24	25
$(g^*-x_i^t)$	0	4	4	10	10	6	2	2	2	0	0
L	1	4	1	5	1	1	1	1	2	1	1
$L^* (g^*-x_i^t)$	0	16	4	50	10	6	2	2	4	0	0
$x_i^{t+1} = L^* (g^*-x_i^t) + x_i^t$	1	18	9	56	21	22	23	24	27	24	25
$\sim x_i^{t+1} = L^* (g^*-x_i^t) + x_i^t$	1	2	7	12	17	22	23	24	25		

Fig. 9 Example of global pollination rule for the proposed FPA-based approach

are found to be less than zero, the absolute value of the element is taken into consideration. Thereafter, a Levy vector (a random vector following the Levy distribution) equal to the length of the flower in consideration is generated. This Levy vector is multiplied elementwise by the difference of the flower in consideration, and the best flower found so far. This may cause some of the elements' value more than the value of a maximum number of cells. In order to repair this, these elements are replaced with nearest neighbouring elements. Finally, all the zeros present in the flower are removed, if any. An example of a global pollination rule is shown in Fig. 9.

2.2.4 Local Pollination Rule

Local pollination rule as mentioned in the previous section is used to improve the population individual locally. In this pollination rule, the flower in consideration and two random flowers taken from the same population could be of different length. At first, two random numbers are generated using uniform distribution and corresponding to the random numbers, and two individuals are selected from the population. Like global pollination rule, zeros are added at the end of the flowers to make them uniform sized. Then, the elementwise subtraction between the two flowers is performed. Following which, all the elements, which are less than zero, are made zero. Thereafter, a random vector equal to the length of the difference is generated, where values of the elements are generated following the uniform distribution. This random vector is then multiplied elementwise by the difference of the flower obtained earlier. Then, this multiplication is added elementwise into the flower in consideration. This addition might cause few elements to have the value more than the number of cells, and in order to avoid these cells, all these values are replaced by the neighbouring elements starting from the first element, which is made as the starting point of the robot to make the path feasible. Finally, all the zeros present in the flower are removed and to repair the last element of the path is used as the starting point and the same process as discussed in the population generation is used to reach the end point. An example of local pollination rule is shown in Fig. 10.

x_i^t	1	2	3	4	9	14	19	24	25		
xj^t	1	2	3	4	9	14	19	23	24	25	0
x_k^t	1	2	3	4	9	14	19	18	23	24	25
$(xj^t - xk^t)$	0	0	0	0	0	0	0	5	1	1	-25
$\in$	0	0	0	1	0	0	0	1	1	0	
$\in * (xj^t - xk^t)$	0	0	0	0	0	0	0	5	1	0	
$x_i^{t+1} = \in * (xj^t - xk^t) + x_i^t$	1	2	3	4	9	14	19	29	26	25	
$\sim x_i^{t+1} = \in * (xj^t - xk^t) + x_i^t$	1	2	3	4	9	14	19	18	23	24	25

Fig. 10 Example of local pollination rule for the proposed FPA-based approach

2.2.5 Fitness Function

To compare the different population individuals, a fitness function has been used, which shows the quality of that individual. In case of mobile robot path planning problem, several criteria like safety, smoothness, covered distance, etc. can be considered. In the development of a robust soft computing algorithm, the selection of a fitness function is very important, as the algorithm would apply the information generated by this function. In the present work, four different criteria have been taken into account which was recently used in the paper by Lamini et al. [36]. The details of the fitness function can be found in that paper. These criteria used are energy, covered path length, safety-first level (SFL) and safety-second level (SSL). Equation 4 shows the fitness function.

$$FF = \frac{1}{w_1 * l(p) + w_{s1} * s_1(P) + w_{s2} * s_2(P)} - E \tag{4}$$

$l_i(p)$ represents the path length.

$$A = \sum_{i=1}^{N-1} \begin{cases} 1, & \text{If the next element is the neighbouring element} \\ \text{euclidean distance}, & \text{otherwise} \end{cases}$$

$s_1\,(p)$ represents the safety-first level (SFL). If an obstacle is found in the first security level of the present position $p\,(x_i, y_i)$ of the robot, the s_i gets a fine of $+1$. The value can be calculated using

$$s_1(p) = \sum_{i=1}^{N-1} s_i$$

$s_2\,(p)$ represents safety-second level (SSL). If an obstacle is found in the second security level of the present position $p\,(x_i, y_i)$ of the robot, the s_i gets a fine of $+1$. The value can be calculated using:

$$s_2(p) = \sum_{i=1}^{N-1} s_i$$

E is the energy term which is a penalty for energy consumption, if the robot makes a turn, E gets a fine of $+1$, and gets 0 if it continues moving straight. w_l, ws_1 and ws_2 are the weight coefficients for the covered path length, safety-first level and safety-second level properties for a given path p.

3 Results and Discussions

This work describes the different approaches used for mobile robot path planning problems. Approaches that fall under soft computing are also detailed and their working, and optimality criteria with their applications have also been provided. Thereafter, a flower pollination-based algorithm has been attempted to model the problem for the path planning problem, and also the results of it has also been shown. The proposed approach has been tested on a sample cell, which has a grid size of (5 * 5) and shown in Fig. 8, to verify the results. Then, another different map, which is larger than the previous one and more complex, has been taken from the work by Lamini et al. [36] where the grid sizes were (29 * 30). This data set is given in Table 1.

When the proposed algorithm is run with the optimal set of parameters with 40 independent runs, in case of environment a, the average number of turns for the optimal path yielded by the proposed FPA is found to be 2.4. Figure 11 shows the optimal path found by the proposed FPA approach for the environment 'a'. In the figure, point number 1 is the start point for the robot (in blue colour), point number 25 is the end point (in yellow colour) and static obstacles are shown in grey colour.

When the proposed algorithm is run with the optimal set of parameters with 40 independent runs, in case of environment 'b', the average number of turns for the optimal path yielded by the proposed FPA is found to be 8.6. As mentioned earlier that the optimal path should be safe and should take the least energy to travel through, one of the important criteria was the number of turns which require additional energy by the robotic system. These frequent turns make the path zigzag which also reduces the smoothness of the movement of the mobile robot.

Table 2 displays the number of turns for each environment case found in the optimal path yielded by the proposed FPA. Figure 12 shows the optimal path found by the proposed FPA approach.

Table 1 Robot navigation environments

Environments	Rows	Columns
Environment 'a'	5	5
Environment 'b'	29	30

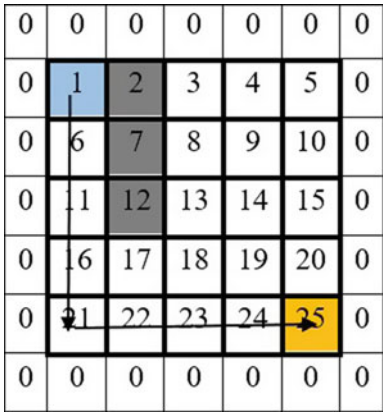


Fig. 11 Optimal path yielded by the proposed approach for the environment '*a*'

Table 2 Number of turns for each environment case found in the optimal path yielded by the proposed FPA

Environments	Number of turns in the optimal path
Environment '*a*'	1
Environment '*b*'	7

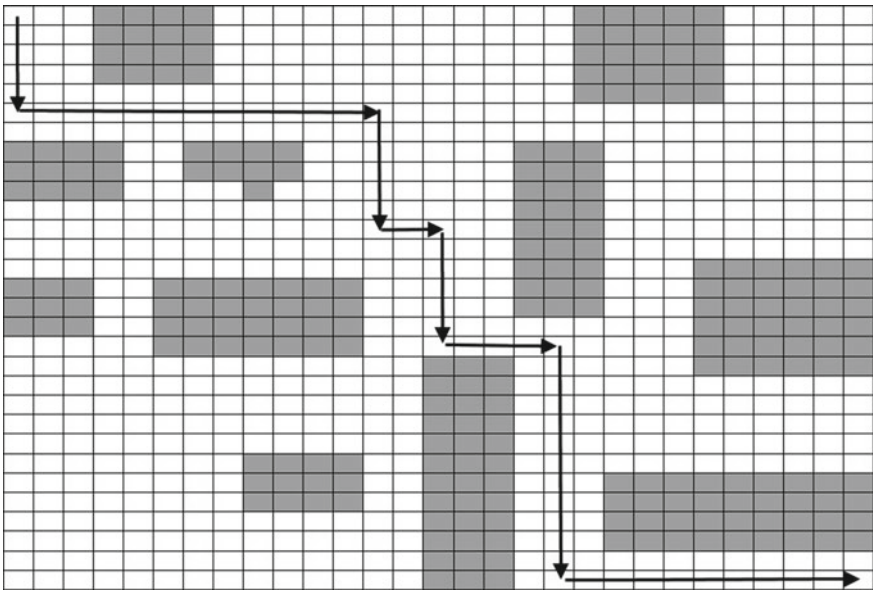


Fig. 12 Optimal path yielded by the proposed approach for environment '*b*'

4 Conclusions for the Book Chapter

Mobile robot path planning has been a hot area of research for a long time. To solve this problem, first classical algorithms were developed which used to be quite computationally expensive then with the advent of artificially intelligent algorithms, and soft computing-based algorithms also have been applied in this domain. Since the power of the computers nowadays has gone manifold, these algorithms and their hybrid versions have been very popular. These soft computing planners for mobile robot navigation problem is not only limited to the path planning on land but also can be used to plan in different environments like the aerial, underwater and different terrains. In this chapter, basic concepts related to mobile robot navigation have been discussed followed by the brief details of different soft computing-based algorithms that have been used for robot navigation problem. A section has been devoted to discuss the challenges in the application of the soft computing-based algorithm in this field. Then, an existing algorithm, namely flower pollination algorithm, has been developed for the mobile robot path planning in a static environment. The working operators, namely global pollination and local pollination of the FPA-based approach, are also elaborated and how the challenges of application of FPA are overcome. Then, finally, the results of the proposed approach have been shown using a few examples, followed by the discussions.

References

1. Abbas NH, Saleh BJ (2016) Design of a kinematic neural enhanced hybrid firefly for mobile robots based on enhanced hybrid firefly-artificial bee colony algorithm. Al-Khwarizmi Eng J 12(1):45–60
2. Abbas NH, Abdulsaheb JA (2017) An adaptive multi-objective artificial bee colony algorithm for multi-robot path planning. Assoc Arab Univ J Eng Sci 24(3):168–189
3. Abramowitz M, Stegun IA (1964) Handbook of mathematical functions with formulas, graphs, and mathematical tables. National Bureau of Standards Applied Mathematics Series, USA
4. Alotaibi ETS, Al-Rawi H (2018) A complete multi-robot path-planning algorithm. Auton Agent Multi-Agent Syst 32(5):693–740
5. Atyabi A, Phon-Amnuaisuk S, Ho CK (2010) Applying area extension PSO in robotic swarm. J Intell Robot Syst 58:253–285
6. Baxter JL, Burke EK, Garibald JM, Norman M (2009) Shared potential fields and their place in a multi-robot co-ordination taxonomy. Robot Auton Syst 57(10):1048–1055
7. Bhagade AS, Puranik PV (2012) Artificial bee colony (ABC) algorithm for vehicle routing optimisation problem. Int J Soft Comput Eng 2(2):329–333
8. Bhattacharjee P, Rakshit P, Goswami I, Konar A, Nagar AK (2011) Multi-robot path-planning using artificial bee colony optimisation algorithm. In: 2011 Third world congress on nature and biologically inspired computing. IEEE, pp 219–224
9. Brand M, Yu X-H (2013) Autonomous robot path optimisation using firefly algorithm. In: International conference on machine learning and cybernetics, vol 3. Tianjin, pp 14–17
10. Castillo O, Neyoy H, Soria J, Melin P, Valdez F (2015) A new approach for dynamic fuzzy logic parameter tuning in ant colony optimisation and its application in fuzzy control of a mobile robot. Appl Soft Comput 28:150–159

11. Chakraborty J, Konar A, Jain LC, Chakraborty UK (2009) Cooperative multi-robot path planning using differential evolution. J Intell Fuzzy Syst 20(1, 2):13–27
12. Christensen AL, Rehan OG, Dorigo M (2008) Synchronization and fault detection in autonomous robots. In: IEEE/RSJ intelligent conference on robots and systems, pp 4139–4140
13. Contreras-Cruz MA, Ayala-Ramirez V, Hernandez-Belmonte UH (2015) Mobile robot path planning using artificial bee colony and evolutionary programming. Appl Soft Comput 30:319–328
14. Couceiro MS, Rocha RP, Nuno F (2013) A PSO multi-robot exploration approach over unreliable MANETs. Adv Robot 27(16):1221–1234
15. Das PK, Sahoo BM, Behera HS, Vashisht S (2016a) An improved particle swarm optimisation for multi-robot path planning. In: 2016 international conference on innovation and challenges in cyber security (ICICCS-INBUSH). IEEE, pp 97–106
16. Das PK, Behera HS, Panigrahi BK (2016) A hybridization of an improved particle swarm optimisation and gravitational search algorithm for multi-robot path planning. Swarm Evol Comput 28:14–28
17. Dorigo M (1992) Optimisation, learning and natural algorithms. Thesis, Department of Electronics, Politecnico di Milano, Italy
18. Elshamli A, Abdullah HA, Areibi S (2004) Genetic algorithm for dynamic path planning. In: Canadian conference on electrical and computer engineering 2004 (IEEE Cat. No. 04CH37513), vol 2, pp 677–680
19. Faridi AQ, Sharma S, Shukla A, Tiwari R, Dhar J (2018) Multi-robot multi-target dynamic path planning using artificial bee colony and evolutionary programming in unknown environment. Intel Serv Robot 11(2):171–186
20. Ge SS (2006) Autonomous mobile robots: sensing, control, decision making and applications. CRC press
21. Gemeinder M, Gerke M (2003) GA-based path planning for mobile robot systems employing an active search algorithm. Appl Soft Comput 3:149–158
22. Guan-Zheng T, Huan HE, Aaron S (2007) Ant colony system algorithm for real time globally optimal path planning of mobile robots. Acta Autom Sin 33(3):279–285
23. Hachour O (2008) Path planning of autonomous mobile robot. Int J Syst Appl Eng Dev 2(4):178–190
24. Hidalgo-Paniagua A, Vegae Rodriguez MA, Ferruz J, Pavon N (2015) Solving the multi-objective path planning problem in mobile robotics with a firefly-based approach. Soft Comput. 1–16
25. Hong Q, Ke X, Alexander T (2013) An improved genetic algorithm with co-evolutionary strategy for global path planning of multiple mobile robots. Neurocomputing 120:509–517
26. Holland J (1975) Adaptation in natural and artificial systems. MIT Press, Cambridge, MA, USA
27. Jianjun N, Wang K, Huang H, Wu L, Luo C (2016) Robot path planning based on an improved genetic algorithm with variable length chromosome. In: 12th international conference on natural computation, fuzzy systems and knowledge discovery (ICNC-FSKD). https://doi.org/10.1109/FSKD.2016.7603165
28. Kala R (2012) Multi-robot path planning using co-evolutionary genetic programming. Expert Syst Appl 39(3):3817–3831
29. Kala R (2014) Coordination in navigation of multiple mobile robots. Cybern Syst 45(1):1–24
30. Kang X, Yue Y, Li D, Maple C (2011) Genetic algorithm based solution to dead problems in robot navigation. Int J Comput Appl Technol 41(3–4):177–184
31. Karaboga D (2005) An idea based on honey bee swarm for numerical optimisation. Erciyes University, Engineering Faculty, Computer Engineering Department. Technical report-tr06
32. Kavraki LE, Kolountzakis MN, Latombe JC (1998) Analysis of probabilistic roadmaps for path planning. IEEE Trans Robot Autom 14(1):166–171
33. Kennedy J, Eberhart R (1995) Particle swarm optimisation. In: Proceedings of ICNN'95—international conference on neural networks, Perth, WA, Australia, vol 4, pp 1942–1948. https://doi.org/10.1109/icnn.1995.488968

34. Khatib O (1985) Real-time obstacle avoidance for manipulators and mobile robots. In: Paper presented at the 1985 IEEE international conference on robotics and automation, St. Louis, MO
35. Kumar PB, Sahu C, Parhi DR (2018) A hybridised regression-adaptive ant colony optimisation approach for navigation of humanoids in a cluttered environment. Appl Soft Comput 68:565–585
36. Lamini C, Benhlima S, Elbekri A (2018) Genetic algorithm based approach for autonomous mobile robot path planning. Procedia Comput Sci 127:180–189. https://doi.org/10.1016/j.procs.2018.01.113
37. Li G, Chou W (2018) Path planning for mobile robot using self-adaptive learning particle swarm optimisation. Sci China Inf Sci 61(5):052204
38. Liang JH, Lee CH (2015) Efficient collision-free path-planning of multiple mobile robots system using efficient artificial bee colony algorithm. Adv Eng Softw 79:47–56
39. Liu F, Liang S, Xian X (2014) Optimal robot path planning for multiple goals visiting based on tailored genetic algorithm. Int J Comput Intell Syst 7(6):1109–1122
40. Liu J, Yang J, Liu H, Tian X, Gao M (2017) An improved ant colony algorithm for robot path planning. Soft Comput 21(19):5829–5839
41. Liu S, Mao L, Yu J (2006) Path planning based on ant colony algorithm and distributed local navigation for multi-robot systems. In: Proceedings of the 2006 IEEE international conference of mechatronics and automation, 1733–1738
42. Ma Q, Lei X (2010) Dynamic path planning of mobile robots based on ABC algorithm. In: International conference on artificial intelligence and computational intelligence. Springer, Berlin, , pp 267–274
43. Mitic M, Miljkovic Z (2015) Bio-inspired approach to learning robot motion trajectories and visual control commands. Expert Syst Appl 42:2624–2637. https://doi.org/10.1016/j.eswa.2014.10.053
44. Mohanty PK, Parhi DR (2015) A new hybrid optimisation algorithm for multiple mobile robots navigation based on the CS-ANFIS approach. Memetic Comput 7(4):255–273
45. Patle BK, Parhi DRK, Jagadeesh A, Kashyap SK (2018) Matrix-Binary Codes based Genetic Algorithm for path planning of mobile robot. Comput Electr Eng 67:708–728
46. Patle BK, Ganesh LB, Pandey A, Parhi DRK, Jagadeesh A (2019) A review: on path planning strategies for navigation of mobile robot. Defence Technol 15(4):582–606. https://doi.org/10.1016/j.dt.2019.04.011
47. Potter MA, Jong KAD (1994) A cooperative co-evolutionary approach to function optimisation. In: Davidor Y, Schwefel HP, Männer R (eds) Proceedings of the third conference on parallel problem solving from nature. Springer, Berlin, pp 249–257
48. Potter MA, Jong KAD (2000) Cooperative coevolution: an architecture for evolving co-adapted subcomponents. Evol Comput 8(1):1–29
49. Pradhan SK, Parhi DR, Panda AK, Behera RK (2006) Potential field method to navigate several mobile robots. Appl Intell 25(3):321–333
50. Purian FK, Sadeghian E (2013) Mobile robots path planning using ant colony optimisation and fuzzy logic algorithm in unknown dynamic environment. In: International conference on control, automation, robotics and embedded systems (CARE), pp 1–6
51. Qiongbing Z, Lixin D (2016) A new crossover mechanism for genetic algorithms with variable-length chromosomes for path optimisation problems. Expert Syst Appl 60:183–189
52. Rakshit P, Konar A, Bhowmik P, Goswami I, Das S, Jain LC, Nagar AK (2013) Realization of an adaptive memetic algorithm using differential evolution and q-learning: a case study in multi-robot path planning. IEEE Trans Syst Man Cybern Syst 43(4):814–831
53. Raja P, Pugazhenthi S (2009) Path planning for mobile robots in dynamic environments using particle swarm optimisation. In 2009 IEEE international conference on advances in recent technologies in communication and computing, pp 401–405
54. Rajput U, Kumari M (2017) Mobile robot path planning with modified ant colony optimisation. Int J Bio-Inspired Comput 9(2):106–113

55. Sadhu AK, Konar A, Bhattacharjee T, Das S (2018) Synergism of firefly algorithm and Q-learning for robot arm path planning. Swarm Evol Comput. https://doi.org/10.1016/j.swevo.2018.03.014
56. Saffari MH, Mahjoob MJ (2009) Bee colony algorithm for real-time optimal path planning of mobile robots. In: Soft computing, computing with words and perceptions in system analysis. Decision and control, pp 1–4
57. Sánchez-Ante G, Latombe JC (2002) Using a PRM planner to compare centralised and decoupled planning for multi-robot systems. In: Proceedings of IEEE international conference on robotics and automation, Washington, DC
58. Shi P, Cui Y (2010) Dynamic path planning for mobile robot based on genetic algorithm in unknown environment. In: Proceedings of the Chinese control and decision conference, Xuzhou, China, pp 4325–4329
59. Shing MT, Parkar GB (1993) Genetic algorithm for the development of real-time multi-heuristic search strategies. In: Proceedings 5th conference on genetic algorithm. Los Aitos, California: Morgan Kaumann Publication, pp 565–570
60. Sutantyo D, Levi P, Moslinger C, Read M (2013) Collective-adaptive levy flight for underwater multi-robot exploration. In: International conference on mechatronics and automation, pp 456–462
61. Sutantyo D, Levi P (2015) Decentralised underwater multi robot communication using bio-inspired approaches. Artif Life Robot 20:152–158
62. Selekwa MF, Dunlap DD, Shi D, Collins EGJ (2008) Robot navigation in very cluttered environments by preference-based fuzzy behaviours. Robot Auton Syst 56(3):231–246
63. Tang Q, Eberhard P (2011) Cooperative motion of swarm mobile robots based on particle swarm optimisation and multibody system dynamics. Mech Base Des Struct Mach 39(2):179–193
64. Tang X, Li L, Jiang B (2014) Mobile robot SLAM method based on multi-agent particle swarm optimised particle filter. J China Univ Posts Telecommun 21(6):78–86
65. Thabit S, Mohades A (2018) Multi-robot path planning based on multi-objective particle swarm optimisation. IEEE Access 7:2138–2147
66. Wang G, Guo L, Duan H, Wang H, Liu L, Shao MA (2012a) Hybrid metaheuristic DE/CS algorithm for UCAV three-dimension path planning. Sci World J 583973. https://doi.org/10.1100/2012/583973
67. Wang G, Guo L, Hong D, Duan H, Liu L, Wang HA (2012b) Modified firefly algorithm for UCAV path planning. Int J Hosp Inf Technol 5(3):123e44
68. Xiao J, Michalewicz Z, Zhang L, Trojanowski K (1997) Adaptive evolutionary planner/navigator for mobile robot. IEEE Trans Evol Comput 1(1)
69. Xuan VH, Cheolkeun H, Jewon L (2013) Novel hybrid optimisation algorithm using PSO and MADS for the trajectory estimation of a four track wheel skid-steered mobile robot. Adv Robot 27(18):1421–1437
70. Yang SX, Hu Y, Meng MQ (2006) A knowledge based GA path planning of multiple mobile robots in dynamic environments. In: 2006 IEEE conference on robotics: automation and Mechatronics, Bangkok, Thailand, pp 1–6
71. Yang XS (2008) Nature-inspired metaheuristic algorithm. Luniver press
72. Yang XS (2012) Flower pollination algorithms. In: Yang XS (ed) Nature-inspired optimisation algorithms. Elsevier, London, pp 155–173
73. Yang XS, Deb S (2009) Cuckoo search via Levy flights. In: Nature and biologically inspired computing, NaBIC 2009, IEEE World Congress, pp 210–214
74. Yang XS, Karamanoglu M (2013) Swarm intelligence and bio-inspired computation: an overview. In: Swarm intelligence and bio-inspired computation. Elsevier, pp. 3–23

Chapter 7
Smartphone Indoor Localization Using Bio-inspired Modeling

Rafael Alexandrou, Harris Papadopoulos and Andreas Konstantinidis

1 Introduction

The fact that people spend the vast majority of their time inside buildings including their home, work place, libraries, airports, etc., in combination with the ubiquitous availability of the smartphone devices has boosted the interest of Indoor localization services (ILS) for applications such as wayfinding, labor optimization, asset tracking, and more.

The inability, however, of the Global Navigation Satellite Systems (GNSS), such as GPS, to accurately localize a device in an indoor environment due to the satellite signal attenuation while passing through solid objects (such as ceiling, or concrete walls), has directed the indoor localization community to research alternative techniques. For example, a variety of indoor localization techniques utilize technologies such as Bluetooth Beacons (BLE), infrared, audio and visual analytics, Li-Fi technologies, RFID, sensor networks and their combinations for localizing a device in an indoor environment with a fine-grain accuracy [1]. These techniques, however, require the deployment of specialized equipment such us antennas, beacons, and custom transmitters a priory [2], which is time consuming and costly.

Numerous ILS that rely on geolocation databases containing data retrieved from the existing infrastructure of a building or the environment, such as wireless, magnetic and light signals have been, therefore, implemented to alleviate the aforementioned challenges. These ILS, such as *Google, Indoo.rs, Navizon, IndoorAtlas, ByteLight* and *Anyplace*[1] [3] provide the accurate location (position) of a user upon request without the need of expensive additional hardware installation.

[1] Available at: http://anyplace.cs.ucy.ac.cy/.

R. Alexandrou · H. Papadopoulos · A. Konstantinidis (✉)
Frederick University, 7 Y. Frederickou Str. Pallouriotisa, 1036 Nicosia, Cyprus
e-mail: com.ca@frederick.ac.cy

R. Alexandrou
e-mail: st008390@stud.fit.ac.cy

H. Papadopoulos
e-mail: com.ph@frederick.ac.cy

X.-S. Yang and Y.-X. Zhao (eds.), *Nature-Inspired Computation in Navigation and Routing Problems*, Springer Tracts in Nature-Inspired Computing,
https://doi.org/10.1007/978-981-15-1842-3_7

In particular, ILS geolocation DB entries act as reference points for the requested localization tasks. A smartphone can, therefore, determine its location at a coarse granularity (i.e., km or hundreds of meters) up to a fine granularity (i.e., 1–2 m), by comparing against the reference points, either on the service (server-side) or on the smartphone itself (client-side). As explained in [4], one fundamental drawback of server-side ILS is that these receive information about the location of a user while servicing them, generating a variety of location privacy concerns (e.g., surveillance or data for unsolicited advertising). These concerns do not exist with the client-side ILS, as the necessary data are downloaded and the localization is directly performed on the smartphone, with no location-sensitive information being revealed to any type of service. The major drawback of the latter, however, is in terms of performance, since the data needed for localization can be potentially very large (e.g., WiGLE.net had 8.2 billion unique records by August, 2019).

In this chapter, we assume that ILS are fundamentally un-trusted and the already resource-constraint smartphone devices in terms of storage, energy and computational power will struggle to perform the indoor localization process locally on the device considering the ever-increasing volume of indoor location data. Therefore, we aim at examining a bio-inspired computing technique, that is an artificial neural network, for modeling the geolocation data at the server-side and then forwarding the model to the client-side to be used for predicting the exact user location. This will allow smartphone users to navigate in indoor environments with reduced resources consumption and high location accuracy without revealing their exact location at a central ILS, as this will be thoroughly explained in Sect. 4.

The rest of the chapter is organized as follows. Section 2 covers the background on indoor localization with smartphones and provides an overview on bio-inspired computing. Section 3 introduces related work on bio-inspired computing techniques and their applications on localization and navigation challenges. Furthermore, Sect. 4 presents the proposed ANN approach, which is then evaluated in Sect. 5. Finally, Sect. 6 summarizes our conclusions and introduces several future challenges.

2 Background

In this section, background on indoor localization that lies at the foundations of the proposed approach is initially introduced, followed by an overview of bio-inspired computing.

2.1 Indoor Localization with Smartphones

In the literature, there is a wide range of technologies dealing with smartphone device localization in outdoor and indoor environments. In outdoor environments, the Global Navigation Satellite Systems (GNSS), such as GPS and Galileo, are the

predominant technologies for localization. GNSS technologies, however, require high energy consumption and their weak satellite signal is often negatively affected from the environment (e.g., blocked by physical obstacles, cloudy days, dense urban areas, etc.). Besides GNSS, numerous proprietary solutions are also proposed by the localization community [1] such as *BLE sensors, visual or acoustic analysis, RFID, wireless sensor networks, laser and Li-Fi, IMUs* and their combinations into hybrid systems. Even though in most cases, these techniques provide a high localization accuracy, their drawback is the necessity of the deployment and calibration of additional equipment such as beacons, custom transmitters, proximity sensors, cameras dedicated to localization. The installation and calibration process of this equipment is often expensive in terms of both cost and time, while the approaches we discuss in this chapter mainly rely on conventional smartphones and wireless LANs already deployed in most buildings, as explained below:

(i) **Global Navigation Satellite Systems (GNSS)**: use radio signals broadcasted from satellites for localizing a device with high accuracy (which is often less than 1 m error). Since the localization process is carried out on the smartphone device, we consider that there are no privacy concerns with this approach. However, GNSS has an expensive energy tag and is unavailable or significantly demoted in indoor environments, due to the attenuation of the satellites signal strength [1].

(ii) **Cell/Wi-Fi Databases**: composed of radio signals retrieved from mobile cell towers and/or Wi-Fi access points (APs). They offer coarse accuracy that often spans from 1000 to less than 200 m error. The cell/Wi-Fi databases are often constructed offline by contributors (such as an Android smartphone user transmitting Wi-Fi AP and cell tower data to Google). In this case, smartphone users can request their current location from a cloud-based localization service. Subsequently, the localization process is mainly carried out on the server and consequently the service fundamentally violates a user's location privacy.

(iii) **Wi-Fi Fingerprints**: construct a database with radio signals from Wi-Fi APs similarly to (ii), but at a much higher density. In the literature, fingerprinting systems such as the Anyplace [5–7] achieved the second-highest known accuracy [8], with an average error of 1.96 m. Anyplace consists of two phases: The offline phase that utilizes a smartphone application, named "the logger," to record the so-called *Wi-Fi Fingerprints*, which are composed of *received signal strength indications (RSSi)* of Wi-Fi APs at certain locations (x, y) of a building. Thereinafter, the Wi-Fi Fingerprints are joint into an $N \times M$ matrix, commonly known as the *Wi-Fi RadioMap*, where N is the number of unique (x, y) fingerprints and M the total number of APs. In the online phase, a smartphone user observes its current RSS fingerprint and compares it against the RadioMap in order to find the best match, using known algorithms such as KNN and WKNN [9] that work as follows:

The *K nearest neighbor (KNN)* approach initially finds the K nearest fingerprints around the user's device using the Euclidean distance $d_u = ||V_i - V_u||, \forall V_i \in RM$ between the user's currently observed fingerprint V_u against

all fingerprints V_i in the RadioMap. Then, it calculates the user location using the convex combination of those K fingerprints. The *Weighted-KNN (WKNN)* approach works similarly to KNN with the difference that the K nearest neighbors are assigned a weight equal to:

$$w_i \propto \frac{1}{||V_i - V_u||}.$$

This assigns a more fair importance to the closest fingerprints and considers less the farthest fingerprints in the calculations; as opposed to KNN that assigns all fingerprints an equal importance (i.e., a weight equal to $w_i = 1/K$), which may decrease the localization accuracy since fingerprints that are far away may also be included in the calculation.

The RSS fingerprint approaches can be classified as follows:

(a) ***Server-Side (SS)* approach**: the bulk of the localization process is carried out by the indoor localization service (ILS) that has an unlimited energy, storage and processing budget. Therefore, the client-side requires little network messaging, minimal energy consumption and insignificantly small storage requirements. In this case, however, the *SS* fundamentally violates the smartphone user location privacy since the bulk of operation is carried out by the ILS.

(b) ***Client-Side (CS)* approach**: the smartphone user requests and downloads the Radiomap from the ILS and therefore the bulk of the localization process is carried out by the smartphone device. Therefore, the *CS* approach preserves the user location privacy, but unfortunately, downloading the whole RadioMap may result in high consumptions of the precious and limited smartphone battery, storage space and bandwidth, due to the fact that RadioMaps can potentially be very large.

The examined bio-inspired computing technique aims at alleviating all the aforementioned challenges of the *CS* and *SS* approaches by modeling the necessary data needed for indoor localization, i.e, the required RM at the server side and then forwarding the model to the client side where it will be used for predicting the exact user location. The prediction model will include the data of the whole RM and therefore will not allow the server to gain knowledge regarding the user's exact location and it will be of a much smaller size; decreasing in this way the resources consumption required for the localization process without deteriorating the localization accuracy.

2.2 Bio-inspired Computing: An Overview

Bio-inspired Computing (BIC) is the field that designs and develops computational algorithms and models inspired by biological mechanisms and living phenomena for effectively tackling real-life problems. BIC emerges from the combination of several fields such as mathematics, biology and computer science and more specifically

topics such as evolution, connectionism and social behavior. In the literature, BIC is often used to deal with artificial intelligence challenges by emulating the learning processes of biological organisms [10]. BIC techniques can be roughly classified as follows:

(i) **Bio-inspired optimization** techniques are based on the principles of biological systems to optimize the solution of a specific mathematical problem. In particular, real-life problems often have high complexity that makes it difficult to search for all possible solutions. Therefore, optimization algorithms are developed to mimic the biological evolution or behavior of animal/insect groups by defining deterministic or random rules that can be applied for approximating the optimal solution of such optimization problems. The two main bio-inspired optimization algorithms are:

Swarm Intelligence (SI) [11] refers to systems consisting of a population of simple agents that interact with each other and their environment. Interactions between agents lead to an intelligent global behavior of the population which is unknown to an individual. In particular, such systems are inspired by the collective behavior of self-organized systems [11]; having a variety of examples such as ant-based routing, human swarming, swarm grammars, swarmic art, crowd simulation, and swarm robotics. The SI approach finds applications on a vast amount of different areas including engineering, power control, medical, finance, etc., [12, 13].

Evolutionary Algorithms (EA) [14] use mechanisms inspired by biological evolution such as selection, recombination and mutation. EAs are defined by their population-based nature while evolving via simulated generations having applications [15, 16] in a variety of areas including natural sciences, earth sciences, finance and economics, social sciences, scheduling, etc.

(ii) **Bio-inspired Modeling** refers to biologically inspired computational models that aim to resolve challenging problems in an intelligent manner [17]. In particular, such models aim to propose unconventional architectures and novel problem paradigms. Artificial Neural Networks (ANNs) together with SI and EA (mentioned above) are considered as the major approaches for generating bio-inspired models.

Artificial Neural Networks (ANNs) [18] are networks of connected artificial neurons inspired by the composition of biological brains. In particular, ANNs attempt to perform tasks without being explicitly programmed. Contrary, they consider examples in order to find identifying characteristics from the learning material itself, without any prior knowledge about the material. The author of [19] provided a provisional definition as follows. "*A neural network is an interconnected assembly of simple processing elements, units or nodes, whose functionality is loosely based on the animal neuron. The processing ability of the network is stored in the interunit connection strengths, or weights, obtained by a process of adaptation to, or learning from, a set of training patterns*". Similarly to a biological brain, artificial neurons are able to transfer signals (inputs) to

the upcoming artificial neurons through connections or edges. In Feed-Forward ANN, which are the most widely used type, neurons are arranged in layers and the signal is transferred from the neurons of one layer to those of the next. Each neuron performs a simple transformation to the total signal it receives from the previous layer and sends the result as its output for the next layer. The outputs of the final layer, called *output layer*, neurons constitute the outputs of the ANN. The learning process adjusts the strengths of individual connections, thus increasing or decreasing the intensity of the signal transferred through them, in a way that brings the outputs of the output layer neurons closer to the desired output of the training patterns. The final trained ANN is obtained after many successive presentations of the training patterns and corresponding connection strength adjustments. These signals are used in a way to formulate non-linear functions that will perform different kinds of transformations on their input, by adjusting the weight of an input signal, in order to classify the input.

ANN models are used in a variety of areas including social network filtering, medical diagnosis, computer vision, and localization techniques. Some examples of ANN real-life applications are estrogen receptor status prediction for breast cancer [20], stroke risk estimation [21], stock market prediction [22], total electron content prediction [23], malicious URL detection [24], and android malware detection on smartphones [25].

3 Literature Review

Bio-inspired computing, as a general field with a wide variety of application areas, has been also used extensively for addressing navigation and localization challenges. The majority of the research studies in the literature deal with wireless sensor network node location estimation and robots/vehicles autonomous navigation and localization in outdoor and indoor environments. In most cases, bio-inspired computing approaches are used for image processing, self location estimation, and even robot control. Regarding indoor localization applications the focus is mainly on the estimation of a requested location and the optimization of the localization process. This section presents research studies related to applications of bio-inspired computing techniques for localization and navigation challenges in both indoor and outdoor environments, as summarized in Table 1.

3.1 Localization Using Bio-inspired Techniques

In [26], the simultaneous localization and mapping (SLAM) problem on mobile robot agents, which refers to tracking an agent while simultaneously construct a map of an unknown environment, has been tackled using a genetic algorithm (GA).

Table 1 Bio-inspired localization and navigation

References	Application area	Objective	Bio-inspired technique
[26]	Robotics	SLAM in robot control	Genetic algorithm
[27]	Robotics	SLAM in dynamic environments	Short-term memory neural networks
[28]	Robotics	SLAM in underwater environments	Continuous attractor neural networks
[29]	Tracking	Localization of moving magnetic Objects	Genetic algorithm
[30]	Robotics, autonomous vehicles	Autonomous vehicle localization model	Neural networks
[31–33]	Robotics	SLAM	RatSLAM
[34]	Wireless sensor networks	Autonomous deployment and localization of sensor nodes	Particle swarm optimization, Bacterial foraging algorithm
[35]	Wireless sensor networks	Localization of mobile anchor node	Centroid algorithm, Genetic algorithm
[36]	Wireless sensor networks	Sensor node localization scheme	Artificial neural networks, Genetic algorithm
[37]	Wireless sensor networks	Sensor node localization scheme	Genetic algorithm
[38]	Wireless sensor networks	DV-hop optimization	Genetic algorithm
[39]	Wireless sensor networks	Coverage and localization optimization	NSGA-II
[40]	Wireless sensor networks	Localization error optimization	Hybrid genetic algorithm-differential evolution
[41]	Wireless sensor networks	Localization as optimization problem	Salp swarm algorithm
[42]	Wireless sensor networks	Cooperative localization in industrial WSN	Dragonlfy algorithm, particle swarm optimization
[43]	Indoor localization	Fingerprinting	Boosting multi layer neural networks
[44]	Indoor localization	Fingerprinting algorithms' comparison	Types of recurrent neural networks
[45]	Indoor localization	Fingerprinting deep learning algorithms' comparison	Types of deep learning algorithms
[46]	Indoor localization	Fingerprinting with deep learning	Greedy deep learning algorithm
[47]	Indoor localization	Fingerprinting with deep learning	Deep neural networks
[48]	Indoor localization	Fingerprint-image localization	Convolutional neural networks
[49]	Indoor localization	Localization as optimization problem	Particle swarm optimization, JADE
[50]	Indoor localization	Localization as optimization problem	Particle swarm optimization
[51]	Indoor localization	Coverage versus energy consumption	Multi-objective evolutionary algorithm based on decomposition
[52]	Indoor localization	Hybrid fuzzy fingerprinting	Particle swarm optimization and gravitational search algorithm, neural networks

Similarly, [27] proposed a short-term memory (STM) neural network to deal with the SLAM problem in dynamic environments. Additionally, the authors in [28] proposed a continuous attractor neural network (CANN) to address the SLAM problem for underwater environments using visual references. In [29], a GA was tuned for providing fast and reliable solutions for localizing the trajectory of ferromagnetic moving objects within a bounded perimeter. A more recent research study [30] presented an application of autonomous vehicle navigation using artificial neural network models; the latter were trained over image and orientation of mobile robots. Finally, the research studies in [31–33] introduced several variations of the RatSLAM approach aiming at solving the visual SLAM problem on a wheel-chair robot and a humanoid robot, by utilizing different bio-inspired techniques.

The authors in [34] utilized both particle swarm optimization (PSO) and Bacterial foraging algorithm (BFA) for providing autonomous deployment and localization of wireless sensor nodes in the context of multi-objective optimization. Contrary, [35] proposed a received signal strength intensity (RSSI)-based localization technique of mobile anchor nodes using a centroid algorithm for initial estimation and a genetic algorithm (GA) for precise localization. The research studies in [36, 37] propose various RSSI-based localization schemes in the area of wireless sensor networks. In particular, [36] utilize a GA for initially adjusting the structure of an artificial neural network (ANN), which is then used for localization and [37] combine trilateration techniques with GA for error minimization during localization. The research study in [38] hybridized the DV-Hop range-free localization algorithm with a GA for improving the localization accuracy compared to existing algorithms. In [39], a NSGA-II algorithm is utilized for maximizing the coverage and optimizing the audio localization in a WSN, at the same time. In [40] a Hybrid GA with differential evolution (GADE) is proposed for minimizing the localization error of trilateration techniques. The authors of [41] propose a salp swarm algorithm (SSA) for dealing with the localization of sensor nodes in WSN. In [42], a Hybrid bio-inspired optimization algorithm for localization is proposed that combines a dragonfly algorithm (DA) with PSO. Here, it is important to note that none of the above research studies deal explicitly with smartphone user indoor localization.

3.2 Smartphone Indoor Navigation Using Bio-inspired Techniques

In [43], a boosting multi-layer neural network (MLNN) is proposed to improve the fingerprinting indoor localization technique in terms of localization accuracy. Similarly, [44] presents a comparison of several fingerprinting indoor localization approaches including recurrent neural networks (RNN), long short-term memory (LSTM), gated recurrent unit (GRU) and bidirectional LSTM (BiLSTM). The experimental evaluation demonstrated the superiority of the RNN approach with respect to the other conventional fingerprinting approaches. Both of the aforementioned

research studies have proposed fingerprinting techniques that reduce the localization error, but none examined the smartphone resources consumption and/or the smartphone user privacy violation.

The authors of [45] presented a deep learning-based fingerprinting localization schema, which was compared with deep neural network (DNN), deep belief network (DBN), and Gaussian-Bernoulli deep belief networks (GB-DBN) in terms of localization accuracy and generalization error reduction. A deep learning approach is also presented in [46] that utilizes deep architectures and channel state information (CSI) on fingerprinting for localization error minimization compared to existing techniques. Similarly, [47] presents a WiDeep approach, that is a deep ANN fingerprinting approach, which leverages from probabilistic denoising auto-encoder. All three research studies provide solutions for improving fingerprinting localization accuracy. However, the deep neural network consumes a high volume of resources during the training and localization phase. More recently, [48] proposes a hybrid fingerprint-image localization with convolutional neural networks (CNN) for representation of fingerprints as images. The results of this research work show the robustness of the framework, however, it is evident that image processing requires high computational effort compared to the conventional fingerprinting techniques.

An indoor localization optimization problem is formulated in [49] and tackled with a particle swarm optimization (PSO) approach, named JADE. Likewise, [50] proposed a fingerprinting localization and tracking system with PSO and Kalman filter (KF). The authors of [51] presented MILoS, a multi-objective indoor localization service that utilizes a Multi-Objective Evolutionary Algorithm Based on Decomposition (MOEA/D) to maximize the coverage and minimize the energy consumption, at the same time. The research study in [52] proposes a Hybrid Fuzzy Fingerprinting approach that trains ANN models composed of weight vectors optimized using particle swarm optimization and Gravitational Search Algorithm (FPSOGSA).

Finally, some other attempts that are strictly related to the propositions of this research study include [4] that proposes a simple heuristic on top of a bloom-filter data structure, named Temporal VectorMap (TVM), for providing fine-grain localization accuracy while preserving the users privacy and optimizing the performance issues mentioned in Sect. 2.1. Moreover, in [53], the *Grap (Graph Prefetching)* framework structurally analyzes building topologies to identify important areas, which are flagged as virtual targets and then an online heuristic that decides which areas to be downloaded and prefetched in the smartphone device for offline indoor navigation.

4 Indoor Localization Using Artificial Neural Networks

This section introduces the system model and presents an indoor localization approach that aims at improving the existing client-side fingerprinting technique by utilizing artificial neural networks.

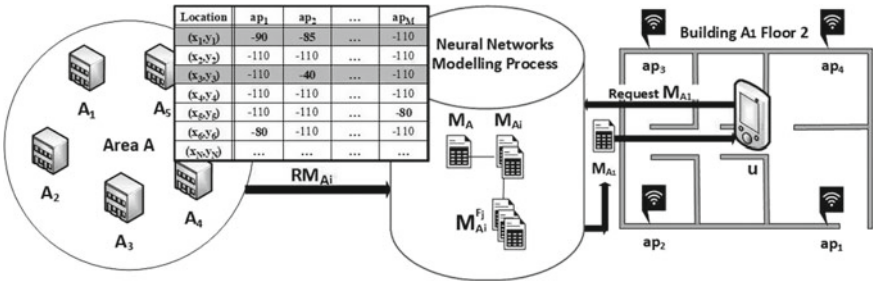


Fig. 1 Bio-inspired fingerprinting

4.1 System Model

We assume an area A divided into several building areas $A_1, .., A_n$, which consist of several floors $F_1, .., F_m$ (see Fig. 1). Each A_i and its corresponding F_j contain a finite set of (x, y) points and its covered by a set of Wi-Fi access points $\{ap_1, ap_2, \ldots, ap_M\}$, each covering a planar points. Area A_i is not necessarily continuous and can be considered as the joint area of all $ap_i \in AP$ (i.e., global coverage). Each ap_i has a unique ID (i.e., MAC address) that is publicly broadcasted and passively received by anyone moving in the a points of ap_i. The signal intensity at which the ID of ap_i is received at location (x, y), is termed the *Received Signal Strength (RSS)* of ap_i at (x, y), having a value in the range $[-30$ to $-110]$ dB.

Let a static (cloud-based) localization service s have constructed beforehand an $N \times M$ table, coined *RadioMap* (RM), which records the RSS of the $ap_i \in AP$ broadcasts at specified $(x, y) \in A$ locations. When an ap_i is not seen at a certain (x, y) the RM records "-110" in its respective cell. A user u localizes through the indoor positioning service s, using the ID and RSS broadcasts of surrounding $ap_i \in AP$ while moving. This information is termed, hereafter, *RSS Vector* or *Fingerprint* (V_u) of user u, which changes from location to location and over time. Contrary to RM rows having M attributes, V_u has only $M' \ll M$ attributes.

4.2 Research Goal

As mentioned in Sect. 2.1, conventional server-side (*SS*) and client-side (*CS*) techniques have some advantages but also face particular drawbacks. The *SS* approach violates the smartphone user privacy P_u since the whole process is performed at the server side, thus server has knowledge of the exact user location, but it requires minimum resources at the smartphone device. On the other hand, the *CS* approach alleviates the privacy violation concern, but requires high resources in terms of energy consumption, computational power and storage space since the whole radiomap RM

needs to be downloaded and stored on the smartphone device and all calculations should be performed locally on the smartphone for finding the user's exact location.

This research study aims at utilizing a bio-inspired modeling approach, that is an artificial neural network (*ANN*), for calculating prediction models M of various granularity from a subset of data RM' of the whole radiomap data RM that will be used for indoor localization, each providing different levels of user privacy and performance. In particular, a prediction model M_A will be calculated , for the subset radiomap data $RM' = RM_A$ of the whole area A, a M_{A_i} for the $RM' = RM_{A_i}$ of particular building areas A_i and even a $M_{A_i}^{F_j}$ for the $RM' = RM_{A_i}^{F_j}$ of a particular floor F_j of area A_i, at the server side s. Any of these models will be then forwarded to the user u upon request to be used for predicting the exact user location on the smartphone device.

Our research goal is to *preserve the user privacy while maintaining high localization accuracy (i.e., less localization error) and reduced resources consumption on the smartphone device, at the same time*.

Privacy Violation is the probability of the host server (s) guessing the user's (u) actual location, and it is given by

$$P_u = \frac{|RM'|}{|RM|} \tag{1}$$

where $|RM'|$ is the size of data either used by s for generating M or directly forwarded to u by s and RM is the whole radiomap.

The **Localization Error** is defined as the Haversine distance (in km), between the predicted location (Lat_2, Lon_2) and the actual location (Lat_1, Lon_1) of u,

$$d = 2r \times \arcsin\left(\sqrt{\sin^2\left(\frac{Lat_2 - Lat_1}{2}\right) + \cos(Lat_1)\cos(Lat_2)\sin^2\left(\frac{Lon_2 - Lon_1}{2}\right)}\right) \tag{2}$$

where $r = 6371$ km is the radius of earth.

Finally, the resources consumption includes the **storage space** $StS = |RM'| \times kb$, measured in kilobytes (kb), that is required by u to localize, and the **computational power** $CP = |RM'| \times t$ that refers to the processing time, in milliseconds (ms), on the smartphone device for performing a localization step. Note that the **energy consumption**, in joules, is equal to $E = (StS \times E1) + (CP \times E2)$, where $E1$ is the energy needed to download one kb of data on u and $E2$ the energy needed for processing one kb during the localization step.

For example, in Fig. 1, consider an *ANN* running at s and training various models M that correspond to the RMs of several areas, which can be used for indoor location predictions. A user u enters floor F_2 of building A_1 and desires to navigate toward a nearby point of interest. u uses a smartphone application to choose and download from s an indoor location prediction model based on personal preferences. That is an area level prediction model M_A for high user privacy, but with less location accuracy and higher resources consumption, or a building level prediction model M_{A_1}, .., M_{A_n}

that will provide less user privacy than before, since the server will increase the probability of guessing the exact user location, but with higher localization accuracy and less resources consumption. Finally, u requests M_{A_1} and s responds back with the requested prediction model. As a result, u navigates with a guaranteed building level privacy, since s just knows that u is in building A_1, with fine-grain accuracy (as this will be experimentally shown in the next evaluation Sect. 5), it consumes less E and StS, since $|M_{A_1}| \ll |RM_{A_1}|$, and requires less CP, since the prediction process is much faster than using the WKNN approach on the whole RM. In this way, the proposed technique alleviates all aforementioned challenges of the conventional fingerprinting techniques.

4.3 Radiomap Modeling Using Artificial Neural Networks

The radiomap modeling process is divided into three phases: Data preparation and Normalization, model training and model extraction.

The *data preparation and normalization* phase deals with the normalization of the radiomap data in order to be fed into an artificial neural network (ANNs). In particular, as explained before, radiomaps consist of crowdsourced data represented as vectors of RSSI values of surrounding Wi-Fi APs mapped to a specific location within an area. Furthermore, RSSI values are denoted in a range of -30 to -110 db, where the latter indicates a minimum intensity that cannot be sensed by the smartphone device. In addition, their corresponding location is defined in real values varying based on the area they represent. Therefore, since an area's coordinates will vary only at the least decimals, we eliminated the starting, common per area, decimals. This allowed our model to be trained while focusing on changes that matter to localization in the particular area. Additionally, the normalization minimized the rounding issues that might have occurred from the use of the original values.

During the *model training* phase, ANNs were trained to model the radiomaps of particular areas. The trained models are able to predict the user's current location based on a vector V_u that contains the RSSIs of the user's surrounding Wi-Fi APs, arranged in the same order for all patterns. The models used were fully connected feed-forward ANNs with sigmoid hidden and linear output activation functions and were trained with a stochastic gradient descent optimizer called Adam. Implementation was performed using the Scikit-Learn library in a Python environment with the ANNs regression algorithm MLPRegressor. Following a trial and error approach for the network's structure led to an average optimum scenario for similar-sized RMs. For example, floor-level RMs required a hidden layer consisting of 300 hidden units and| building level RMs required a hidden layer consisting of 1000 hidden units.

Finally, *model extraction* dealt with extracting the trained model in order to be used on a smartphone device. In particular, Scikit-Learn uses a Python tool to extract models commonly known as *pickle* or its updated version called *joblib*. Both tools extract unnecessary information along with the model itself, thing that led to larger files compared to the original RMs. Since one of our major objectives is to minimize

the storage space on the smartphone device, we instead followed a manual way for extracting the models. In particular, we extracted from the model its corresponding coefficients, and intercepts and saved them into two small files. On the smartphone device, we loaded the two files and reconstructed the model locally, a priori to localization. Note that this reconstruction does not imply re-training the model, since the coefficients, and intercepts are known from the training phase.

5 Experimental Evaluation

5.1 Datasets and Evaluation Metrics

For the experimental evaluation of our proposed bio-inspired indoor localization approach, we used a real dataset consisted of ≈45,000 reference fingerprints taken from ≈120 Wi-Fi APs installed in four floors of a building in Cyprus. The dataset gathered scales up to several GBs to ensure the validity of our approach. Furthermore, it contains five radiomaps of which one corresponds to the full building and the other to the four floors of the building. In addition, for the sake of this experimental evaluation, each radiomap was randomly divided into a *Train set* and a *Test set* representing 80% and 20% of the data, respectively. The models were trained on the training set, as explained in Sect. 4.3 and evaluated using the aforementioned test set through several experimental studies.

The proposed fingerprinting approach is evaluated in terms of *privacy violation*, *localization error*, *storage space*, and *computational power* on the client device, as introduced in Sect. 4.2.

5.2 Evaluation Results

This subsection compares the proposed ANN approach with the conventional weighted K nearest neighbors (WKNN) server-side (SS) and client-side (CS) approaches in terms of the performance metrics introduced in Sect. 4.2. Figure 2-right shows that the required *Storage Space* of WKNN SS approach is negligible, since most data is stored at the server. Contrary, the ANN approach requires less storage space than the WKNN CS approach. In particular, the proposed approach requires less storage space that ranges from ≈62% for the second floor to 1.69% for floor −1 of the reference building.

In terms of *Localization Error*, our experimental studies show that the proposed ANN approach maintained an acceptable level of localization error compared to both SS and CS, as demonstrated in Fig. 2-left. Moreover, it is also revealed that the ANN modeling approach improves storage space requirements and execution time at the smartphone device in the sake of a worst localization error. This trade-off can be

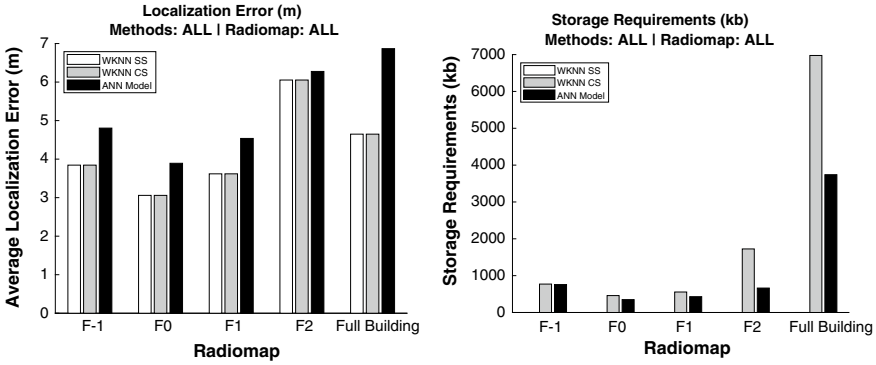


Fig. 2 Localization error and storage space requirements

monitored according to the user's preference. The localization error of the proposed approach ranges between 3.68% (which is approximately 0.27 m) at the best-case scenario to 47.74% (which is approximately 2.20 m) at the worst-case scenario.

Table 2 shows a comparison of the localization techniques in terms of average *Computational Power* (i.e., execution time) per localization step on the user's u smartphone device. It is important to notice that for WKNN SS, the majority of processing is performed at s and the execution time on u is negligible. The results show that the WKNN CS approach highly depends on the data size and, therefore, the execution time increases as the data size increases, as opposed to the proposed ANN approach that maintains a low execution time in all cases.

Finally in terms of *Privacy Violation*, Fig. 3 clearly shows that WKNN SS violates user privacy by 100% since s knows the exact user location. Contrary, the proposed ANN inherits the privacy levels of the WKNN CS approach due to the fact that s just knows the location data either utilized to create the prediction model, or requested and forwarded to the client side for performing the localization process locally on the smartphone device.

Table 2 Average computational power per localization step (in ms)

Radiomap	WKNN SS	WKNN CS	ANN
f-1	≈0	66.885	0.0138
f0	≈0	29.911	0.0104
f1	≈0	32.144	0.0048
f2	≈0	47.163	0.0079
Full building	≈0	392.603	0.027

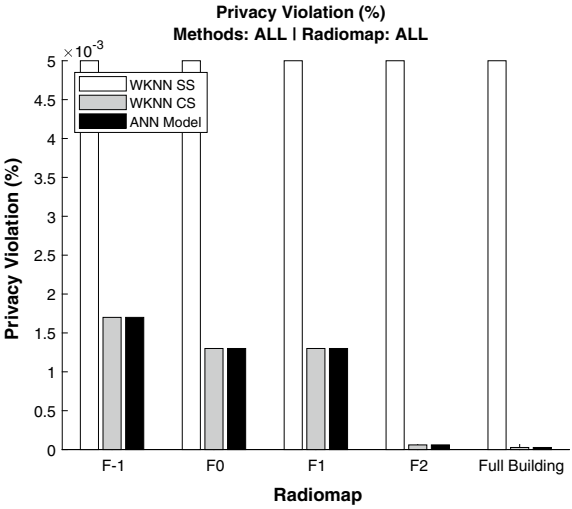


Fig. 3 Privacy violation

6 Conclusions and Future Challenges

In this chapter, a variety of localization and navigation challenges tackled by bio-inspired techniques are initially reviewed and discussed. Then a bio-inspired indoor localization fingerprinting approach that utilizes artificial neural networks (ANNs) is proposed. The ANN is used to model RMs of various granularity at the server-side, which are then forwarded to the smartphone user upon request for offline indoor navigation. The experimental results demonstrate the superiority of the proposed approach when compared with conventional client-side and server-side fingerprinting approaches, in terms of resources consumption, while preserving user privacy and maintaining localization accuracy. Some future challenges on smartphone user indoor localization and the applicability of bio-inspired techniques are follow.

The single objective optimization of the performance objectives introduced in Sect. 5 (that is storage space and energy consumption, computational power and privacy) using metaheuristics such as genetic algorithms (GAs) can be a promising extension. Another future challenge along that direction could be the hybridization of single objective optimization techniques (e.g., GAs) with problem-specific local search heuristics (commonly known as memetic algorithms) for a more effective search of the objective space. Additionally, it is clearly shown from the experimental studies of this chapter that the indoor localization problem is composed of multiple conflicting objectives. For example, at the client-side fingerprinting approaches, the more localization data are forwarded from the server to the client, the better the localization accuracy and the privacy preservation, but more resources consumption is required by the smartphone device. Therefore, the simultaneous optimization of multiple objectives requires a problem formulation in the context of multi-objective optimization (MOO) than needs to be tackled using techniques such as multi-objective evolutionary algorithms.

Moreover, artificial neural network (ANN) optimization is an emerging field due to the need of selecting the best values of multiple parameters, often achieved with a trial and error process. In the literature, many research studies utilize optimization techniques for optimizing the values of ANN parameters and/or determining the weights of ANN models. Therefore, a future direction would have been to utilize evolutionary algorithms for optimizing the ANN parameters and investigate whether this benefits the objectives of the proposed indoor localization challenge. In particular, the optimization of the ANN parameters using an EA at the server side may decrease the number of neurons in the prediction model and therefore decrease the size of data required to be forwarded to the client side for localizing without affecting the accuracy and the privacy guarantees. This will be, therefore, beneficial for the storage space, energy consumption and computational power needed by the smartphone device.

Furthermore, a dimensionality reduction step before the ANN model training phase is effective for removing redundant and irrelevant data. As a future work, dimensionality reduction techniques can be examined with the proposed indoor localization approach in order to eliminate the dimensions of the model's input data, based on their relevance to the localization process (e.g., eliminate data of non-sensed Wi-Fi APs). This will result in further reducing the model size and consequently improve smartphone resource preservation.

Acknowledgements This work is part of the "EnterCY" Integrated Project with project number INTEGRATED/0609/0020, which is funded by the Research & Innovation Foundation (RIF) of Cyprus.

References

1. Gu Y, Lo A, Niemegeers I (2009) A survey of indoor positioning systems for wireless personal networks. IEEE Commun Surv Tutor 11:13–32
2. Peter A, Tella Y, Dams G (2015) An overview of indoor localization technologies and applications. J Comput Technol 4(5):2278–3814
3. Petrou L, Larkou G, Laoudias C, Zeinalipour-Yazti D, Panayiotou CG (2014) Crowdsourced indoor localization and navigation with anyplace. In: Proceedings of the 13th international conference on information processing in sensor networks, IPSN'14. IEEE Press, Berlin, 15–17 Apr 2014, pp 331–332
4. Konstantinidis A, Zeinalipour-Yazti GCD, Mpeis P, Pelekis N, Theodoridis Y (2015) Privacy-preserving indoor localization on smartphones. IEEE Trans Knowl Data Eng 27(11):3042–3055
5. Zeinalipour-Yazti D, Laoudias C (2017) The anatomy of the anyplace indoor navigation service. SIGSPATIAL Spec 9:3–10
6. Georgiou K, Constambeys T, Laoudias C, Petrou L, Chatzimilioudis G, Zeinalipour-Yazti D, (2015) Anyplace: a crowdsourced indoor information service. In 2015 16th IEEE international conference on mobile data management, vol 1, June 2015, pp 291–294
7. Laoudias C, Constantinou G, Constantinides M, Nicolaou S, Zeinalipour-Yazti D, Panayiotou C (2012) The airplace indoor positioning platform for android smartphones. In: Proceedings of the 13th IEEE International Conference on Mobile Data Management (MDM '12), IEEE Computer Society, July 2012, pp 312–315, Bangalore, India, ISBN: 978-0-7695-4713-8
8. Lymberopoulos D, Liu J, Yang X, Choudhury RR, Handziski V, Sen S, Lemic F, Buesch J, Jiang Z, Zou H, Jiang H, Zhang C, Ashok A, Xu C, Lazik P, Rajagopal N, Rowe A, Ghose A,

Ahmed N, Hevesi P (2015) A realistic evaluation and comparison of indoor location technologies: experiences and lessons learned. In: IPSN 2015 - Proceedings of the 14th International Symposium on Information Processing in Sensor Networks, April 2015, pp 178–189
9. Li B, Salter J, Dempster AG, Rizos C (2006) Indoor positioning techniques based on wireless lan. In: LAN, First IEEE International Conference on Wireless Broadband and Ultra Wideband Communications, pp 13–16
10. Rai D, Seth K (2013) Bio-inspired optimization techniques: a critical comparative study. ACM SIGSOFT Softw Eng Notes 38:1–7
11. Beni G, Wang J (1993) Swarm intelligence in cellular robotic systems. In: Dario P, Sandini G, Aebischer P (eds) Robots and biological systems: towards a new bionics? Springer, Berlin, pp 703–712
12. Ma R-J, Yu N-Y, Hu J-Y (2013) Application of particle swarm optimization algorithm in the heating system planning problem. Sci World J 2013
13. Alrashidi M, El-Hawary M (2009) A survey of particle swarm optimization applications in electric power systems. IEEE Trans Evol Comput 13:913–918
14. Vikhar PA (2016) Evolutionary algorithms: a critical review and its future prospects. In: International Conference on Global Trends in Signal Processing, Information Computing and Communication (ICGTSPICC), Dec 2016, pp 261–265
15. Liu Q, Cai W, Shen J, Fu Z, Liu X, Linge N (2016) A speculative approach to spatial-temporal efficiency with multi-objective optimization in a heterogeneous cloud environment. Secur Commun Netw 9(17):4002–4012
16. Ma X (2016) Intelligent tourism route optimization method based on the improved genetic algorithm. In: 2016 international conference on smart grid and electrical automation (ICSGEA), Aug 2016, pp 124–127
17. Sajja PS, Akerkar R (2013) 12-bio-inspired models for semantic web. In: Yang X-S, Cui Z, Xiao R, Gandomi AH, Karamanoglu M (eds) Swarm intelligence and bio-inspired computation. Elsevier, Oxford, pp 273–294
18. Hassoun MH (1995) Fundamentals of artificial neural networks, 1st edn. MIT Press, Cambridge
19. Gurney K (1997) An introduction to neural networks. Taylor & Francis Inc., Bristol
20. Dhondalay GK, Tong DL, Ball GR (2011) Estrogen receptor status prediction for breast cancer using artificial neural network. In: 2011 international conference on machine learning and cybernetics, vol 2, July 2011, pp 727–731
21. Papadopoulos H, Kyriacou E, Nicolaides A (2017) Unbiased confidence measures for stroke risk estimation based on ultrasound carotid image analysis. Neural Comput Appl 28:1209–1223
22. Chen W, Zhang Y, Yeo CK, Lau CT, Lee BS (2017) Stock market prediction using neural network through news on online social networks. In: 2017 international smart cities conference (ISC2), Sept 2017, pp 1–6
23. Konstantinidis A, Haralambous H, Agapitos A, Papadopoulos H (2011) A gp-moea/d approach for modelling total electron content over cyprus. CoRR, abs/1111.5720
24. Shivangi S, Debnath P, Saieevan K, Annapurna D (2018) Chrome extension for malicious urls detection in social media applications using artificial neural networks and long short term memory networks. In: 2018 international conference on advances in computing, communications and informatics (ICACCI), Sept 2018, pp 1993–1997
25. Papadopoulos H, Georgiou N, Eliades C, Konstantinidis A (2018) Android malware detection with unbiased confidence guarantees. Neurocomputing 280:3–12 (Applications of neural modeling in the new era for data and IT)
26. Duckett T (2013) A genetic algorithm for simultaneous localization and mapping. In: IEEE International Conference on Robotics and Automation, Mar 2003
27. Li Y, Li S, Ge Y (2013) A biologically inspired solution to simultaneous localization and consistent mapping in dynamic environments. Neurocomputing 104:170–179
28. Guth FA, Silveira L, Amaral M, Botelho S, Drews P (2013) Underwater visual 3d slam using a bio-inspired system. In: 2013 symposium on computing and automation for offshore shipbuilding, Mar 2013, pp 87–92

29. Alimi R, Weiss E, Ram-Cohen T, Geron N, Yogev I (2015) A dedicated genetic algorithm for localization of moving magnetic objects. Sensors (Basel, Switz) 15:23788–23804
30. Espada Y, Cuperlier N, Bresson G, Romain O (2018) Application of a bio-inspired localization model to autonomous vehicles. In: 2018 15th international conference on control. In: Automation, robotics and vision (ICARCV), pp 7–14
31. Milford MJ, Wyeth GF, Prasser D (2004) Ratslam: a hippocampal model for simultaneous localization and mapping. In: IEEE international conference on robotics and automation, 2004. Proceedings. ICRA'04. 2004, vol 1, Apr 2004, pp 403–408
32. Müller S, Weber C, Wermter S (2014) Ratslam on humanoids–a bio-inspired slam model adapted to a humanoid robot. In: Wermter S, Weber C, Duch W, Honkela T, Koprinkova-Hristova P, Magg S, Palm G, Villa AEP (eds) Artificial neural networks and machine learning–ICANN 2014. Springer International Publishing, Cham, pp 789–796
33. Milford M, Jacobson A, Chen Z, Wyeth G (2016) RatSLAM: using models of rodent hippocampus for robot navigation and beyond, Apr 2016, pp 467–485
34. Kulkarni RV, Venayagamoorthy GK (2010) Bio-inspired algorithms for autonomous deployment and localization of sensor nodes. IEEE Trans Syst Man Cybern Part C Appl Rev 40:663–675
35. Huanxiang J, Yong W, Xiaoling T (2010) Localization algorithm for mobile anchor node based on genetic algorithm in wireless sensor network. In: 2010 international conference on intelligent computing and integrated systems, Oct 2010, pp 40–44
36. Chagas SH, Martins JB, de Oliveira LL (2012) An approach to localization scheme of wireless sensor networks based on artificial neural networks and genetic algorithms. In: 10th IEEE international NEWCAS conference, June 2012, pp 137–140
37. Panchapakesan A (2010) An efficient genetic algorithm based localization scheme for wireless sensor networks. In: National Conference on Recent Trends in VLSI, Information and Communication, Feb 2010
38. Peng B, Li L (2015) An improved localization algorithm based on genetic algorithm in wireless sensor networks. Cogn Neurodyn 9:249–256
39. Mnasri S, Thaljaoui A, Nasri N, Val T (2015) A genetic algorithm-based approach to optimize the coverage and the localization in the wireless audio-sensors networks. In: 2015 international symposium on networks, computers and communications (ISNCC), May 2015, pp 1–6
40. Srideviponmalar P, Kumar VJS, Ramachandran H (2018) Hybrid genetic algorithm-differential evolution approach for localization in WSN, Jan 2018, pp 263–271
41. Babayigit B, Nayyef H, Rudaini H (2018) Salp swarm algorithm for localization of wireless sensor networks, Oct 2018
42. Babayigit B, Nayyef H, Rudaini H (2018) Salp swarm algorithm for localization of wireless sensor networks. In: 3rd International Mediterranean Science and Engineering Congress, Çukurova University, Adana, Turkey, Oct 2018
43. Dai H, Ying W, Xu J (2016) Multi-layer neural network for received signal strength-based indoor localisation. IET Commun 10(6):717–723
44. Hoang M, Yuen B, Dong X, Lu T, Westendorp R, Reddy K (2019) Recurrent neural networks for accurate rssi indoor localization. arXiv:1903.11703 [online], Mar 2019
45. Félix G, Siller M, Álvarez EN (2016) A fingerprinting indoor localization algorithm baseddeep learning. In: 2016 eighth international conference on ubiquitous and future networks (ICUFN), July 2016, pp 1006–1011
46. Wang X, Gao L, Mao S, Pandey S (2017) Csi-based fingerprinting for indoor localization: a deep learning approach. IEEE Trans Veh Technol 66:763–776
47. Abbas M, Elhamshary M, Rizk H, TorkiM, Youssef M (2019) Wideep: WiFi-based accurate and robust indoor localization system using deep learning. In: Percom 2019, Jan 2019
48. Shao W, Luo H, Zhao F, Ma Y, Zhao Z, Crivello A (2019) Indoor positioning based on fingerprint-image and deep learning. In: IEEE Access, Nov 2018, vol 6, pp 74699–74712
49. Bergenti F, Monica S (2019) A bio-inspired approach to wifi-based indoor localization. In: Cagnoni S, Mordonini M, Pecori R, Roli A, Villani M (eds) Artificial life and evolutionary computation. Springer International Publishing, Cham, pp 101–112

50. Ding G, Tan Z, Wu J, Zeng J, Zhang L (2015) Indoor fingerprinting localization and tracking system using particle swarm optimization and kalman filter. IEICE Trans Commun E98.B
51. Pericleous S, Konstantinidis A, Demetriades A (2019) A multi-objective indoor localization service for smartphones. In: ACM Symposium on Applied Computing, Apr 2019
52. Rohra JG, Perumal B, Thakur SJNP, Bhatt R (2017) User localization in an indoor environment using fuzzy hybrid of particle swarm optimization and gravitational search algorithm with neural networks, Feb 2017, pp 286–295
53. Konstantinidis A, Irakleous P, Georgiou Z, Zeinalipour-Yazti D, Chrysanthis PK (2018) Iot data prefetching in indoor navigation soas. In: ACM transactions on internet technology'18, Dec 2018

Chapter 8
A New Obstacle Avoidance Technique Based on the Directional Bat Algorithm for Path Planning and Navigation of Autonomous Overhead Traveling Cranes

Asma Chakri, Amar Skendraoui, Rabia Khelif and Haroun Ragueb

1 Introduction

Overhead traveling cranes (OTC) are being used in many industries worldwide. These cranes operate under different conditions and some of these operating conditions can be dangerous to humans. Though most cranes are operators by human operators, safety requirements make it necessary to remove human operators completely from any potential dangerous conditions. Consequently, autonomous overhead traveling cranes for such operating conditions become a necessity in practice.

The developments of such autonomous cranes have been the subject of several research studies. Terashima and Suzuki [1] proposed a navigation system and path planning for an overhead traveling crane, based on a strategy derived from the resolution of the diffusion equation. The latter reflects the diffusion process of chemical species in a medium, and its basic idea is to assimilate the movement of the crane to the spread of chemical elements. This method is an off-line method, which requires some prior knowledge of the OTC working environment. Mandelli and Haider [2] proposed a methodology to convert old cranes of a cement plant that exists in the

A. Chakri (✉) · H. Ragueb
Laboratory of Energy and Mechanical Engineering, Faculty of Engineering Sciences, University M'Hamed Bougara of Boumerdes (UMBB), Avenue of Independence, 35000 Boumerdes, Algeria
e-mail: chakri.as623@gmail.com

A. Chakri · R. Khelif
Industrial Mechanics Laboratory, Department of Mechanical Engineering, University Badji Mokhtar of Annaba (UBMA), Sidi Ammar, BP12-23000 Annaba, Algeria

A. Skendraoui
Department of Technology, Higher School of Industrial Technologies of Annaba (ESTI—Annaba), City of Saf-Saf, 23000 Annaba, Algeria

X.-S. Yang and Y.-X. Zhao (eds.), *Nature-Inspired Computation in Navigation and Routing Problems*, Springer Tracts in Nature-Inspired Computing,
https://doi.org/10.1007/978-981-15-1842-3_8

USA, to fully automated ones. The monitoring of these cranes for a period of six years shows that the maintenance costs decreased by 30%.

Akamatsu et al. [3] introduced an online path planning method where they used the potential method based on the resolution of the diffusion equation. In addition, ultrasonic sensors have been added to detect small changes in the working space, to update the path planning system. Omar et al. [4] developed an autonomous OTC system using a fuzzy control system which mimics the thought of an expert crane operator in setting some of the key control parameters. Wecker et al. [5] developed an obstacle detection system using three cameras installed at the hook to enable an OTC to move without collision. Miyoshi et al. [6] presented a path planning method with consideration of the hook rotation. This technique is very useful when handling long objects such as pipes, steel profiles, containers and others.

Kaneshige et al. [7] proposed an improvement to the control method presented in [3] by including a load swinging suppression system to allow the transport of an open container filled with liquid. Smoczek et al. [8] developed an obstacle recognition system using two cameras to create a 3D reconstruction of the environment, and then, they used the A^* search algorithm to bypass obstacles. Yang et al. [9] developed a vision system for obstacle avoidance using surveillance cameras installed at the trolley. This system recognizes the hook load and creates or delimits a safety zone. If, during the movement of the OTC, an object or obstacle enters the safety zone, the system will send an alarm signal and stops.

Metaheuristic algorithms have been used extensively in many applications, including applications to develop autonomous mobile robots for obstacle avoidance using genetic algorithm [10], particle swarm optimization (PSO) [11, 12], bacterial mimetic algorithm (BMA) [13], and more recently the invasive weed optimization (IWO) [14], harmony search algorithm (HS) [15] and cuckoo search algorithm (CS) [16]. These algorithms have been widely used to solve engineering problems. Sivakumar et al. [17] applied the A^* algorithm for the path planning of two mobile cranes working in cooperation. Ali et al. [18] solved the same problem by using the genetic algorithm. Wang et al. [19] used the ant colony algorithm (ACA) for planning obstacle avoidance of a mobile crane in a complex environment. Despite the success in robots and mobile cranes control, the implementation of these algorithms for automating the OTC remains an unexploited research area.

The bat algorithm is a new bio-inspired metaheuristic algorithm, introduced by Xin-She Yang in 2010 [20]. Due to its ease implementation and efficiency, this algorithm had attracted much intention, and it has been used to solve a diverse range of engineering problems. Despite that, many studies reported that this algorithm may suffer from the premature convergence problem that can occur under certain conditions; therefore, several improved variants have been proposed to improve the exploitation and exploration capability of the algorithm [21–25].

Recently, an improved version of the bat algorithm, called the directional bat algorithm (dBA), has been introduced by Chakri et al. [26], and since, it has been used to solve different engineering problems [27–32]. The dBA makes use of the directional echolocation characteristics used by micro-bats to define and guide the direction of the next movement. By integrating this feature, the exploration and

exploitation capabilities of the algorithm have significantly enhanced, and the results showed that the directional bat algorithm performs better than several bat algorithm variants such as [21–25] and other state-of-the-art algorithms such as the restart CMA evolution strategy with the increasing population size (IPOP-CMA-ES) [33] and the self-adaptive differential evolution (SaDE) [34].

In this chapter, we develop a new methodology for the path planning of autonomous overhead traveling cranes with automatic obstacle avoidance. A new online obstacle avoidance mechanism based on the directional bat algorithm is introduced. In the next section, the classical bat algorithm and the directional bat algorithm are described briefly with the emphasis on the key features in addition to a short review on recent advance in BA improvement. In Sect. 3, the proposed strategy of the path planning with obstacle avoidance is presented in detail. In Sect. 4, simulation results are presented and analyzed. Finally, conclusions are drawn and discussed in Sect. 5.

2 BA, dBA and Variants

2.1 The Standard Bat Algorithm

The standard bat algorithm was first developed by Xin-She Yang [20] through observation of micro-bats flying behavior. As the micro-bats are primarily blind, they use echolocation process to fly through dark, search for preys and navigation in different environments. To translate this behavior into meaningful mathematical algorithm, Yang proposed the following idealized rules:

1. All bats use echolocation to sense distance, and the location of a bat x_i is encoded as a solution to an optimization problem under consideration [20].
2. Bats fly randomly with velocity v_i at position x_i with a varying frequency (from a minimum $f_{\min}$ to a maximum frequency $f_{\max}$) or a varying wavelength λ and loudness A to search for prey. They can automatically adjust the wavelengths (or frequencies) of their emitted pulses and the rate of pulse emission r depending on the proximity of the target [20].
3. Loudness varies from a large positive value A_0 to a minimum constant value $A_{\min}$ [20].

In a d-dimensional space, each bat should have its position and velocity defined (x_i and v_i, respectively), and when flying, they are updated as follows [20]:

$$f_i = f_{\min} + \left(f_{\max} - f_{\min}\right)\text{rand} \tag{1}$$

$$v_i^{t+1} = v_i^t + \left(x^* - x_i^t\right) f_i \tag{2}$$

$$x_i^{t+1} = x_i^t + v_i^{t+1} \tag{3}$$

with rand ∈ [0,1] is a random vector, and x^* is the current global best location among the bat swarm. In addition to guided navigation defined by Eqs. (1)–(3), each bat is allowed to have a local detour using the random walk process given by:

$$x_{\text{new}} = x_{\text{old}} + \varepsilon < A^{t+1} > \tag{4}$$

where $\varepsilon \in [-1, 1]$ is a random number, and $< A_i^{t+1} >$ is the average loudness of all the bats at this time step.

The loudness A_i and the rate of pulses emission r_i are updated during the navigation and the hunting process of bats. As they approach their preys, the loudness decreases, and the pulse rate increases according to the following:

$$A_i^{t+1} = \alpha A_i^t, \tag{5}$$

$$r_i^{t+1} = r_i^0 \left[1 - \exp(-\gamma t)\right] \tag{6}$$

where $0 < \alpha < 1$ and $\gamma > 0$ are constants. As $t \to \infty$, we have $A_i^t \to 0$ and $r_i^t \to r_i^0$. The initial loudness A_0 can typically be $A_0 \in [1, 2]$, while the initial emission rate $r^0 \in [0, 1]$. The pseudo-code of the standard bat algorithm is presented in Algorithm 1. The global convergence of the standard BA has been investigated and proved by Chen et al. [35]. They build a Markovian model of BA, and the analysis showed that the bat swarm constitutes a finite homogeneous Markov chain that satisfies the criterion of global convergence.

Algorithm 1. The standard bat algorithm

01. Define the objective function
02. Initialize the bat population $L_i \leq x_i \leq U_i$ (i = 1,2,..,n) and v_i
03. Define frequencies f_{min} and f_{max}
04. Initialize pulse rates r_i and loudness A_i
05. ***While*** ($t \leq t_{max}$)
06. Adjust frequency, Eq. (1).
07. Update velocities, Eq. (2).
08. Move the bats using Eq. (3).
09. ***if*** ($rand > r_i$)
10. Generate a local solution around the selected best solution, Eq. (4).
11. ***end if***
12. ***if*** ($rand < A_i$ & $F(x_i) < F(x^*)$)
13. Accept the new solutions
14. Reduce A_i, Eq. (5).
15. Increase r_i, Eq. (6).

16. ***end if***
17. Rank the bats and find the current best x^*
28. ***end while***
29. Results processing.

2.2 Recent Advance in Improving the Bat Algorithm

To improve exploration and exploitation capabilities of the standard bat algorithm, and thus its efficiency and reliability, several authors have proposed new improved variants of BA in the last few years. A survey of the state-of-the-art bat algorithm variants can be found in [36–39]. Recently, Cai et al. [40] proposed introducing the triangle-flipping strategy to update the velocity defined by Eq. (2). Three kinds of flipping strategy have been investigated, namely directing strategy, random strategy and hybrid strategy between the two previous ones. The performances of the new bat algorithm with different flipping strategies were verified using the CEC2013 benchmark functions, and the results showed that the use of the hybrid triangle-flipping strategy has better outcomes. Shan and Cheng [41] proposed to replace Eqs. (1)–(3) that controls the bat global search by a hybrid strategy using: one, the covariance matrix adaptive evolution strategy (CMA-ES [42]), two, sinusoidal function for updating frequency similar to the one used in SinDE [43]. The yielded improved bat algorithm, shortly called MCMABA, has been tested on several classical benchmark functions where most of them have $x_i = 0$ as a solution, and however, the examination of the convergence curves shows that algorithm reaches an acceptable solution with few iterations even for high-dimensional problem ($d = 30$ in this case). This happens when the equation controlling the bat movements has the tendency to converge to $x_i = 0$ within few iterations [38], in this case, we suspect that this behavior is caused by Eqs. 13 and 14 in Ref. [41]. The same observation can be applied to the BA variant proposed by Gan et al. [44] which is based on stochastic inertia weight and iterative local search (ILSSIWBA). In this case, the iterative locale search, controlled by Eq. 7 in Ref. [44], has the tendency to converge also to $x_i = 0$ in few iterations.

Liu et al. [45] suggested to incorporate three modifications to the main flowchart of BA in order to improve its performances. The first modification is related to the first stage of population initialization. The second modification concerns the location updating process, whereas the third modification is about adding external optimization using either predatory search [46], or the algorithm proposed by Chen et al. [47] to improve PSO. The authors examined performances of various combinations of the three modifications, and they found that the simultaneous use of the three modifications with predatory search as an external optimization strategy gives the best results. Lyu et al. [48] developed an improved self-adaptive variant of BA with step-control and mutation mechanism (SABA). The main idea of SABA consists

of defining a new self-adaptive flying mechanism and executing mutation operation in sometimes when local search is not performed. This modification allows better space exploration due to mutation and enhanced exploitation by step control. Reddy and Ganguli [49] suggested that in order to improve BA performances, modification should be carried on the BA structure. They mainly adopted the dBA flowchart [26] and introduced several modifications inspired by variants of PSO [50, 51] and BA [52, 53]. The algorithm was tested using several benchmark functions and compared to the outcomes of different algorithms. The results showed significant enhancement compared to the classical bat algorithm.

Wang et al. [54] proposed a novel BA with different strategies, exactly eight, that control the bat movements. In addition to the original BA equations of movements, Eqs. (1)–(3), one strategy was from an improved BA variant [55], two strategies were based on Lévy flight mechanism [56, 57], one strategy is the GA two-point crossover operator, another is based on PSO equations, and lastly, the local search and the local disturbance strategies. Each of these strategies has a selection probability that increases as the strategy produces better results during the iteration process. The proposed algorithm has been tested on CEC2013 benchmark, and the results were very quite satisfying, and however, after the insertion of so many strategies and modification, the algorithm becomes crowded with setting parameters that have tuned carefully. Bekdas et al. [58] incorporated the structure dynamic equations on the BA framework so that the algorithm carries out an optimal tuning of mass dampers to improve the seismic safety of structures. The algorithm was tested using a ten-story civil structure benchmark and compared to the outcomes of different algorithms such as PSO and HS and others. The comparison showed that the novel proposed BA can achieve better results.

2.3 The Directional Bat Algorithm

The new directional bat algorithm (dBA) has introduced four major modifications [26], and one of the main improvements is to introduce the directional echolocation behavior to the standard bat algorithm, with the aim to enhance its exploitation and exploration capabilities so as to improve its performance. The dBA uses a similar pseudo-code as that of the standard bat algorithm, though the new modifications have been incorporated in the updating equations. In [26], the new directional bat algorithm was developed by considering the following assumptions:

- Each bat knows the position of the others and can somehow know if the food exists around each bat or not.
- Each bat emits two pulses into two different directions before choosing in which direction it will go.
- All the bats emit a pulse in the direction of the best bat (solution) where the food is considered to exist and the other pulse to the direction of a randomly chosen bat.

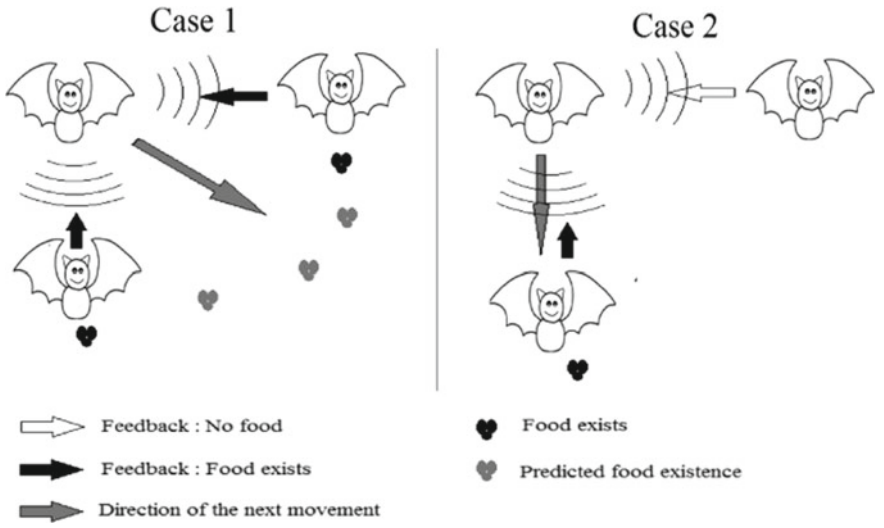


Fig. 1 Hypothetical schema of the directional echolocation behavior of bats

As shown in Fig. 1, a bat emits two pulses in two different directions, one to the direction of randomly selected bat, and the other in the direction of the bat with best position (the best solution). From the echoes (feedback), the bat can know if the food exists around these two bats or not. Usually, around the bat with the best position the food exists (because it has the best fitness value), but around the randomly selected bat, it depends on its objective fitness value. If it has a better fitness value as the actual bat, then the food is considered to exist; otherwise, there is not a food source in the neighborhood.

If the food was confirmed to exist around the two bats (case 1), the current bat moves to a direction at the surrounding neighborhood of the two bats. If not (case 2), it moves toward the best bat. The mathematical formulas of the bat movements are:

$$\begin{cases} X_i^{t+1} = X_i^t + (X^* - X_i^t) f_1 + (X_k^t - X_i^t) f_2 & (\text{if } F(X_k^t) < F(X_i^t)) \\ X_i^{t+1} = X_i^t + (X^* - X_i^t) f_1 & \text{Otherwise} \end{cases} \tag{7}$$

where X_k^t is the location of randomly selected bat ($k \neq i$), and X^* is the best solution. $F(\,)$ is the fitness function that is associated with the objective value of the optimization problem under consideration. f_1 and f_2 are the frequencies of the two pulses updated as the following:

$$\begin{cases} f_1 = f_{\min} + (f_{\max} - f_{\min})\text{rand1} \\ f_2 = f_{\min} + (f_{\max} - f_{\min})\text{rand2} \end{cases} \tag{8}$$

where two random vectors rand1 and rand2 are drawn from a uniform distribution $U(0, 1)$.

In addition, the bats are allowed to move from their current positions to new random positions generated by the following equation:

$$X_i^{t+1} = X_i^t + < A^t > \varepsilon \, w_i^t \tag{9}$$

where $<A^t>$ is the average loudness of all bats, and $\varepsilon \in [-1, 1]$ is a random vector. Here, w_i is a parameter applied to reduce the space of the allowed movement, while the iterative process proceeds. It is updated as follows:

$$w_i^t = \left(\frac{w_{i0} - w_{i\infty}}{1 - t_{\max}} \right)(t - t_{\max}) + w_{i\infty} \tag{10}$$

Here, w_{i0} and $w_{i\infty}$ are the initial and final values that w_i can take during iterations. In general, we can set w_{i0} and $w_{i\infty}$ as follows:

$$w_{i0} = (Ub_i - Lb_i)/4 \tag{11}$$

$$w_{i\infty} = w_{i0}/100 \tag{12}$$

where t is the current iteration, and $t_{\max}$ is the maximum number of iterations. Ub_i and Lb_i are the upper and lower bounds.

The formula proposed by Yang [20] to update the pulse rate and loudness reaches their final value during the iterative process quickly, thus reducing the possibility of the auto-switch from the random walk to the local search. Therefore, Chakri et al. [26] have proposed to use these monotonically increasing pulse rate and decreasing loudness, respectively:

$$r^t = \left(\frac{r_0 - r_\infty}{1 - t_{\max}} \right)(t - t_{\max}) + r_\infty \tag{13}$$

$$A^t = \left(\frac{A_0 - A_\infty}{1 - t_{\max}} \right)(t - t_{\max}) + A_\infty \tag{14}$$

where the index 0 and ∞ stand for the initial and final values, respectively.

The final improvement is to allow the bats to update the pulse rate and loudness and to accept a new solution if their movement produces a solution better than the old one instead of the global best solution as it is in the original algorithm. This modification was also suggested by Hasançebi et al. [52]. The flowchart of the directional bat algorithm is presented in Fig. 2.

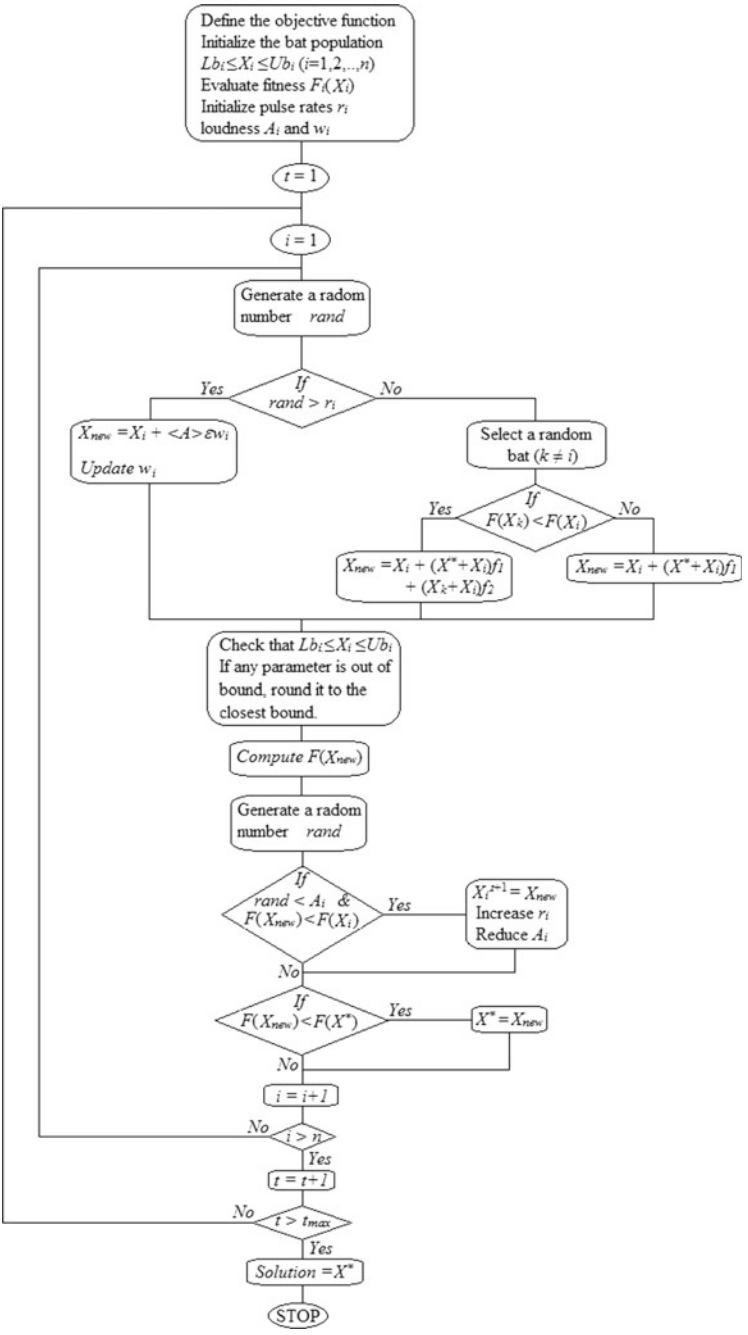


Fig. 2 Flowchart of the dBA

3 The Proposed Strategy for OTC Autonomous Path Planning

In order to investigate the possibility of an automatic OTC system using the dBA approach, we first assume that the hangar height of the overhead traveling crane is limited, which can reduce the simulations in 2D. We thus consider only the movement according to the width (trolley movement) and length (bridge movement). In addition, we assume that the control system of the crane has no preliminary knowledge of the position of the obstacles and the working environment; however, the obstacles detection is done online via proximity sensors on the limited periphery and can provide the (x,y) coordinates of the obstacle. We also assume that only the start position and the target are known. Moreover, for purely numerical reasons and to facilitate simulations, we assume that the OTC movement is done step by step in the workspace, and the trajectory is a set consisting of control points $\{P_s, P_1, P_2, \ldots, P_n, P_t\}$ where P_s and P_t represent the starting point and the target point, respectively, while the P_i $(i = 1\text{–}n)$ are the positions of the trolley in the 2D space.

Theoretically speaking, the optimum path and the shortest path between two points is a straight line if there is no obstacle. The truck crane moves from its current position to the new position following a straight path. If we fix the origin of the coordinate system (Oxy) at a corner of the hangar, the path described by the trolley displacement can be obtained as follows:

$$\begin{cases} P_{i+1} = P_i + \text{sign}(P_t - P_i)\begin{pmatrix} \sin(\alpha) \\ \cos(\alpha) \end{pmatrix} R \\ P_t = \begin{pmatrix} x_t \\ y_t \end{pmatrix} \text{ and } P_i = \begin{pmatrix} x_i \\ y_i \end{pmatrix} \\ tg(\alpha) = \begin{pmatrix} \frac{x_a - x_i}{y_a - y_i} \end{pmatrix} \end{cases} \tag{15}$$

where R represents the step, and x and y are the trolley coordinates. P_{i+1} is the next position, and P_i is the previous one.

During its movement, the OTC is equipped with an obstacle detection system with a radius (R_0), which describes a perimeter called the *safety perimeter*. In order that the crane moves without any collision, the step R should be smaller than R_0. If an obstacle $P_o\,(x_o, y_o)$ is within the safety perimeter, a signal is sent to the control system for path correction. The path correction mechanism essentially forms a constrained optimization problem with the objective function being the minimal distance from the next position P_{i+1} to the target, subject to the constraint that keeps a distance greater than or equal to R_0 to the obstacle. Therefore, the optimization problem can be defined as follows:

$$\begin{cases} \text{Minimize } d_t = \sqrt{(x_t - x_{i+1})^2 + (y_t - y_{i+1})^2} \\ \text{Subject to} \\ \min(d_{oj}) = \min\left(\sqrt{(x_{oj} - x_{i+1})^2 + (y_{oj} - y_{i+1})^2}\right) \geq R_0 \\ x_i - R \leq x_{i+1} \leq x_i + R \\ y_i - R \leq y_{i+1} \leq y_i + R \end{cases} \tag{16}$$

The index i represents the current position, while $i + 1$ represents the next move.

The search space with a solution to such a problem inside the safety perimeter called *the secure search space*, and this secure search space can be explored effectively by dBA. We opted for a square search space with a length equal to $2R$ to simplify the optimization problem and thus reduce the complexity of the search space (thus reducing the computation time). If we had defined a circular search space with a radius of R, we would need to add a second constraint which increases the computation time. Therefore, to enable the crane to move autonomously without collision with obstacles, we propose this framework illustrated in Fig. 3 to control the movement of the overhead traveling crane.

To better understand the obstacle avoidance mechanism, Fig. 4 shows a hypothetical schema of the proposed control system in this paper. Initially, the user provides the starting and target positions. The crane begins to move to the target, following a

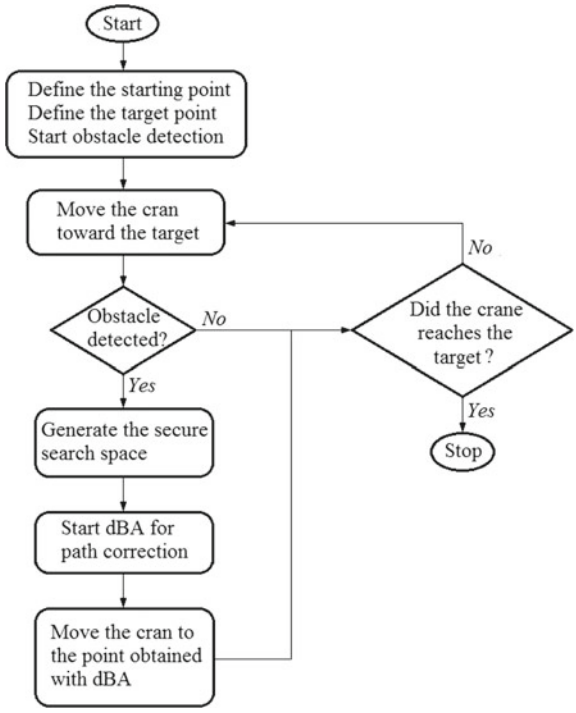


Fig. 3 Framework of the crane's movement

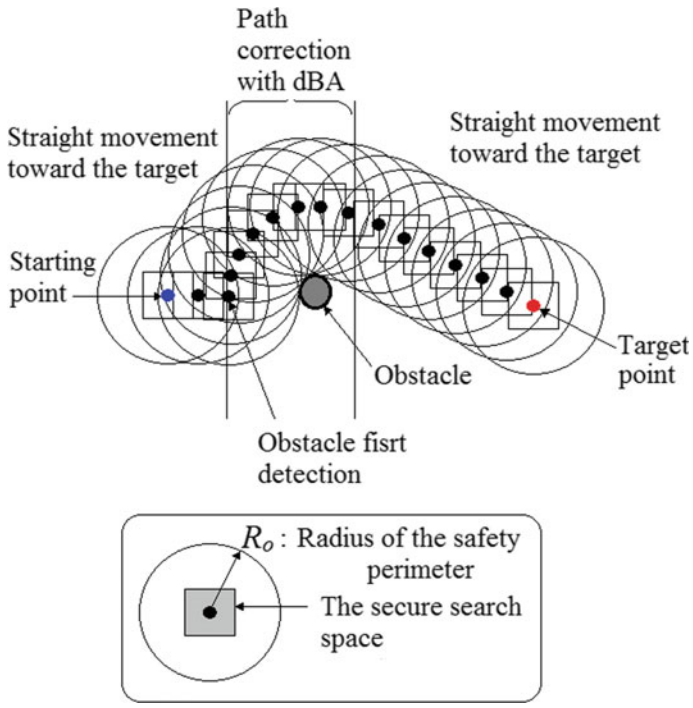


Fig. 4 Hypothetical schema of the obstacle avoidance mechanism

straight path. If, during its movement, an obstacle is detected at the edge of the safety perimeter, the path correction system is activated. In this phase, the path correction system uses the dBA to find the direction of the next move. This is repeated as it is needed until the crane going around the obstacle. When the obstacle is out of the security perimeter, the OTC continues to move toward the target in a straight path.

4 Simulation, Results and Discussions

To test the minimization efficiency of the new directional bat algorithm, a comparison with several standard algorithms, namely the standard bat algorithm (BA) [20], particle swarm optimization (PSO) [51], differential evolution (DE) [59] and genetic algorithm (GA) [60], on benchmark multimodal functions has been carried out. The considered benchmark functions are: the spherical function, Rosenbrock's function, Rastrigin's function, Griewank's function, Ackley's function, Alpine's function, Weierstrass' function and finally Salomon's function. The mathematical details of the previous functions are described in Appendix.

Since the obstacle avoidance problem is a two-dimensional problem, we consider the minimization of these functions in a two-dimensional space ($D = 2$). In addition, for a fair comparison, the common parameters of the algorithms must be the same, we set the population size $N = 10$ and the number of iterations $t_{\max} = 20$. The parameter settings of each algorithm are listed in Table 1. Each algorithm was run 100 times, and the mean and the standard deviation of the obtained minimum of each run are summarized in Table 2. As we can see, dBA obtained the best mean for every problem compared with the standard algorithms. This is due to the embedded modifications which had improved both exploration and exploitation capabilities of the algorithm. In addition, dBA has a better ability to find a feasible solution with a low number of function evaluations. That means that dBA can be potentially used in low-specification hardware and can still work efficiently.

Table 1 Algorithms' parameters setting

Algorithms	Settings
dBA	$r_0 = 0.1, r_\infty = 0.7, A_0 = 0.9, A_\infty = 0.6, f_{\min} = 0$ an $f_{\max} = 2$
BA	$r_0 = 0.1, A_0 = 0.9, \alpha = \gamma = 0.9, f_{\min} = 0$ and $f_{\max} = 2$
PSO	$c_1 = 1.5, c_2 = 1.2$ and w is a monotonically decreasing from 0.9 to 0.4
DE	"DE/rand/1/bin" strategy with CR = rand[0.2, 0.9] and F = rand[0.4, 1]
GA	Crossover probability = 0.95 and mutation probability = 0.05

Table 2 Comparison between dBA and the standard algorithms on benchmark functions

Function		dBA	BA	PSO	DE	GA
Spherical	Mean	0.181001	446.1696	4.548414	1.511587	69.84505
	StD	1.308170	494.3148	14.17570	2.655406	263.1806
Rosenbrock	Mean	1.112144	57.17420	3.171311	2.108372	198.2591
	StD	1.987260	210.2229	5.178383	2.564265	811.0343
Rastrigin	Mean	1.757334	7.796023	3.039737	1.775359	9.197475
	StD	1.184982	4.946270	2.612374	1.207610	5.411830
Griewank	Mean	0.136200	0.637118	0.186973	0.141056	0.470444
	StD	0.080813	0.380729	0.135418	0.080684	0.327793
Ackley	Mean	1.432040	12.17725	2.863056	1.951879	5.610177
	StD	1.151961	4.694127	2.082912	1.131155	3.336199
Alpine	Mean	0.000561	0.028292	0.001443	0.000710	0.014127
	StD	0.000407	0.036066	0.002611	0.000423	0.014031
Weierstrass	Mean	0.018404	0.134584	0.042181	0.021645	0.147582
	StD	0.011680	0.054456	0.035523	0.011339	0.060173
Salomon	Mean	0.351917	49.64252	0.973821	0.593340	10.71349
	StD	0.279611	46.87526	1.555780	0.932064	37.21352

StD standard deviation

Table 3 Means and standard deviations of results obtained by different algorithms

	dBA	BA	PSO	DE	GA
Min	6.435292	6.4353704	6.4363972	6.4353745	6.4391189
Mean	6.437134	6.4457429	6.4427854	6.4389122	6.4757661
Max	6.440841	6.4641947	6.4582331	6.4801772	6.5500459
StD	0.001259	0.0059314	0.0040632	0.0047332	0.0239345

StD Standard deviation

In the second test, we use some other effective algorithms such as the standard bat algorithm (BA), particle swarm optimization (PSO), differential evolution (DE) and genetic algorithm (GA) to solve the problem described in Eq. (16) with $x_i = (5.4\text{ m}, 14.5\text{ m})^{\mathrm{T}}$, $x_o = (6.75\text{ m}, 15\text{ m})^{\mathrm{T}}$, $x_t = (9\text{ m}, 20\text{ m})^{\mathrm{T}}$, $R_0 = 1.5$ m and $R = 0.1$ m. In this problem, we assume that the crane is in the position x_i near the obstacle x_o, and we want to find the next position x_{i+1}, with the minimum distance to x_t, while avoiding collision with any obstacle in the neighborhood. With the same parameter configuration as the previous test, the results are presented in Table 3. Each algorithm was run 100 times, and the best, the mean, worst and the standard deviation (StD) are presented. As we can see, dBA achieves better results compared to other algorithms. The differences between dBA and the others algorithms are around 1 and 2 cm, which cannot be considered as negligible because these differences can accumulate with the increase in the number of navigation points that have to be generated by the algorithm to avoid obstacles and the number of obstacles. As a result, the paths generated by dBA have the lowest distance between the starting point and the target point.

Consider now an overhead traveling crane with a width (x) of 22.5 m and length (y) 50 m. To simulate the movements of the OTC, we assume that we want to move an object from its current position P_s ($x_s = 1$ m, $y_s = 2$ m) to the other corner at P_t ($x_t = 20$ m, $y_t = 45$ m). The distance between the two points is 47.01 m. Here, we will consider three tests with different complexities. In the first phase, we will put an obstacle P_{o1} in the straight path between P_t and P_s, to examine the behavior of the proposed algorithm. After that, we add a second obstacle P_{o2} in the path of the trolley according to the new corrected trajectory. In the third phase, and to increase the complexity of the environment, we add five more obstacles deliberately put in the path of the crane. The coordinates of each obstacle are shown in Table 4.

We assume that the diameter of the object that we want to move is 1 m; therefore, to allow to the OTC to move without collision, we set $R_0 = 1.5$ m and $R = 0.1$ m. The same parameter settings of dBA used in the previous experiment for benchmarking

Table 4 Obstacles' coordinates

	P_{o1}	P_{o2}	P_{o3}	P_{o4}	P_{o5}	P_{o6}	P_{o7}
x_{oj} (m)	6.75	8	7	10	13	15	18
y_{oj} (m)	15	20	20	22	25	35	40

is used here, in addition to $N = 10$ and $t_{max} = 20$. In the first test, one obstacle has been deliberately put on the straight path between the starting point and the target. The path generated by the OTC movement is illustrated in Fig. 5. As it can be seen, in the first place, the crane moves toward the target in straight path. When it reaches a certain distance to the obstacle (1.5 m), the crane detects the existence of the obstacle and engages the auto-correction of the path with dBA. As a result, it bypasses successfully the obstacle and continues its movement toward the target in a straight line. In the second test, another obstacle has been put on the trajectory after correction. Figure 6 shows that the proposed method enables the OTC to avoid the second obstacle.

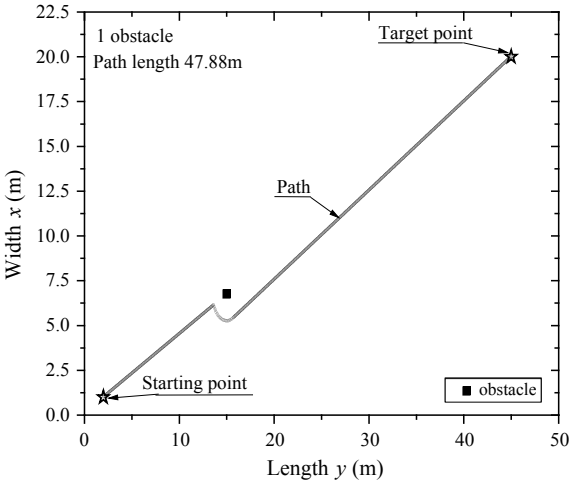


Fig. 5 OTC's path with one obstacle

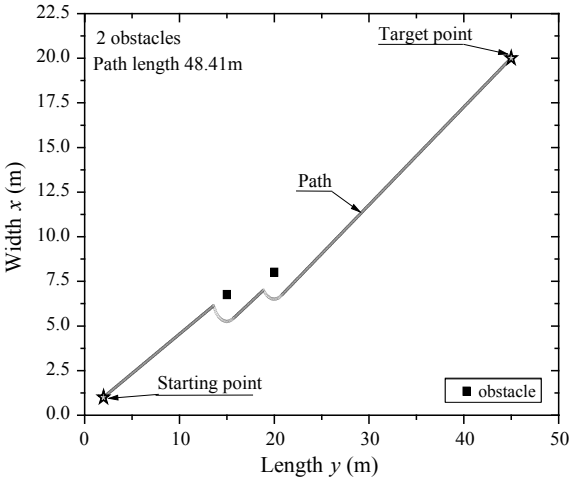


Fig. 6 OTC's path with two obstacles

In the third test, seven obstacles have been imposed on the path of the OTC. As it is shown in Fig. 7, the OTC finds its way easily and avoids all the obstacles smoothly. We also observe that the proposed method can handle any new obstacle and can avoid any closely placed obstacles, as like what happens near the obstacles (e.g., obstacles 2 and 3 in Fig. 7).

Figure 8 represents the presumed trajectory if there were no obstacles or after bypass an obstacle. It is clear that when the crane bypasses an obstacle, it moves in a straight line toward the target. Figure 9 shows the way back path in the same environment. As we can see, it is different from the other path to go. This is due to the fact that the obstacles were intentionally placed in the path of the OTC whenever

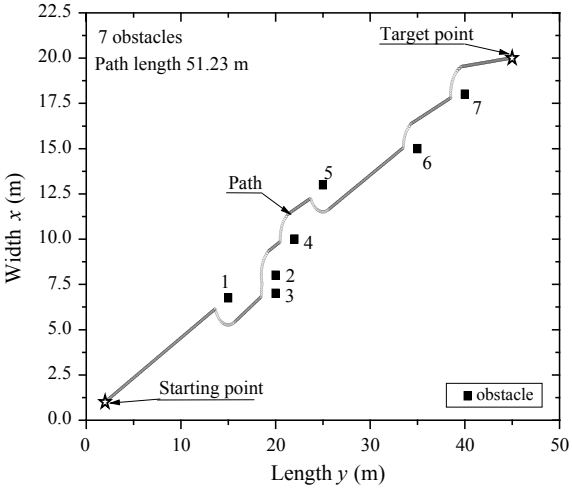


Fig. 7 OTC's path with seven obstacles

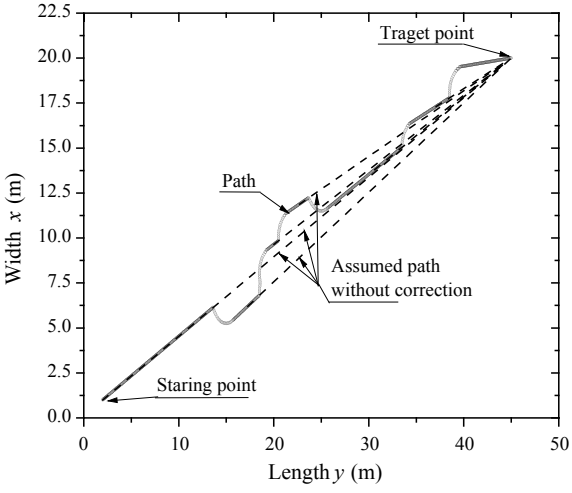


Fig. 8 Assumed path without correction

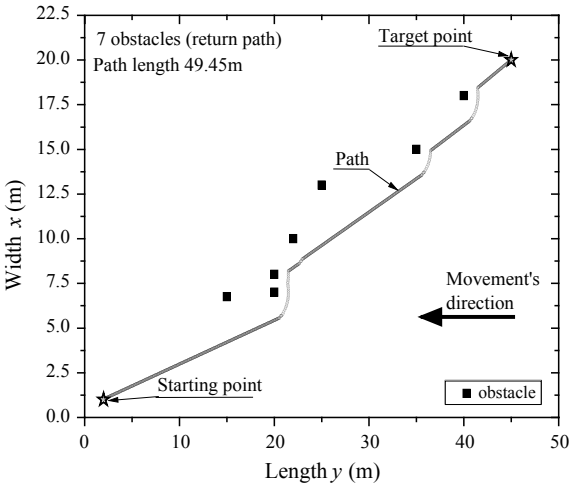


Fig. 9 Way back path with the same obstacles

it bypasses an obstacle for switch-on the obstacle avoidance mechanism. It is worth pointing that it is impossible that the two paths be the same because the proposed algorithm does not take into consideration the complete topology of the environment and only proximity sensors are used to detect obstacles during the movement of the overhead traveling crane.

In the last test, we have assumed the existence of a large obstacle in the way of the OTC. The obstacle is detected by sensors and considered as a set of multiple obstacle points. This set of points is then fed to the algorithm for path planning. The generated path is presented in Fig. 10. As we can see, the proposed method can bypass large objects without colliding.

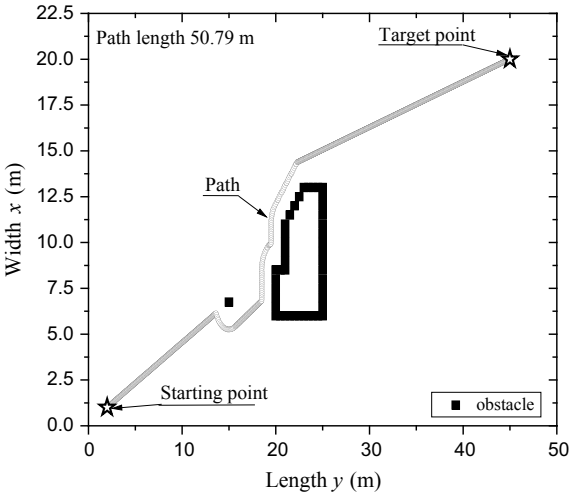


Fig. 10 Actual path taken by the crane to avoid large obstacles or barriers

5 Conclusions

In this paper, a new path planning method for autonomous OTC with obstacles avoidance has been developed, based on the directional bat algorithm. This new method can enable the crane to be autonomous and can reduce the operating costs with improved work efficiency. The results show that this technique allows the OTC to bypass obstacles easily. In addition, from the application viewpoint, this technique is less expensive than others that exist in the literature because it does not require prior knowledge of the working environment (a prior knowledge of environment requires general mapping of all workplace and updates every time there is change, subsequently sophisticated equipment must be installed). In our proposed method, proximity sensors (e.g., ultrasonic, lasers or cameras) can be installed within the allowed budget. In addition, due to the low number of iterations and the smaller bat population, this method can be implemented on low-end, low-priced hardware (processor and memory).

For further works, it will be very interesting to extend this method to 3D path planning with obstacle avoidance for high OTC systems, such as tower cranes and the gantry cranes used in harbor and ports. Another interesting field is to validate and analyze the proposed method in a dynamic environment with moving obstacles. In addition, the application of the proposed method to other navigation problems to avoid obstacles can be also very fruitful.

Appendix

(1) Spherical function:

$$F_1 = \sum_{i=1}^{d} x_i^2 \tag{17}$$

where $-100 < x_i < 100, f_{\min} = 0$ and $x_{\text{opt}} = [0, 0]$.

(2) Rosenbrock's function:

$$F_2 = \sum_{i=1}^{d-1} \left[100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2 \right] \tag{18}$$

where $-10 < x_i < 00, f_{\min} = 0$ and $x_{\text{opt}} = [1]$.

(3) Rastrigin's function:

$$F_3 = 10d + \sum_{i=1}^{d} \left[x_i^2 - 10\cos(2\pi x_i) \right] \tag{19}$$

where $-5.12 < x_i < 5.12, f_{\min} = 0$ and $x_{\text{opt}} = [0, 0]$.

(4) Griewank's function:

$$F_4 = \sum_{i=1}^{d} \frac{x_i^2}{4000} - \prod_{i=1}^{d} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \tag{20}$$

where $-100 < x_i < 100, f_{\min} = 0$ and $x_{\text{opt}} = [0, 0]$.

(5) Ackley's function:

$$F_5 = -20\exp\left(-0.2\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}\right) - \exp\left(\frac{1}{d}\sum_{i=1}^{d}\cos(2\pi x_i)\right) + 20 + \exp(1) \tag{21}$$

where $-32 < x_i < 32, f_{\min} = 0$ and $x_{\text{opt}} = [0, 0]$.

(6) Alpine's function:

$$F_6 = \sum_{i=1}^{d} |x_i \sin(x_i) + 0.1x_i| \tag{22}$$

where $-10 < x_i < 10, f_{\min} = 0$ and $x_{\text{opt}} = [0, 0]$.

(7) Weierstrass' function:

$$F_7 = \sum_{i=1}^{d}\left(\sum_{k=0}^{20}\left[0.5^k \cos\left(2\pi \cdot 3^k (x_i + 0.5)\right)\right]\right) - d\sum_{k=0}^{20}\left[0.5^k \cos\left(2\pi \cdot 3^k \cdot 0.5\right)\right] \tag{23}$$

where $-0.9 < x_i < 0.9, f_{min} = 0$ and $x_{opt} = [0, 0]$.

(8) Salomon's function:

$$F_8 = 1 - \cos\left(2\pi \sum_{i=1}^{d} x_i\right) + 0.1\sum_{i=1}^{d} x_i^2 \tag{24}$$

where $-100 < x_i < 100, f_{\min} = 0$ and $x_{\text{opt}} = [0, 0]$.

References

1. Terashima K, Suzuki M (1999) Path planning and navigation of overhead traveling crane with three-dimensional transport based on a diffusion equation strategy. In: Advances in manufacturing. Springer, pp 335–347

2. Mandelli V, Haider A (1999) Converting existing overhead cranes to a fully automatic operation. In: Cement industry technical conference, 1999. Conference record. 1999 IEEE-IAS/PCA, 1999. IEEE, pp 357–375
3. Akamatsu T, Kaneshige A, Terashima K Real time path planning based on the potential method for an autonomous mobile overhead traveling crane. In: 2004 IEEE international symposium on industrial electronics. IEEE, pp 699–704
4. Omar F, Karray F, Basir O, Yu L (2004) Autonomous overhead crane system using a fuzzy logic controller. J Vib Control 10(9):1255–1270
5. Wecker T, Aschemann H, Hofer E (2005) Sensor-based collision avoidance for rope-suspended autonomous material flow systems. In: World congress, pp 2030–2030
6. Miyoshi T, Kawakami S, Terashima K (2008) Path planning and obstacle avoidance considering rotary motion of load for overhead cranes. J Mech Syst Transp Logistics 1(1):134–145
7. Kaneshige A, Miyoshi T, Terashima K (2009) The development of an autonomous mobile overhead crane system for the liquid tank transfer. In: IEEE/ASME international conference on advanced intelligent mechatronics, AIM 2009. IEEE, pp 630–635
8. Smoczek J, Szpytko J, Hyla P (2013) Non-collision path planning of a payload in crane operating space. In: Solid state phenomena, 2013. Trans Tech Publ, pp 559–564
9. Yang J, Chien W, Huang M, Tsai M (2015) Application of machine vision to collision avoidance control of the overhead crane. Paper presented at the international conference on electrical, automation and mechanical engineering Phuket, Thailand
10. Hu Y, Yang SX (2004) A knowledge based genetic algorithm for path planning of a mobile robot. In: 2004 IEEE international conference on robotics and automation, 2004. Proceedings of ICRA'04. IEEE, pp 4350–4355
11. Wang L, Liu Y, Deng H, Xu Y (2006) Obstacle-avoidance path planning for soccer robots using particle swarm optimization. In: IEEE international conference on robotics and biomimetics, 2006. ROBIO'06. IEEE, pp 1233–1238
12. Lu L, Gong D (2008) Robot path planning in unknown environments using particle swarm optimization. In: Fourth international conference on natural computation, ICNC'08. IEEE, pp 422–426
13. Botzheim J, Toda Y, Kubota N (2012) Bacterial memetic algorithm for offline path planning of mobile robots. Memetic Comput 4(1):73–86
14. Mohanty PK, Parhi DR (2014) A new efficient optimal path planner for mobile robot based on invasive weed optimization algorithm. Front Mech Eng 9(4):317–330
15. Kundu S, Parhi DR (2016) Navigation of underwater robot based on dynamically adaptive harmony search algorithm. Memetic Comput 8(2):125–146
16. Mohanty PK, Parhi DR (2015) A new hybrid optimization algorithm for multiple mobile robots navigation based on the CS-ANFIS approach. Memetic Comput 7(4):255–273
17. Sivakumar P, Varghese K, Babu NR (2003) Automated path planning of cooperative crane lifts using heuristic search. J Comput Civ Eng 17(3):197–207
18. Ali MAD, Babu NR, Varghese K (2005) Collision free path planning of cooperative crane manipulators using genetic algorithm. J Comput Civ Eng 19(2):182–193
19. Wang X, Zhang Y, Wu D, Gao SD (2011) Collision-free path planning for mobile cranes based on ant colony algorithm. Key Eng Mater 467–469:1108–1115
20. Yang X-S (2010) A new metaheuristic bat-inspired algorithm. In: González J, Pelta D, Cruz C, Terrazas G, Krasnogor N (eds) Nature inspired cooperative strategies for optimization (NICSO 2010), vol 284. Studies in computational intelligence. Springer, Berlin Heidelberg, pp 65–74
21. Alam MS, Kabir MWU (2014) Bat algorithm with self-adaptive mutation: a comparative study on numerical optimization problems. Int J Comput Appl 100(10):7–13
22. Fister IJ, Fong S, Brest J, Fister I (2014) A novel hybrid self-adaptive bat algorithm. Sci World J 709–738
23. Nguyen T-T, Pan J-S, Dao T-K, Kuo M-Y, Horng M-F (2014) Hybrid bat algorithm with artificial bee colony. In: Pan J-S, Snasel V, Corchado ES, Abraham A, Wang S-L (eds) Intelligent data analysis and its applications, vol II, vol 298. Advances in intelligent systems and computing. Springer International Publishing, pp 45–55

24. Yilmaz S, Küçüksille EU (2015) A new modification approach on bat algorithm for solving optimization problems. Appl Soft Comput 28:259–275
25. Jordehi RA (2015) Chaotic bat swarm optimisation (CBSO). Appl Soft Comput 26:523–530
26. Chakri A, Khelif R, Benouaret M, Yang XS (2017) New directional bat algorithm for continuous optimization problems. Exp Syst Appl 69(C):159–175
27. Mukherjee S, Reddy MP, Ganguli R, Gopalakrishnan S (2018) Ply level uncertainty effects on failure curves and optimal design of laminated composites using directional bat algorithm. Int J Comput Methods Eng Sci Mech 19(3):156–170
28. Chakri A, Yang X-S, Khelif R, Benouaret M (2018) Reliability-based design optimization using the directional bat algorithm. Neural Comput Appl 30(8):2381–2402
29. Chakri A, Khelif R, Benouaret M (2016) Improved bat algorithm for structural reliability assessment: application and challenges. Multidiscipline Model Mater Struct 12(2):218–253
30. Guamán PM, Guerrero-Vasquez LF, Bermeo JP, Chasi PA (2018) Metaheuristic optimization algorithms of swarm intelligence in patch antenna design. In: 2018 IEEE 10th Latin-American conference on communications (LATINCOM), 14–16 Nov 2018, pp 1–6
31. Haji Haji V, Monje CA (2018) Fractional-order PID control of a MIMO distillation column process using improved bat algorithm. Soft Comput
32. Chakri A, Khelif R, Benouaret M (2017) Optimization of the box-girder of overhead crane with constrained new bat algorithm. Synthèse: Revue des Sciences et de la Technologie 35(1):187–203
33. Auger A, Hansen N (2005) A restart CMA evolution strategy with increasing population size. Paper presented at the 2005 IEEE congress on evolutionary computation, Edinburgh, Scotland
34. Qin AK, Suganthan PN (2005) Self-adaptive differential evolution algorithm for numerical optimization. In: The 2005 IEEE congress on evolutionary computation, Edinburgh, Scotland, vol 1782, pp 1785–1791
35. Chen S, Peng G-H, He X-S, Yang X-S (2018) Global convergence analysis of the bat algorithm using a markovian framework and dynamical system theory. Expert Syst Appl 114:173–182
36. Yang X-S (2013) Bat algorithm: literature review and applications. arXiv preprint arXiv: 13083900
37. Chawla M, Duhan M (2015) Bat algorithm: a survey of the state-of-the-art. Appl Artif Intell 29(6):617–634
38. Chakri A, Ragueb H, Yang X-S (2018) Bat algorithm and directional bat algorithm with case studies. In: Yang X-S (ed) Nature-inspired algorithms and applied optimization. Springer International Publishing, Cham, pp 189–216
39. Jayabarathi T, Raghunathan T, Gandomi AH (2018) The bat algorithm, variants and some practical engineering applications: a review. In: Yang X-S (ed) Nature-inspired algorithms and applied optimization. Springer International Publishing, Cham, pp 313–330
40. Cai X, Wang H, Cui Z, Cai J, Xue Y, Wang L (2018) Bat algorithm with triangle-flipping strategy for numerical optimization. Int J Mach Learn Cybernet 9(2):199–215
41. Shan X, Cheng H (2018) Modified bat algorithm based on covariance adaptive evolution for global optimization problems. Soft Comput 22(16):5215–5230
42. Hansen N, Ostermeier A (2001) Completely derandomized self-adaptation in evolution strategies. Evol Comput 9(2):159–195
43. Draa A, Bouzoubia S, Boukhalfa I (2015) A sinusoidal differential evolution algorithm for numerical optimisation. Appl Soft Comput 27:99–126
44. Gan C, Cao W, Wu M, Chen X (2018) A new bat algorithm based on iterative local search and stochastic inertia weight. Expert Syst Appl 104:202–212
45. Liu Q, Wu L, Xiao W, Wang F, Zhang L (2018) A novel hybrid bat algorithm for solving continuous optimization problems. Appl Soft Comput 73:67–82
46. Linhares A (1998) Preying on optima: a predatory search strategy for combinatorial problems. In: SMC'98 conference proceedings. 1998 IEEE international conference on systems, man, and cybernetics (Cat. No.98CH36218), vol 2973, 14–14 Oct 1998, pp 2974–2978
47. Chen M-R, Li X, Zhang X, Lu Y-Z (2010) A novel particle swarm optimizer hybridized with extremal optimization. Appl Soft Comput 10(2):367–373

48. Lyu S, Li Z, Huang Y, Wang J, Hu J (2019) Improved self-adaptive bat algorithm with step-control and mutation mechanisms. J Comput Sci 30:65–78
49. Reddy MP, Ganguli R (2018) Enhancement structures for the bat algorithm. In: 2018 IEEE symposium series on computational intelligence (SSCI), 18–21 Nov 2018, pp 601–608
50. Eberhart RC, Kennedy J (1995) A new optimizer using particle swarm theory. In: Proceedings of the sixth international symposium on micro machine and human science. New York, NY, pp 39–43
51. Eberhart RC, Yuhui S (2001) Particle swarm optimization: developments, applications and resources. In: Proceedings of the 2001 congress on evolutionary computation seoul, vol 81, pp 81–86
52. Hasançebi O, Teke T, Pekcan O (2013) A bat-inspired algorithm for structural optimization. Comput Struct 128:77–90
53. Jaddi NS, Abdullah S, Hamdan AR (2015) Optimization of neural network model using modified bat-inspired algorithm. Appl Soft Comput 37:71–86
54. Wang Y, Wang P, Zhang J, Cui Z, Cai X, Zhang W, Chen J (2019) A novel bat algorithm with multiple strategies coupling for numerical optimization. Mathematics 7(2):135
55. Bahmani-Firouzi B, Azizipanah-Abarghooee R (2014) Optimal sizing of battery energy storage for micro-grid operation management using a new improved bat algorithm. Int J Electr Power Energy Syst 56:42–54
56. Xie J, Zhou Y, Chen H (2013) A bat algorithm based on Lévy flights trajectory. Pattern Recog Artif Intell 26(9):829–837
57. Liu C (2013) Bat algorithm with Levy flight characteristics. CAAI Trans Intell Syst 3:240–246
58. Bekdaş G, Nigdeli SM, Yang X-S (2018) A novel bat algorithm based optimum tuning of mass dampers for improving the seismic safety of structures. Eng Struct 159:89–98
59. Price K, Storn R, Lampinen J (2005) Differential evolution a practical approach to global optimization. Springer, Berlin
60. Michalewicz Z (1995) Genetic algorithms, numerical optimization, and constraints. In: Proceedings of the sixth international conference on genetic algorithms. Morgan Kaufmann, San Mateo, CA, pp 151–158

Chapter 9
Natural Heuristic Methods for Underwater Vehicle Path Planning

Yu-Xin Zhao and De-Quan Yang

1 Path Planning of Underwater Vehicle

1.1 Path Planning

The path planning of vehicles is a significant problem in the field of navigation research; the objective is to navigate the vehicle to the target point without encountering any obstacles. According to the degree of mastering environmental information, the problems can be divided into two types: the first one is global planning with known environmental information, and the other one is local planning with unknown environmental information [1]. The global planning method plans a path for the vehicle according to the acquired environmental information; the accuracy of this path planning depends on that of the obtained environmental information. The global method could usually find the optimal solution, but the exact information of the environment should be provided in advance and also the computational cost is high. The local planning method focuses on the local environmental information of current aircraft so that the aircraft have good collision avoidance capabilities. The navigation method used for underwater vehicles is usually the local planning method because its information acquisition only depends on the sensor system where this changes in real time with environment [2]. Compared with the global planning method, the local planning method is real time and more practical; however, local planning only depends on local information and occasionally produces local poles. Thus, the smooth arrival of the vehicle at the destination could not be guaranteed [3].

From the perspective of the representation dimension of environmental space, there are two types of vehicle path planning: one is path planning in three-dimensional

Y.-X. Zhao (✉) · D.-Q. Yang
College of Automation, Harbin Engineering University, Harbin, China
e-mail: zhaoyuxin@hrbeu.edu.cn

X.-S. Yang and Y.-X. Zhao (eds.), *Nature-Inspired Computation in Navigation and Routing Problems*, Springer Tracts in Nature-Inspired Computing,
https://doi.org/10.1007/978-981-15-1842-3_9

space environments and the other one is path planning in two-dimensional space environments. Any object moves in the three-dimensional space environment; but when the solutions of the problems allow, for more simple calculations, the problems in the three-dimensional space environment can be reasonably abstracted into the two-dimensional space environment. However, in some cases, the movement of the object cannot be reasonably abstracted into two-dimensional space environments, such as the movement of underwater vehicles, because the movement process involves the navigation height, not only the simple change of longitude and latitude. In this case, the path planning in the three-dimensional space environments is required. The path planning in three-dimensional space environments has more abundant data storage, longer calculation time, and higher calculation complexity, resulting in that path planning in three-dimensional space environments is a challenging problem in the field of path planning [4].

The complexity of the environment and the diversity of requirements faced by vehicle path planning induce the following characteristics:

1. Complexity: In complex environments, especially in dynamic time-varying environments, the problems of vehicle path planning are very complex and the calculations cost is high.
2. Randomness: There are many random and uncertain factors in the changes of complex environments. The appearance of dynamic obstacles in unknown environments is often uncertain and is not exactly predictable in advance.
3. Multiple constraints: There are geometric and physical constraints on the motion of the vehicle. Geometric constraints refer to the shape constraints of the vehicle, while physical constraints refer to the velocity and acceleration constraints of the vehicle.
4. Multi-objective: There are many objectives for evaluating the performance of the path in the process of navigation, such as the shortest path, the best time and the least energy consumption; but there are often conflicts between them.

1.2 Objective Optimization

Path planning is an important branch of navigation research. The optimal path planning problem is to find an optimal path that can avoid obstacles from the starting position to the target position in its working space according to the optimization criteria (such as minimum work cost, shortest walking path, and shortest walking time). The optimization objectives for underwater vehicle path planning generally include path length, energy consumption, navigation time, path smoothness, and path safety [5].

The shortest path length is one of the main goals of path. The path length can reflect path planning's effect. In single-objective path planning, it is often taken as the optimization goal. Energy consumption is particularly vital in underwater vehicle path planning; especially, when the underwater vehicle is sailing in a large range

marine environment, the problem of energy consumption becomes very prominent. It is necessary to consider its own energy during the mission and the actual energy consumption when the vehicle tracks the path. Sailing time is related to the length of the path. When the vehicle is required to reach the destination in the shortest time, the sailing time should be taken as the optimization target. The planning results aiming at the path length are often a series of non-smooth broken lines. In order to improve the navigation stability of underwater vehicles and reduce unnecessary energy consumption during turning corners, path smoothness is also set as one of the optimization objectives. Generally, the number of inflection points or the included angle of inflection points is taken as the evaluation criteria. In order to ensure the safe operation of underwater vehicles, path safety is set as one of the optimization objectives. Generally, the shortest distance from the path to the obstacle is also taken as the evaluation criteria.

The path planning problem of underwater vehicles includes both single-objective optimization problems and multi-objective optimization problems. Note that there are internal relations among the optimization objectives. For example, both path length and path smoothness impact the energy consumption and navigation time. Ensuring the safety of the planned path is the premise of the planning task [6].

1.3 Main Processes

The main processes of path planning include environmental data obtaining, environmental modeling, path searching, path optimization, and smoothing.

1.3.1 Environmental Data Obtaining

For the problem of path planning of aircraft, three-dimensional space environment data generally include ocean current and wave information, temperature and salinity information, and three-dimensional seabed terrain information. According to the environment data information, the path planning area can be divided into three categories. The first category is the safety area which the aircraft desire to enter, such as seabed sound track. If the aircraft enter this safety area, navigation safety could be achieved. The second category is the potentially dangerous areas that we do not want the aircraft to enter but are not forcibly forbidden to enter, such as areas with the large difference in seawater salinity. If the aircraft enter this area, it will not cause direct damage; but the difference in seawater salinity can cause great changes in seawater buoyancy and thus result in that the aircraft suddenly sink or float. If the aircraft sink to an area where an obstacle is located, there will be a risk of collision, whereas if the aircraft float to the detectable area, it will be exposed. The third category is the dangerous areas that are forbidden to enter, such as the areas where obstacles are located. If the aircraft enter this area, there could be a risk of collision between the aircraft and the obstacle. Therefore, the aim of aircraft path planning is to navigate

aircraft to enter the safety area, avoid entering the potentially dangerous area, and prohibit entering the dangerous area.

1.3.2 Environmental Modeling

Reasonable environmental modeling can better carry out path planning. The corresponding process requires the known seabed environmental information data as inputs. The collected real three-dimensional seabed terrain elevation data is often the scattered data that are randomly distributed; so, these scattered data are usually rasterized to obtain evenly distributed data on grid points. However, even the evenly distributed grid data fail to meet the requirements of the path planning algorithm. So, it is usually necessary to perform interpolation operations on the evenly distributed grid data to obtain grid data with higher density. The ultimate goal of environmental modeling is to establish an appropriate mathematical model to represent the studied environmental objects. By doing so, the actual physical environment (feasible areas and obstacles) can be simulated by the recognizable and expressible numerical environment, so that the path planning process can be carried out through computer simulation.

1.3.3 Path Searching

The path search algorithm of the vehicle is the focus and main difficulty of the whole vehicle path planning. An appropriate algorithm can quickly and accurately plan a reasonable path, thus saving time and fuel and ensuring the safety of navigation. In order to ensure the feasibility and optimality of the path, the searched path needs to be evaluated and the objective function should be designed. Several factors, such as the path smoothness and path length, are involved. Different weights need to be given to different objective functions. After the algorithm completes the path planning, the target function is used to evaluate the searched path. If the searched path fails to meet the evaluation criteria, the path planning algorithm should be adjusted and searched again. If the path meets the evaluation criteria, the path planning algorithm is terminated.

1.3.4 Path Optimization

Path optimization is the smooth optimization of the path at the inflection point, in order to make the path more conducive to the execution algorithm of the vehicle. Path optimization considers the dynamic characteristics of the intelligent vehicle itself, so that the navigation along the planned trajectory can be realized in practical application. After the path optimization process is completed, an effective path must be output in the path planning area for the navigation of the vehicle.

1.4 The Key to the Problems

The main challenge of vehicle path planning is path planning under three-dimensional seabed environments. The key issues involved mainly include two aspects: one is the three-dimensional seabed environment modeling, and the other is the path planning method. The main aspect of environmental modeling is to reasonably model the path planning environment. This process uses the known three-dimensional seabed environment information, which is a prerequisite for path planning. Through environmental modeling, the actual physical environment can be simulated as a recognizable and expressible computer environment; that is, it can be understood as a data structure, which can describe the physical environment. In fact, environmental modeling establishes a mapping between the computer environment and the actual physical environment. With this mapping, the path planning can be carried out by a computer. The real seabed environment is complex and varied, and the geographical elements are also varied. However, when we establish the virtual seabed environment model, the seabed's environmental elements can be selectively extracted according to the requirements.

The definition of vehicle path planning is to give a description of the vehicle and environment. After the modeling of the vehicle's navigation environment is completed, the task of path planning is to search for an optimal path (or suboptimal path) from the starting point to the target point within the path planning area; during this process, a certain optimization algorithm is applied to ensure safe navigation. The path planning method is responsible for searching the feasible space of the path and finding the optimal path in this space; this is the most important and complicated part in the whole path planning process. To ensure the completion of the navigation task, the path of the vehicle and the process of path planning should be optimized to find the global optimal path in the shortest possible time. Consequently, the vehicle path planning can generate a feasible path with both a high performance and a high quality.

2 Characteristics of Underwater Path Planning

In the marine environment, the complex movement of seawater could include currents, waves, tides, internal waves, storm surges, laminar flow, and turbulent flow of seawater. Subjected to their interference, the vehicle could deviate from the original path and may collide with obstacles [7]. Therefore, the interference of marine environmental factors must be taken into account in the path planning of the vehicle.

2.1 Safe Navigation Factors

Safety is the most basic issue to be considered in the path planning of underwater vehicles. For global path planning problems, safety refers to that planned routes enable the vehicle to avoid known obstacles when obstacles refer to seabed topographical environments; for local path planning problems, safety refers to how to adjust the global planning route locally and avoid effectively sudden static state and dynamic obstacles [8]. The information of sudden static and dynamic obstacles can generally be detected by sensors such as sonar.

2.2 Hidden Navigation Factors

Hidden navigation is an important factor that needs to be considered in the route planning of underwater vehicles. The concealed navigation of underwater vehicles can be realized by improving the concealability of the underwater vehicle and using external environmental information. The external environmental information has certain significances for the path planning of underwater vehicles, mainly including transparency information, seabed terrain information, and marine environmental information. Transparency refers to the visibility of seawater in different regions, which varies with different regions and seasons. Currently, there are transparency data of different sea areas and seasons from satellite remote sensing. Submarine topography can be used to achieve concealed navigation; this chapter will not discuss how to use submarine topography to avoid sonar detection.

2.3 Marine Environmental Factors

The navigation time of the underwater vehicle from the starting point to the target point is mainly determined by the length of the navigation path, the tortuosity of the navigation path, and the current conditions of the navigation area. Underwater vehicles usually work in complex and variable marine environments, where there are eddy currents that could be manifested by a wide range of seawater flowing continuously in time and space. Ocean current, as an energy flow, has a great impact on the operation of the vehicle. Especially, when the vehicle encounters a strong sea current field in the underwater environment, ocean current even jeopardizes the vehicle's smooth execution. The navigation of the vehicle must consider the influence of the current; so judging and predicting the current is the basic requirement for finding the optimal path in the path planning of the vehicle. When the vehicle is sailing in a wide range of marine environments, it is necessary to consider the energy consumption in the mission. We can try to make the vehicle flow along the current and use the energy of the current field to reduce the energy consumption of the

countercurrent navigation. Therefore, it is necessary to plan a path with as little energy consumption as possible and a shortest sailing distance [9].

3 Intelligent Path Planning Algorithm

Due to the complexity, randomness, multi-constraint, multi-objective, and other characteristics of the vehicle path planning problem, the traditional vehicle path planning method often fails to show a good optimization effect. In recent years, with the continuous development of natural heuristic intelligent methods such as neural networks and genetic algorithms, many researchers have used intelligent methods to solve the problem of vehicle path planning and have achieved good results [10]. Among them, the common intelligent path planning methods are as follows: artificial neural network methods, fuzzy logic methods, genetic algorithms, and the ant colony algorithm.

3.1 Neural Network Method

Artificial neural networks are a new interdisciplinary subject inspired by biology and have gradually become an important component of artificial intelligence. Artificial neural networks have good adaptive, self-organizing, and self-learning capabilities; these capabilities enable it to solve the problem of vehicle path planning. The neural network method uses a parallel connection network structure to plan a series of path points, so that the length of the whole path is as short as possible and at the same time it is as far away from obstacles as possible. Collision-free paths are represented by a series of intermediate points and adjacent points are connected by line segments. The collision penalty function is used to quantify the collision property between the path and the obstacle, which is defined as the sum of the collision penalty functions of each path point; the collision penalty function of one point is represented by its neural network for each obstacle. When the artificial neural network is used to solve the problem of vehicle path planning, the capability to avoid obstacles and path planning capability of the vehicle enhances. Currently, the three-layer perceptron model and BP neural network algorithm are widely used.

When using the artificial neural network method for path planning, it has some advantages: the principle of this method is relatively simple, the calculation ratio is relatively small, and the convergence is relatively fast. However, there are still some shortcomings: the neural network needs to adjust network weights according to the feedback information; and the neural network has not been trained at the initial stage of the algorithm, which makes the optimization effect not ideal at the initial stage of the algorithm; and it needs to run for a certain period of time until the system reaches stability before a better optimization effect can be obtained.

3.2 Fuzzy Logic Method

Fuzzy logic was developed from the concept of a fuzzy set proposed by American mathematician Zadeh [11]. According to the fuzzy environment information and the query information table, specific information is obtained, and then the task of path planning can be completed. In the process of vehicle path planning, the description of the environment is often impacted by uncertain factors, which cannot be defined as certain environmental features. Fuzzy logic is not sensitive to the uncertainty of the environment around the vehicle because it requires less precision of the sensor information, thus making the behavior of the vehicle show good consistency, stability, and continuity. The fuzzy logic method is a frequently used method in local path planning technology and it has good applicability for exploring path planning problems without knowing all the environmental information.

The advantage of the robot path planning method based on fuzzy logic is that it can plan a suitable path relatively quickly and accurately in the environment where the robot has partial information or in the dynamic environment. It is also a relatively good method for robots with relatively high navigation speed requirements and relatively strong adaptability to the environment. However, there are also some disadvantages of this method: when the environmental information is complex, i.e., in an environment with many obstacles, the amount of calculations using this method would be relatively large.

3.3 Genetic Algorithm

The genetic algorithm (GA) is a simulated evolutionary optimization algorithm proposed by Prof. John Holland of the University of Michigan in 1975 [12]. With the continuous development of the algorithm, many researchers have successfully applied it to solve the path planning problem [13]. The global path planning method of the vehicle based on the genetic algorithm firstly discretizes the working space of the vehicle with grids. Secondly, it uses a series of orderly arrangement of grid serial numbers to represent the motion path (i.e., an individual) of a vehicle. It uses a group composed of multiple motion paths as the basis for the optimal search. Finally, it uses the genetic operator to carry out the genetic operation on the group to obtain the optimal motion path of the vehicle under the evaluation criterion of "shortest path length."

When the genetic algorithm is used to solve the problem of vehicle path planning, it has the advantages of strong global searching ability and good robustness. However, there are also some disadvantages: when the length of the code needed for optimization is relatively longer, the computation is relatively larger, the running speed is relatively slower, the real-time performance is lacking, and the phenomenon of "premature" convergence is easy to occur, that is, the optimized path is not the global optimal path, but a suboptimal path.

3.4 Ant Colony Algorithm

In 1992, Marco Dorigo proposed a simulated evolutionary algorithm—ant colony algorithm in his Ph.D. thesis [14]. The algorithm originated from research on the foraging behavior of ants. Researchers found that ants in nature have obvious self-organization and self-adaptive behavior in the process of searching for food. The positive feedback and coordination of the ant colony algorithm make it applicable to distributed systems, and the implicit parallelism makes it have strong development potential. It has good adaptability in solving the problem of optimal combination. Therefore, it is applied to the global path planning of vehicles to explore a new path optimization algorithm.

The ant colony algorithm has good effects on small-scale optimization problems; but for complex optimization problems, its optimization performance drops sharply. The main reason for this problem is that at the initial stage of ant colony algorithm optimization, the pheromone level on each path is basically the same, and the ant search shows great blindness. Only after a relatively long time, the pheromone level on each path may show a big difference; the difference in pheromone level could play a guiding role in the selection of ant paths. In addition, because the ant colony algorithm is a positive feedback algorithm, it is also easy to fall into local optimal solutions.

4 Firefly Algorithm

4.1 Bionics Principle

There are many kinds of fireflies in nature, and different kinds of fireflies have different purposes of lighting. Some fireflies attract the opposite sex to mate and reproduce new life; some fireflies attract prey; and some fireflies may play a warning role to tell foreign enemies that they are inviolable. All kinds of uses are to be further explored by scientists.

Fireflies in nature mainly rely on their own light to transmit information or attract food with their companions, but the light will be absorbed in the process of transmission. Therefore, the further the fireflies fly, the worse the accuracy of information transmission is. Based on this information, British scholar yang Xin-she proposed a new natural heuristic intelligent optimization method—firefly algorithm (FA), which is a random optimization algorithm similar to the particle swarm optimization algorithm, while this algorithm awaits more development [15–18].

The firefly algorithm simulates the solution in the solution space as individual fireflies, and each individual firefly contains its own fluorescein value and perception radius. The fluorescein value is used to measure the position of the individual, i.e., the solution. The perceived radius is used to determine the search range of the individual. The individual can only find excellent individuals within a certain search range and

move toward them. When a firefly finds an individual with a large fluorescein value, within its own search range, it moves toward the individual. The firefly algorithm can exchange information between individual fireflies by comparing the fluorescein value of each firefly, thus achieving optimization in the solution space.

The firefly algorithm is inspired by the luminous intensity of fireflies. The fluorescence they emit is mainly used to attract the opposite sex. In order to make the algorithm simpler, more effective and more accurate, we remove some unimportant factors and assume that all fireflies with weak luminous intensity move to fireflies with strong luminous intensity in a certain search area, realizing the iteration of the optimal position. In order to construct the firefly algorithm, the following idealization criteria are used to idealize some characteristics of firefly luminescence:

1. Fireflies are genderless, that is, each firefly will attract all other fireflies regardless of gender.
2. The attraction is proportional to their brightness. For two fireflies, the dimmer fireflies will be attracted by the brighter fireflies and move toward them continuously, while the brighter fireflies will move randomly. This attraction is proportional to brightness, and the attraction gradually decreases as the distance between fireflies increases.
3. The brightness of fireflies is determined by the optimizing objective function value.

Due to the increase of distance and the absorption of light by air, the brightness of a firefly i will gradually decrease when the distance r increases. In order to show the variation of this brightness with distance, the definitions of absolute brightness and relative brightness of fireflies are given.

Absolute brightness: For a firefly i, the initial light intensity (light intensity at $r = 0$) is the absolute brightness of firefly i, denoted by I_i.
Relative brightness: The light intensity of firefly i at the position where firefly j is located in the relative brightness of firefly i to firefly j, denoted by I_{ij}.

Assume that the solution space of the objective function to be optimized is d-dimensional. In this solution space, the firefly algorithm randomly initializes a group of fireflies $\overline{x_1}, \overline{x_2}, \ldots, \overline{x_i}, \ldots, \overline{x_n}$, n is the number of fireflies, and $\overline{x_i} = (x_{i1}, x_{i2}, \ldots, x_{id})$ is a d-dimensional vector, which indicates the position of firefly i in the solution space and can represent a possible solution of the problem. Defining the absolute brightness of fireflies is directly determined by the objective function, then the magnitude of the absolute brightness directly indicates the advantages and disadvantages of the possible solution represented by fireflies; that is, the possible solution, which represented by fireflies with higher absolute brightness, is better.

All fireflies follow the following attraction rules: fireflies with low absolute brightness are attracted by those with high absolute brightness and they (with low absolute brightness) move toward them (with high absolute brightness) continuously. Based on this rule, each firefly would move to the firefly with higher absolute brightness in the solution space, thus updating its position and obtaining a new solution. The firefly with the highest absolute brightness in the whole solution space cannot be

attracted by any firefly and it moves randomly. Through this attraction mechanism based on (absolute) brightness, the entire population can continuously explore in the solution space and all fireflies move to a better solution area.

After all the fireflies move to the new position, the absolute brightness of the fireflies is updated. Note the fireflies are attracted by the fireflies with the higher absolute brightness than them. After a period of iteration, fireflies would gather around the fireflies with higher brightness to obtain the optimal solution of the objective function. This interaction between fireflies is almost independent, so that the solution space can be searched efficiently, and the local optimal solution and the global optimal solution can be obtained at the same time.

4.2 Algorithm Description

The firefly algorithm first randomly distributes N firefly individuals in the problem-solving space (each individual firefly represents a solution in the solution space of the problem); each individual firefly carries a quantitative fluorescein value $l_i (i = 1, 2, \ldots, N)$, where the individual's fluorescein value is associated with the individual's location, that is, the fluorescein value is associated with the solution function value $J(x_i)$. At the same time, each individual firefly has its own perception range, the size of which is determined by the perception radius r_{di}. Firefly i searches for all individuals with larger fluorescein values to form a neighborhood set $N_i(t)$ within its perception range ($0 < r_{di} < r_s$, where r_s is the maximum perception radius of the individual), then selects an individual l_j with larger fluorescein value from the neighborhood set $N_i(t)$ according to a probability function; and firefly i moves toward the individual j, and finally updates the perception radius r_{di}. This process is repeated until a certain number of iterations are reached. When all firefly individuals gather at a certain position in the problem-solving space, that is, the optimal solution position of the problem, the problem is solved.

The firefly algorithm can be divided into fluorescein value updating stage, searching for the brightest individual stage, the firefly position updating stage, and the neighborhood radius updating stage.

4.2.1 Update Fluorescein Value

As the fluorescein value of the fireflies is constantly updated, the fluorescein value simulates the fitness of a solution in the solution space. The higher the fluorescein value is, the stronger the attraction of the individual and the greater the probability of other individuals moving toward it are. The fluorescein value is related to the value of the solution function. In the formula, $l_i\ (t + 1)$ represents the fluorescein value of individual i in the $t + 1$ iteration; $l_i\ (t)$ represents the fluorescein value of individual i in the t iteration; $J(x_i\ (t + 1))$ represents the fitness function value of individual i in the $t + 1$ iteration; ρ represents the volatilization coefficient of the fluorescein value;

and γ represents the enhancement coefficient of the fluorescein value.

$$l_i(t+1) = (1-\rho) * l_i(t) + \gamma * J(x_i(t+1)) \tag{1}$$

1. Find the brightest individual

Each individual firefly can be attracted by the individual with high fluorescein value. However, the attraction action is affected by the perception range and the individual can only find and move to the individual with high fluorescein value within its perception range.

The size of the perception range is determined by the perception radius r_{di}. When the distance between individual i and individual j, $\|x_j - x_i\|$ is smaller than the radius r_{di}, individual j is considered to be within the perception range of individual i. If the fluorescence brightness of individual j simultaneously satisfies that l_j is larger than the fluorescence brightness l_i of the individual i, a better individual j is considered to be found within the perception range of individual i.

There may be many individuals whose fluorescein values are greater than their own in the perception range of individual i, so the firefly algorithm establishes a neighborhood set for each individual $N_i(t) = \{j{:}d_{ij}(t) < r_{dj}(t); l_i(t) < l_j(t)\}$; and the set stores all individuals whose fluorescein values are greater than their own in the neighborhood set, where d_{ij} represents the distance between individual i and individual j.

After the neighborhood set $N_i(t)$ is established, the probability that all individuals $j(j \in (N_i(t)))$ in the neighborhood set of individuals are selected as target individuals is calculated according to the Formula (2); and then, the better individuals are selected as targets to move to according to the calculated probability.

$$p_{ij} = \frac{l_j(t) - l_i(t)}{\sum\limits_{k \in N_i(t)} l_k(t) - l_i(t)} \tag{2}$$

2. Update firefly location

After calculating the probability distribution of the neighborhood set for individual i, individual i finds the individual j whose fluorescein value is higher than that of himself in the perception range according to the probability distribution and then moves position according to the formula to approach individual j. i denotes an individual whose position is to be moved; j denotes an individual with a high fluorescein value selected according to the probability distribution, i.e., a target individual to which individual i is gradually approaching in this iteration; s denotes a moving step length; $x_i(t)$ denotes a position before the individual i moves; and $x_i(t+1)$ denotes a position after the individual moves.

$$x_i(t+1) = x_i(t) + s * \left(\frac{x_j(t) - x_i(t)}{\left\| x_j(t) - x_i(t) \right\|} \right) \tag{3}$$

3. Update neighborhood radius

After the position of individual i changes, its perception radius needs to be updated, i.e., the perception range of individual i needs to be enlarged or reduced in order to find more local optimal solutions and thus find global optimal solutions.

The update procedure of the perception radius of individual i is related to the neighborhood set $N_i(t)$. If there are more individuals with high fluorescein value and $|N_i(t)|$ is larger in the perception range of individual i, then the perception radius should be appropriately reduced so that more local optimal solutions can be found. If there are only a few individuals with high fluorescein value and $|N_i(t)|$ is small, within the perception range of individual i, then the perception radius should be appropriately enlarged to facilitate individual i to have more choices and move toward a better position.

The neighborhood radius of individual i is updated according to the according to Formula (4), where r_s represents the maximum perception radius of all individuals; and β represents the change coefficient of perception radius. n_t represents the number of excellent individuals defined in the perceived range, when the number of better individuals in the perception range is greater than n_t, the neighborhood radius of individuals decreases, otherwise it increases. $|N_i(t)|$ represents the number of better individuals found within the individual perception range, i.e., the size of neighborhood set.

$$r_d^j(t+1) = \min\left\{r_s, \max\left\{0, r_d^j(t) + \beta * (n_t - |N_i(t)|)\right\}\right\} \tag{4}$$

4.3 Algorithm Flow

The firefly algorithm needs to perform three stages: algorithm initialization, firefly location update, and firefly brightness update. The basic flow of firefly algorithm is shown in Fig. 1.

In the algorithm initialization stage, the algorithm parameters are set and the location of the firefly is initialized; and the position vector of the firefly is brought into the target function to initialize the brightness of the firefly. In the firefly location updating stage, all firefly positions are updated according to the brightness of the firefly and the attraction rules between the fireflies. In the firefly brightness updating stage, the new position vector of the firefly is brought into the target function to complete the brightness updating of all the fireflies. The commonly used termination conditions of the algorithm are: the algorithm reaches a certain number of iterations and the algorithm obtains an optimized target value that meets the requirements [19].

Combined with the above description of firefly algorithm and when we take the termination condition of the algorithm as “reaching a specific iteration number” as an example, the pseudocode of firefly algorithm can be summarized as follows:

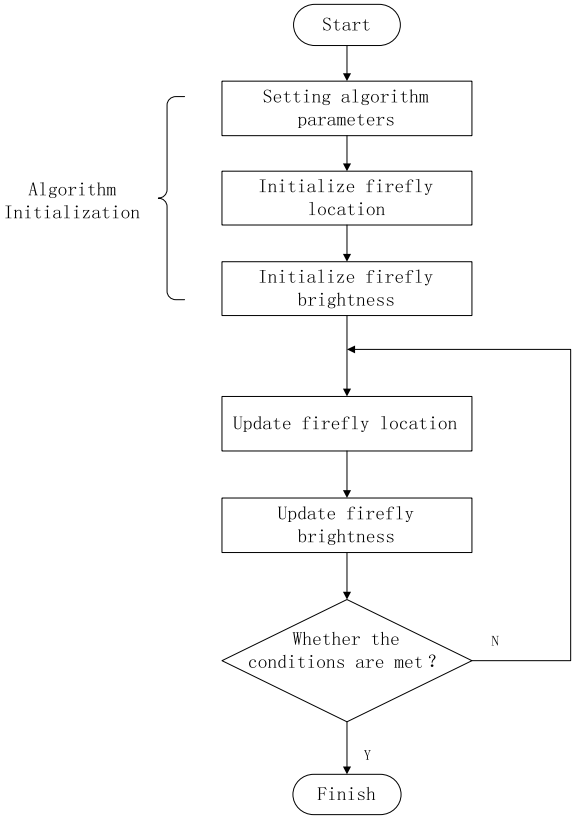


Fig. 1 Firefly algorithm flow chart

Define objective function $f(\bar{x}), \bar{x} = (x_1, x_2, \cdots, x_d)^{\tau}$
Set Max Generation as the number of algorithm iterations
Set algorithm parameters α, β_0, γ
Initialize n fireflies $\overline{x_i} = (i = 1, 2, \ldots, n)$
The absolute brightness of each firefly $f(\overline{x_i})$ is determined by the objective function value $I(\overline{x_i})$
While ($t <$ Max Generation)

for $i = 1$: n All n fireflies
for $j = 1$: n All n fireflies
if $(I(\overline{x_i}) > I(\overline{x_j})$
Calculate the distance between firefly i and firefly j, r_{ij}
Calculate the distance between firefly i and firefly j, $\beta_{ij}(r_{ij})$
Move firefly j to firefly i.
end if
Evaluate the new solution and update the brightness of fireflies.

```
    end for j
    end for i
  end while
  Arrange all fireflies to find the current optimal solution
Analyze the results
```

4.4 Performance Analysis

In the field of intelligent optimization, the genetic algorithm and particle swarm optimization (PSO) algorithm are the two most influential algorithms in this field, and their research is relatively mature [20]. As a novel intelligent optimization algorithm, the firefly algorithm and the two algorithms mentioned above have many similar characteristics [21]. In order to better understand firefly algorithm, and to clarify its connection and difference with genetic algorithm and particle swarm optimization algorithm, the firefly algorithm is compared with these two algorithms.

In order to compare the optimization performance of these three algorithms, this section uses four test functions to compare the firefly algorithm, particle swarm optimization algorithm, and genetic algorithm. These test functions are widely used in the research of evolutionary algorithms. The purpose of the experiment is to find the global minimum of four test functions. The mathematical expression and optimization search range of the test function are shown in Table 1, and their global minimum values are all 0. Among them, the sphere function and the Rosenbrock function are single-mode functions, while the Rastrigin function and the Griewank function are multi-mode functions with multiple local optimal values. The three-dimensional diagrams of the four test functions are shown in Figs. 2, 3, 4, and 5, respectively.

In order to fairly compare the optimization performance of the algorithms, the population size of the three algorithms is set to $N = 40$, and the dimension of each test function is set to 30 dimensions. The firefly algorithm, genetic algorithm, and particle swarm optimization algorithm are used to optimize the above four test

Table 1 Test functions

Function name	Mathematical expression	Search scope
Sphere	$f_1(\vec{x}) = \sum_{i=1}^{n} x_i^2$	$[-100, 100]^n$
Rosenbrock	$f_2(\vec{x}) = \sum_{i=1}^{n-1} \left[100\left(x_i^2 - x_{i+1}\right)^2 + (x_i - 1)^2\right]$	$[-100, 100]^n$
Rastrigin	$f_3(\vec{x}) = \sum_{i=1}^{n} \left[x_i^2 - 10\cos(2\pi x_i) + 10\right]$	$[-100, 100]^n$
Griewank	$f_4(\vec{x}) = \sum_{i=1}^{n-1} \left[100\left(x_i^2 - x_{i+1}\right)^2 + (x_i - 1)^2\right]$	$[-600, 600]^n$

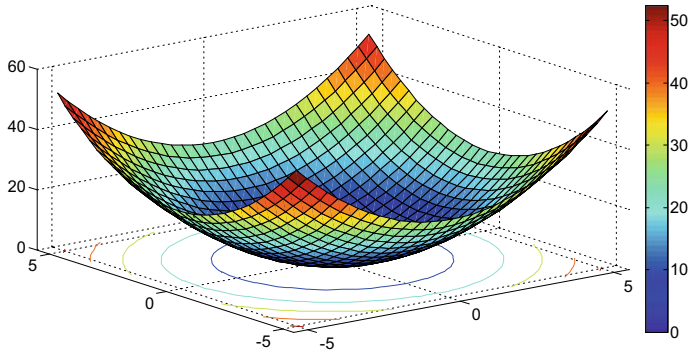


Fig. 2 Three-dimensional diagram of the sphere function

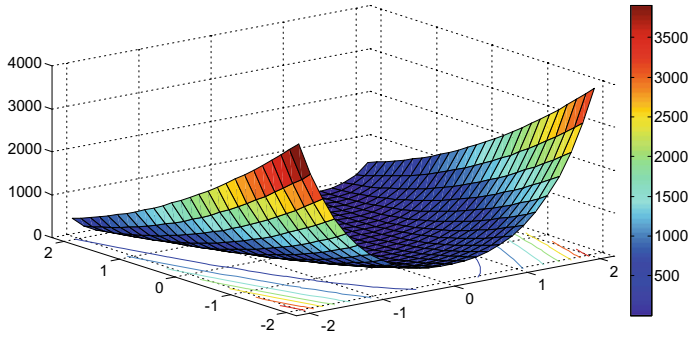


Fig. 3 Three-dimensional diagram of the Rosenbrock function

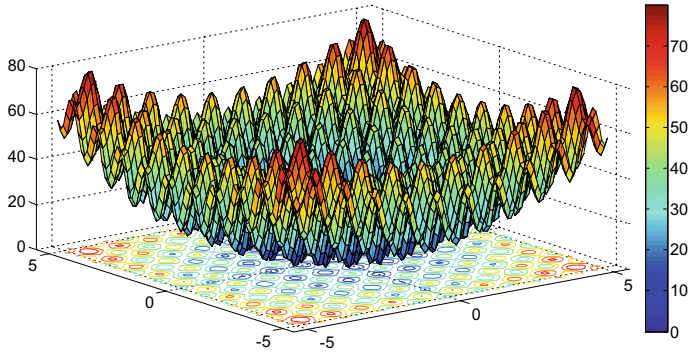


Fig. 4 Three-dimensional diagram of the Rastregin function

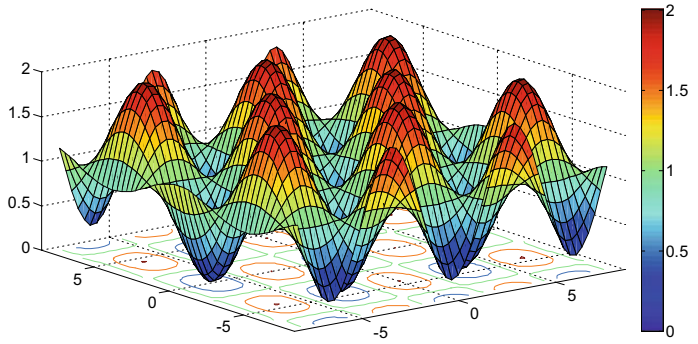


Fig. 5 Three-dimensional diagram of the Griewank function

functions. In order to observe the iterative process of the algorithm, the optimization iterative curves of the three algorithms for the 30-dimensional test function are shown in Fig. 6, in which the abscissa is the iteration number of the algorithm, and the ordinate is the average optimal value obtained by running the algorithm 30 times.

As can be seen from Fig. 6, compared with the genetic algorithm and particle swarm optimization algorithm, the firefly algorithm has a slower convergence speed and usually requires more iterations to find a better optimal value. Especially, for the Griewank function, after 20,000 iterations the optimal value found by firefly algorithm has not been significantly improved. The above experimental results show that the convergence speed of the firefly algorithm is too slow to effectively meet the needs of the problem.

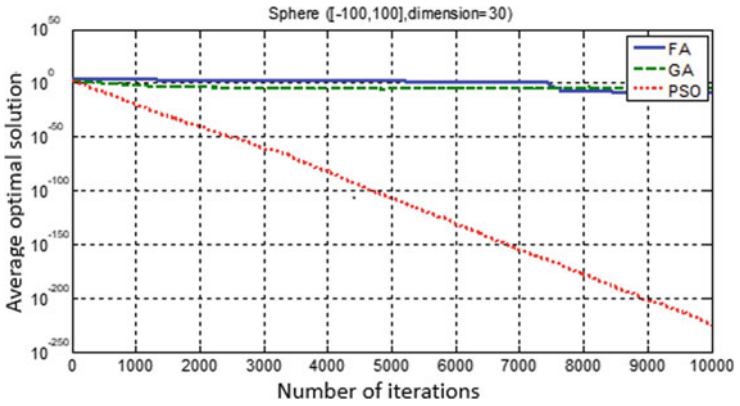
In firefly algorithm, optimization is realized by the interaction between fireflies. The firefly j with lower brightness is attracted by firefly i with higher brightness and move position as follows:

$$\vec{x}_j(t+1) = \vec{x}_j(t) + \beta_{ij}(r_{ij})\left(\vec{x}_i(t) - \vec{x}_j(t)\right) + \alpha(\text{rand} - 1/2) \tag{5}$$

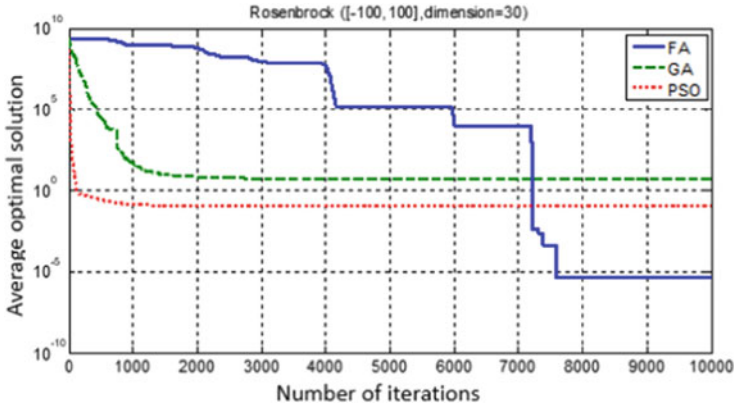
$$\beta_{ij}(r_{ij}) = \beta_0 e^{-\gamma r_{ij}^2} \tag{6}$$

$$r_{ij} = \left\|\vec{x}_i - \vec{x}_j\right\| = \sqrt{\sum_{k=1}^{d}(x_{i,k} - x_{j,k})^2} \tag{7}$$

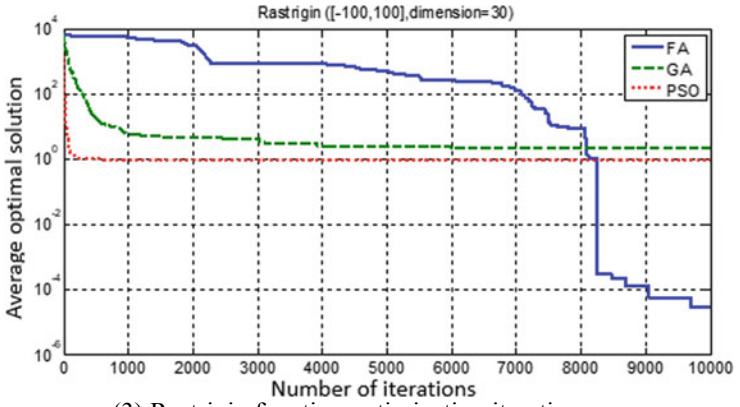
Because the firefly populations are randomly and evenly distributed in the whole search space when the algorithm is initialized, the cartesian distance r_{ij} between fireflies is large, and β_{ij} (r_{ij}) is small and the second term of the position update formula $\beta_{ij}\left(r_{ij}\right)\left(\vec{x}_i(t) - \vec{x}_j(t)\right)$ is very small; that is, attraction has rather little effect on firefly movement. In addition, the third term of the location update formula $a(\text{rand} -1/2) \in a \times [-1/2,1/2]$, when $a = 1$, $a(\text{rand} -1/2) \in [-1/2,1/2]$, this random step size is also relatively small; that is, fireflies have limited free exploration area. Therefore,



(1) Sphere function optimization iteration curves

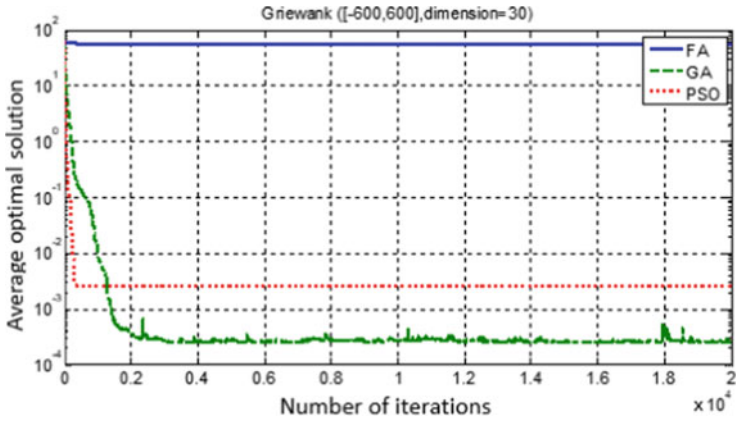


(2) Rosenbrock function optimization iteration curves



(3) Rastrigin function optimization iteration curves

Fig. 6 Optimization iteration curves of FA, GA, and PSO for the four test functions



(4) Griewank function optimization iteration curves

Fig. 6 (continued)

at the beginning of the algorithm, the firefly algorithm has insufficient exploration ability and fails to find the region where the optimal value is located quickly in the whole search space, resulting in a slow convergence speed.

4.5 Algorithm Improvement

When the distance between fireflies is large, the attraction has little effect on the position update of the fireflies, which is consistent with the basic idea of the algorithm. At this time, if fireflies themselves can independently explore a reasonable range, the exploration ability of the algorithm can be enhanced, thus accelerating the convergence speed of the algorithm. It should be noted that when the distance between fireflies is small, the ability of fireflies to explore independently should be limited so that attraction plays a dominant role in the update process of the fireflies' positions. Therefore, we use the new firefly location update formula:

$$\vec{x}_j(t+1) = \vec{x}_j(t) + \beta_{ij}(r_{ij})\left(\vec{x}_i(t) - \vec{x}_j(t)\right) + 2r_{ij}(\text{rand} - 1/2) \tag{8}$$

Apparently, $2r_{ij}(\text{rand} - 1/2) \in [-r_{ij}, r_{ij}]$ is the random step size of fireflies which varies with r_{ij}. After the above position updating formula is adopted, when the distance between fireflies is large and the attraction has a little guiding effect on the position updating of fireflies, the fireflies can move autonomously within the range $[-r_{ij}, r_{ij}]$, which is convenient for the algorithm to explore a larger search space. When the distance between fireflies is small, the range in which the fireflies can move autonomously decreases with decrease in distance, and the influence of attraction on the position update greatly increases, thus guiding the fireflies to move to the fireflies

with higher brightness. The algorithm for updating the location of fireflies is called the modified firefly algorithm (MFA).

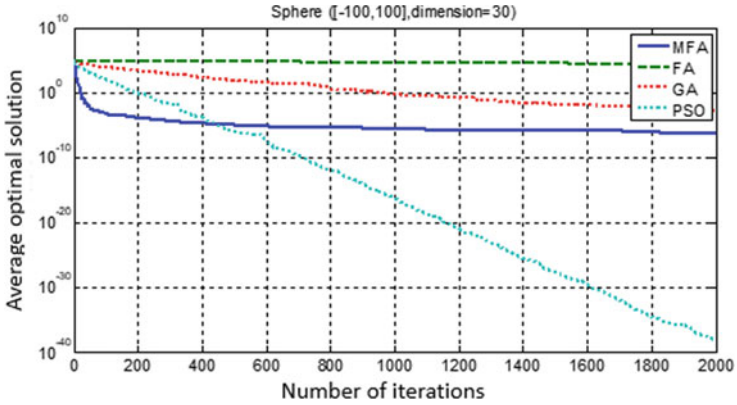
Through simulation experiments, the performance of the MFA is tested and compared with the optimization results of the basic firefly algorithm, genetic algorithm, and particle swarm optimization algorithm. In order to compare the accuracy of the optimal value obtained by the algorithm, a specific number of iterations are taken as the termination condition of the algorithm and a specific number of iterations are checked. When the algorithm reaches a certain number of iterations, the algorithm terminates. We run the experiment 30 times, compare the statistical values (including the optimal value, pessimistic value, average value, and standard deviation) of the optimization results of the algorithm, and draw the optimization iteration curve of the algorithm.

The statistical values of the optimization results of the algorithm are shown in Table 2. The bold typeface is used to identify the optimal values of each row of data. The optimization iteration curve of the algorithm is shown in Fig. 7.

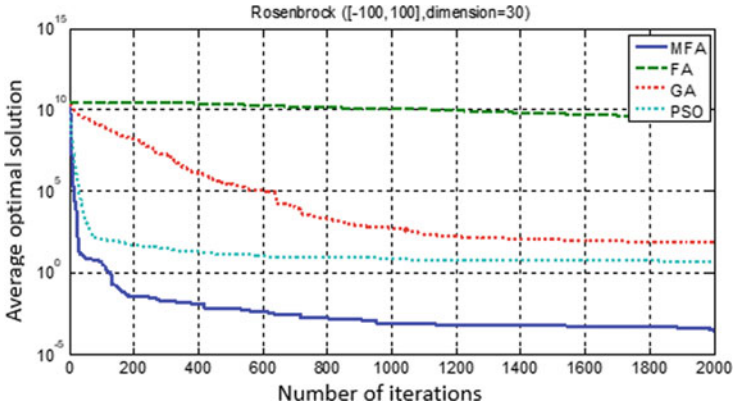
For the sphere function, PSO has the highest optimization accuracy and the lowest standard deviation of the optimization results, while MFA has higher optimization accuracy than FA and GA. For the Rosenbrock function, Rastrigin function and

Table 2 Algorithm optimization results

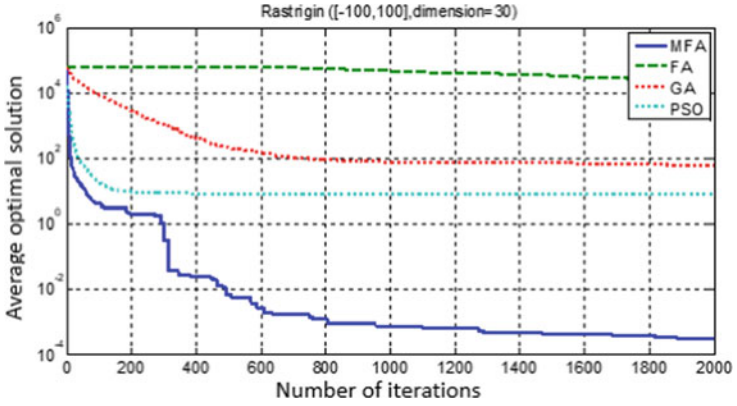
Function and its search range	Statistical value	MFA	FA	GA	PSO
Sphere $[-100, 100]^{30}$	Optimal value	1.94E−12	5.93E+01	4.26E−04	**3.79E−46**
	Pessimistic value	5.42E−06	4.62E+04	1.33E−02	**1.34E−37**
	Average	6.22E−07	2.22E+04	2.30E−03	**6.11E−39**
	Standard deviation	1.35E−06	1.30E+04	2.90E−03	**2.54E−38**
Rosenbrock $[-100, 100]^{30}$	Optimal value	**4.84E−08**	5.16E+04	1.16E+00	5.82E−04
	Pessimistic value	**3.30E−03**	7.75E+09	1.75E+02	6.77E+01
	Average	**2.78E−04**	2.61E+09	7.31E+01	5.18E+00
	Standard deviation	**6.53E−04**	2.18E+09	5.01E+01	1.24E+01
Rastrigin $[-100, 100]^{30}$	Optimal value	**3.49E−11**	6.08E+03	1.90E+01	9.95E−01
	Pessimistic value	**2.50E−03**	3.94E+04	1.41E+02	1.59E+01
	Average	**2.83E−04**	2.54E+04	6.06E + 01	7.96E+00
	Standard deviation	**5.86E−04**	9.22E+03	1.41E+02	3.68E+00
Griewank $[-600, 600]^{30}$	Optimal value	**9.35E−13**	4.26E+02	5.41E−04	0.00E+00
	Pessimistic value	**1.99E−06**	6.72E + 02	3.42E−01	9.36E−02
	Average	**2.21E−07**	5.81E+02	9.23E−02	2.39E−02
	Standard deviation	**3.91E−07**	6.60E+01	7.39E−02	2.59E−02



(1) Sphere function optimization iteration curves

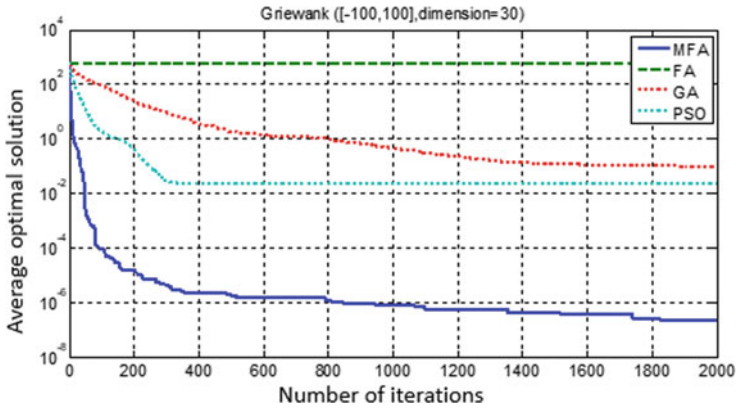


(2) Rosenbrock function optimization iteration curves



(3) Rastrigin function optimization iteration curves

Fig. 7 Optimization iteration curve of algorithm under a large search range



(4) Griewank function optimization iteration curves

Fig. 7 (continued)

Griewank function, the MFA is apparently superior to another three algorithms in terms of precision and standard deviation of the optimization results. The MFA can get a better optimal value at the beginning of the iterations. After 2000 iterations, the accuracy of the obtained optimal value is higher than that of other three algorithms. For the sphere function, although the MFA fails to perform as well as PSO in optimization accuracy, it also obtains a satisfied optimal value. It can be seen that the MFA performs well after a certain number of iterations and can obtain better optimization results than the FA.

5 Route Planning Based on Firefly Algorithm

5.1 Environmental Modeling

Environmental modeling is an important part and also the basic link of vehicle path planning. The essence of environmental modeling is based on known environmental information. By extracting and analyzing relevant environmental features, we can then transform this environmental feature into a feature space that the vehicle can recognize and understand. When we combine the specific path planning algorithm to select the appropriate environment modeling method, a reasonable modeling method can reduce the search volume and reduce the complexity of time-space in the path planning process. In path planning technology, grid method modeling has always been favored by scholars and most of them are used in vehicle path planning. Three-dimensional space path planning also uses the grid method. The main idea is to first divide the solid into a plane and then into a plane grid.

The working environment of the vehicle is defined in the three-dimensional rectangular coordinate system *O-XYZ*, with the origin point *O* as the starting point of the vehicle, the point *S* as the end point, and *OS* is on the *Y*-axis. In the coordinate system *O-XYZ*, the cubic *ABCD–EFGH* is made, in which the *ABCD* plane is a square plane on the *XOZ* plane; *AB* is parallel to the *X*-axis; *BC* is parallel to the *Z*-axis; the origin *O* is the midpoint of the square plane *ABCD*, and $AE = h$. Firstly, the planning space *ABCD–EFGH* is divided into $n + 1$ equal parts along *OS* and the parallel plane to the *ABCD* plane is made through each bisection point, so that we can obtain n planes $\Pi_j (j = 0, 1, 2, \ldots, n)$. Then, the arbitrary plane Π_j is divided into m equal parts along the edge $A'B'$, and m equal parts along the edge $A'D'$; so we divide the plane Π_j into $m \times m$ grids, and the planning space *ABCD–EFGH* is discretized into some points, and we call the set of points *S**.

Let the coordinate of the sequence number of any discrete point in the planning space be P(i, j, k), where $i = \{0, 1, 2, \ldots, m\}$, $j = \{0, 1, 2, \ldots, m\}$, $k = \{0, 1, 2, \ldots, m\}$, the point $P(x, y, z)$ corresponding to the *O-XYZ* coordinate system can be obtained by the following formula:

$$x = -l + \frac{2l \times i}{m} \tag{9}$$

$$y = \frac{j \times h}{n} \tag{10}$$

$$z = -l + \frac{2l \times k}{m} \tag{11}$$

5.2 Route Expression

Any route is connected by a series of waypoints. When we assume that n waypoints (excluding start and end points) form a route, the route can be expressed as P = (O, p_1, p_2, …, p_n, S), where (p_1, p_2, …, p_n) is the sequence of waypoints; the goal of route planning is to determine these n waypoints. In order to determine these waypoints and reduce the number of calculations needed, three-dimensional route planning based on the firefly algorithm generally adopts the following design ideas: determine the planning space according to the navigation tasks, and set the route planning process as the process of finding waypoints in each plane equally divided along a certain coordinate axis direction, and then connect all waypoints into lines in sequence.

Let the coordinates of the starting point and the target point of the navigation task be $P_o(x_o, y_o, z_o)$ and $P_s(x_s, y_s, z_s)$, respectively. If $|x_o - x_s| \geq |y_o - y_s|$, then the latitude direction is selected as the main direction of the route search, and the waypoints are dispersed in the equal latitude planes divided along the *OX* direction; otherwise, the longitude direction is selected as the main direction of the route search and the

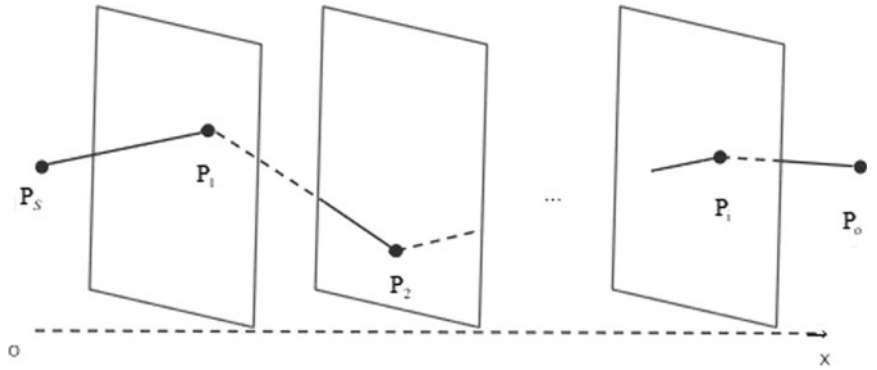


Fig. 8 Schematic diagram of three-dimensional route expression

waypoints are scattered in equal longitude planes divided along *OY* direction. When the latitude direction is used as the main direction of the route search, the number of route points to be searched is n, and thing similar here would be better the route start point P_o and end point P_s are located, and we record them as Ω_0 and Ω_{n+1}, respectively. Then, the planning space between Ω_0 and Ω_{n+1} is equally divided into $n + 1$ segments along the *OX* direction to obtain n planes parallel to Ω_0 and Ω_{n+1}, which are denoted as Ω_i, $i = \{1, 2, \ldots, n\}$. When we use the firefly algorithm for route planning, first, find the first waypoint P_1 of the target route on the first plane Ω_1, and then find the second waypoint P_2 of the target route on the second plane Ω_2 by P_1 and so on to find the rest of the waypoints. The starting point P_s, the respective waypoints, and the ending point P_o are sequentially connected to form a target route (Fig. 8).

5.3 Evaluation Function

The length of the route is one of the most commonly used route evaluation indicators. A route with n route points consists of $n - 1$ route segments. The length of the route is the sum of the lengths of each route segment. The formula for calculating the route length is as follows:

$$\begin{cases} E = L_{\mathrm{SP}_1} + \sum_{i=1}^{n-1} \Delta L_1 + L_{P_n O} \\ \Delta L_i = \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2 + (z_i - z_{i-1})^2} \end{cases} \tag{12}$$

where (x_i, y_i, z_i) is the coordinate of the ith waypoint, $(x_{i-1}, y_{i-1}, z_{i-1})$ is the coordinate of the $i - 1$th waypoint, $1 \leq i < n$; L_{SP1} is the distance between the starting point and the first waypoint; ΔL_i, the distance between the ith waypoint and the $i + 1$th waypoint; L_{PnO} is the distance between the last waypoint and the target point.

5.4 Process Design

Step 1 Determine the starting point and target point of the route according to the task information, then determine the navigation area and finally rasterize the navigation environment to construct the planning space.

Step 2 Set the firefly population parameters and initialize the firefly population in the planning space.

First, set the firefly population parameters, including the population size M, the random coefficient starting value α_b, the random coefficient ending value α_e, the maximum attractive force β_0, the absorption coefficient starting value γ_b, the absorption coefficient ending value γ_e, the dimension n, and the maximum iteration number of times N.

The initialization method of firefly population is as follows: the main direction of the route planning is determined according to the positional relationship between the starting point and the target point; then, the planning space is divided along the direction to obtain n planes. The initialization is to find a waypoint on each plane. The motion state of the vehicle includes four states: straight, floating, diving, and steering. Considering the vehicle's own performance and maneuver constraints, floating and diving must meet certain angle constraints and the steering must meet the minimum turning radius constraint. These constraints are reflected in the route planning to limit the selection of waypoints to a certain range. According to the maximum floating and dive angles, the range of motion of the aircraft in the vertical direction can be determined. According to the minimum turning radius, the range of motion of the horizontal steering of the vehicle can be determined. Let the range of the upper and lower movement of the vehicle be H_{Scale} and the range of left and right movements be Y_{Scale}, then the actual selection range of the waypoints in each plane is a rectangular area with side lengths of $2H_{\text{Scale}}$ and $2Y_{\text{Scale}}$. A waypoint is randomly selected as the first waypoint within the selection range of the first waypoint; then, the second waypoint is determined using the above method according to the position of the point. All waypoints are found by analogy, thus the initialization of a firefly is completed. Other fireflies are initialized in the same way.

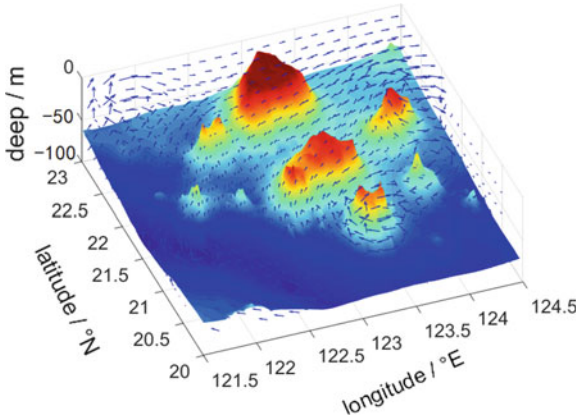
Step 3 Use the improved firefly algorithm to search for the optimal route and obtain the optimal route.

Calculate the brightness value of each firefly according to the evaluation function, and then sort the whole firefly population according to the brightness values of the firefly population; calculate the brightness variance of the firefly population, and calculate the absorption coefficient and random coefficient of the firefly algorithm according to the brightness variance; then update the position of each firefly according to the position movement formula. In the location update process, the actual selection range of the waypoints also needs to be considered . When the firefly's node position

Table 3 Firefly population parameters

Race size	Ethnic dimension	Maximum gravity β_0	Absorption coefficient starting value γ_b	Absorption coefficient termination value γ_e	Random coefficient starting value α_b	Random coefficient termination value α_e
10	30	1	1.0	0.8	0.1	1.0

Fig. 9 Planning space



exceeds the set range, the position update is re-executed until a node that satisfies the condition is found. Repeat the above operation until the end condition is met.

5.5 *Simulation*

The effectiveness and correctness of the route planning based on the improved firefly algorithm are verified by simulation experiments. The firefly population parameter settings are as follows (Table 3). In the simulation test, route planning is carried out in the sea area with a range of 3° × 3° (Fig. 9). In order to facilitate the obstacle avoidance ability of the observation algorithm and highlight the obstacle characteristics of the planning space, the actual length of the abscissa and ordinate is reduced to (1852 × 60)/1000 in the following test.

5.5.1 Algorithm Validity

In order to verify the effectiveness of the firefly route planning algorithm, experiments were carried out for different current conditions with the starting point and the target point at the same depth and also at different depths.

Experiment 1: When the current is small, the depth of the starting point and the target point are both 50 m. The latitude and longitude of the starting point are (122.5° E, 21.17° N), and the latitude and longitude of the target point are (123.63° E, 22.37° N). The three-dimensional and top-view images of the planning results are shown in Figs. 10 and 11, respectively.

Experiment 2: When the current is small, the depth of the starting point is different from the depth of the target point. The latitude and longitude of the starting point are (122.5° E, 21.17° N), the depth is 100 m, the latitude and longitude of the target point are (123.63° E, 22.37° N), and the depth is 150 m. The three-dimensional and top views of the planning results are shown in Figs. 12 and 13, respectively.

Experiment 3: When the current is large, the depth of the starting point and the target point are both 100 m. The latitude and longitude of the starting point are (122.5° E, 21.17° N), and the latitude and longitude of the target point are (123.63°

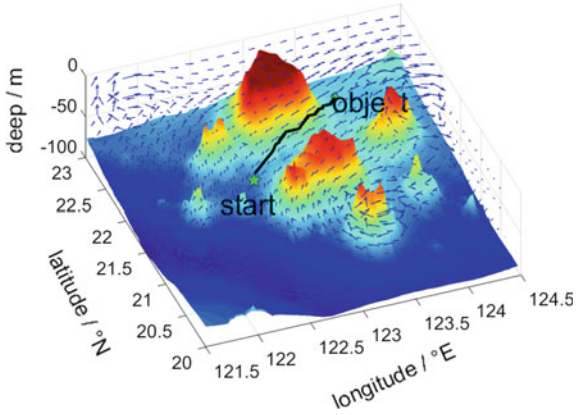


Fig. 10 Three-dimensional map of route planning results

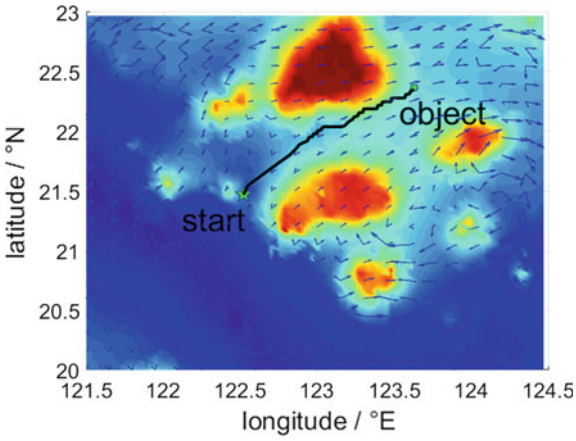


Fig. 11 Top view of route planning results

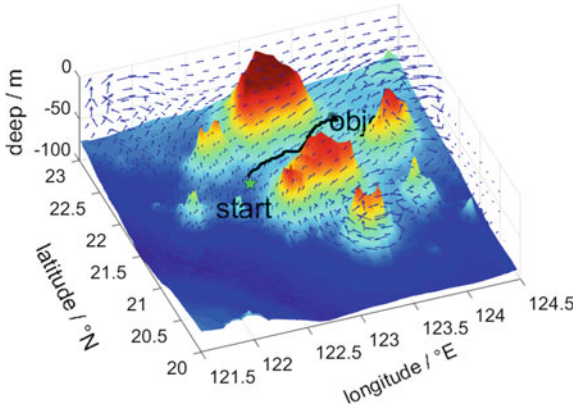


Fig. 12 Three-dimensional map of route planning results

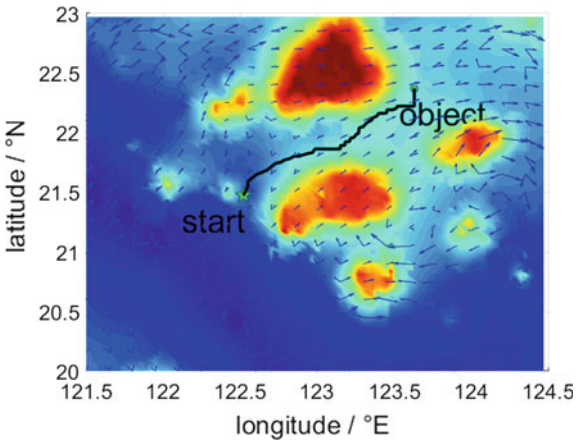


Fig. 13 Top view of route planning results

E, 22.37° N). The three-dimensional and top views of the planning results are shown in Figs. 14 and 15, respectively.

Experiment 4: When the current is large, the depth of the starting point is different from the depth of the target point. The latitude and longitude of the starting point are (122.5° E, 21.17° N), the depth is 100 m, the latitude and longitude of the target point are (123.63° E, 22.37° N), and the depth is 150 m. The three-dimensional and top views of the planning results are shown in Figs. 16 and 17, respectively.

It can be seen from the results of the above experiments that the route planning method based on the firefly algorithm can successfully search for the route from the start point to the end point in various situations. The route is smooth, and obstacles can be successfully avoided, demonstrating the effectiveness of the algorithm.

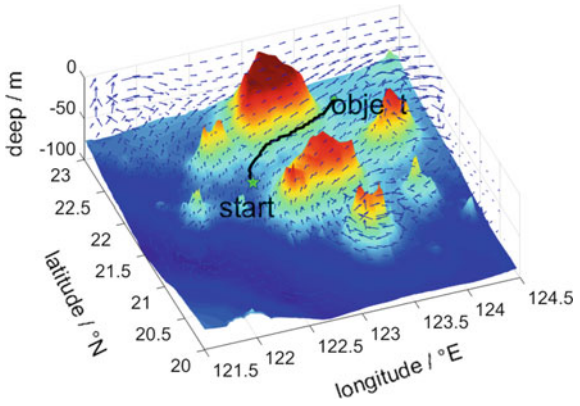


Fig. 14 Three-dimensional map of route planning results

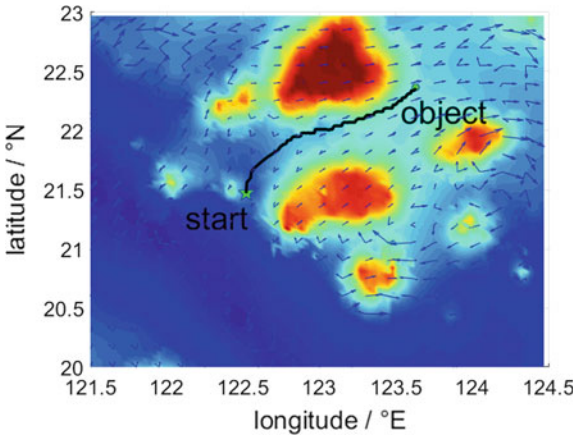


Fig. 15 Top view of route planning results

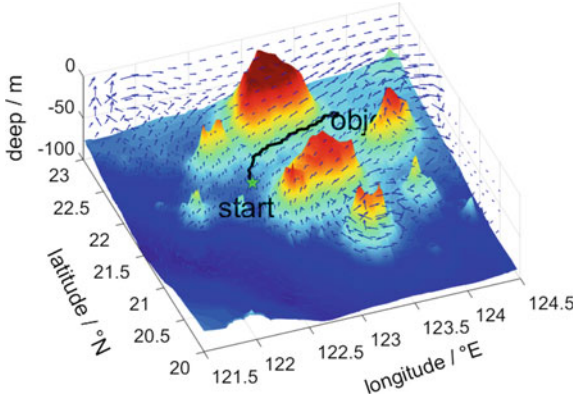


Fig. 16 Three-dimensional map of route planning results

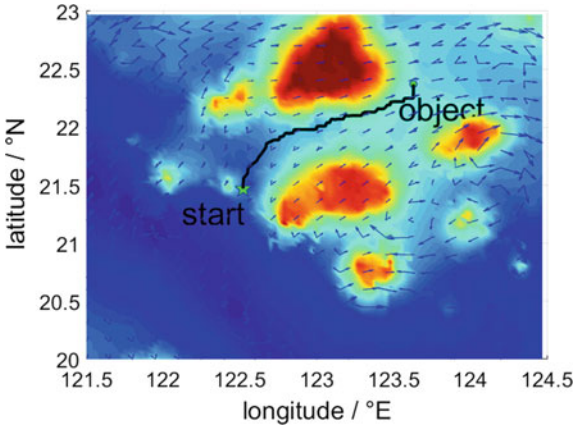


Fig. 17 Top view of route planning results

5.5.2 Algorithm Performance

The feasibility of the firefly algorithm for three-dimensional route planning is verified by the above experiments. The performance of the algorithm is tested below by comparison experiments with particle swarm optimization algorithm. The experiment uses particle swarm optimization algorithm as the comparison algorithm to compare and analyze the route planning results of the two algorithms. The selection of the simulation environment is consistent with the above. The starting point of the route is (122.5° E, 21.17° N); the depth is 100 m; the latitude and longitude of the target point are (123.63° E, 22.37° N); and the depth is 150 m.

The two algorithm populations are set to 10, and the dimensions are taken as 5, 10, 20, 30, 40, and 50, respectively, for the route planning test. If the algorithm can find the route to avoid obstacles, the search is considered successful. For each case, the stopping condition of the firefly algorithm and particle swarm optimization algorithm is 400 consecutive iterations. The algorithm runs independently 50 times, and the success rate of each algorithm, the route length, and the average time consumption under the optimal conditions are counted (Table 4).

It can be seen from the performance test result and simulation diagram that the success rate of route planning using the firefly algorithm is higher and the route length is shorter, which indicates that the algorithm has better performance. However, when the dimension is small, the success rate of the two algorithms is low. This is because the dimension of the algorithm corresponds to the number of waypoints in the planned route. The smaller dimensions would cause the algorithm to lose its obstacle avoidance ability in the area without route points, resulting in the planned route crossing obstacles. It can be seen from the average time consumption that the firefly algorithm takes longer than the particle swarm optimization, which is mainly because each firefly has to adjust its position according to the brighter firefly in the process of execution. With the increase in the dimension, the planning speed of the two algorithms is decreasing, and the time taken for the algorithm to finish increases.

Table 4 Statistics on the impact of algorithm dimensions on planning results

Dimension	Algorithm	Number of successes	Success rate (%)	Optimal route length (m)	Average time consuming (s)
5	Particle swarm optimization algorithm	14	28	1688.87	50.32
	Firefly algorithm	2	4	1688.87	57.17
10	Particle swarm optimization algorithm	14	28	1688.87	56.70
	Firefly algorithm	24	48	1688.87	70.69
20	Particle swarm optimization algorithm	30	60	1688.87	75.54
	Firefly algorithm	46	92	1688.87	101.85
30	Particle swarm optimization algorithm	50	100	1688.87	90.06
	Firefly algorithm	50	100	1688.87	134.34
40	Particle swarm optimization algorithm	50	100	1688.96	107.54
	Firefly algorithm	50	100	1688.87	163.28
50	Particle swarm optimization algorithm	50	100	1689.0	121.98
	Firefly algorithm	50	100	1688.87	196.49

Therefore, in the actual route planning, the dimension of the firefly needs to meet the actual application requirements. The experimental results show that the firefly algorithm can achieve a success rate of 100% when the dimension is 30, and the time taken for the algorithm to finish is the least.

References

1. Stentz A (1994) Optimal and efficient path planning for partially-known environments. In: 1994 IEEE international conference on proceedings of robotics and automation. IEEE
2. Petres C, Yan P, Patron P et al (2007) Path planning for autonomous underwater vehicles. IEEE Trans Rob 23(2):331–341
3. Khatib O (1986) Real-time obstacle avoidance for manipulators and mobile robots. Auton Robot Veh
4. Aghababa PM (2012) 3D path planning for underwater vehicles using five evolutionary optimization algorithms avoiding static and energetic obstacles. Appl Ocean Res 38:48–62
5. Dian-Fu Z, Fu L (2013) Research and development trend of path planning based on artificial potential field method. Comput Eng Sci 35(6):88–95
6. Li G, Yamashita A, Asama H et al (2012) An efficient improved artificial potential field based regression search method for robot path planning. In: IEEE international conference on mechatronics and automation. IEEE
7. Zhao Y, Jia R, Jin N et al (2016) A novel method of fleet deployment based on route risk evaluation. Inf Sci 372:731–744
8. Zhao Y, Li W, Shi P (2016) A real-time collision avoidance learning system for unmanned surface vessels. Elsevier Science Publishers B. V
9. Garau B, Alvarez A, Oliver G (2005) Path planning of autonomous underwater vehicles in current fields with complex spatial variability: an A* approach
10. Yang XS, Deb S, Fong S et al (2016) From swarm intelligence to metaheuristics: nature-inspired optimization algorithms. Computer 49(9):52–59
11. Zadeh LA (1965) Fuzzy sets. Inf Control 8(3):338–353
12. Holland JH (1975) Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. U Michigan Press
13. Nearchou AC (1998) Path planning of a mobile robot using genetic heuristics. Robotica 16(5):575–588
14. Dorigo M, Gambardella LM (1997) Ant colony system: a cooperative learning approach to the traveling salesman problem. IEEE Trans Evol Comput 1(1):53–66
15. Yang XS (2008) Nature-Inspired metaheuristic algorithms
16. Yang XS (2009) Firefly algorithms for multimodal optimization. Mathematics 5792:169–178
17. Yang XS (2010) Firefly algorithm, Lévy flights and global optimization. In: Research and development in intelligent systems, p XXVI
18. Yang XS (2010) Engineering optimization: an introduction with metaheuristic applications. Wiley Publishing
19. Ma Y, Zhao Y, Wu L et al (2015) Navigability analysis of magnetic map with projecting pursuit-based selection method by using firefly algorithm. Neurocomputing 159:288–297
20. Yang XS, Deb S, Zhao Y et al (2018) Swarm intelligence: past, present and future. Soft Comput
21. Bhushan B, Pillai SS (2013) Particle swarm optimization and firefly algorithm: performance analysis. In: 2013 3rd IEEE international advance computing conference (IACC). IEEE