Applied Soft Computing Journal 83 (2019) 105611

Contents lists available at ScienceDirect

Applied Soft Computing Journal

journal homepage: www.elsevier.com/locate/asoc

Water wave optimization for combinatorial optimization: Design strategies and applications

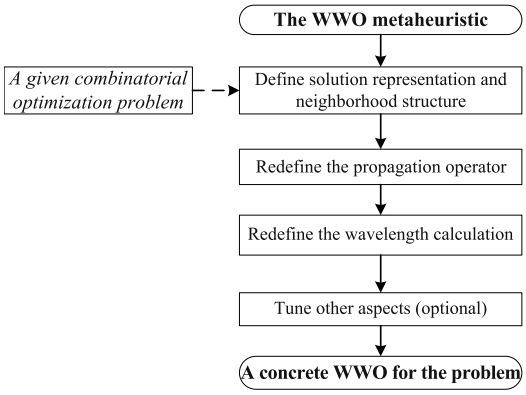
Yu-Jun Zheng [a,b,*], Xue-Qin Lu [b], Yi-Chen Du [b], Yu Xue [c,d], Wei-Guo Sheng [a]

[a] Institute of Service Engineering, Hangzhou Normal University, Hangzhou 311121, China
[b] College of Computer Science & Technology, Zhejiang University of Technology, Hangzhou 310023, China
[c] School of Computer & Software, Nanjing University of Information Science & Technology, Nanjing 210044, China
[d] School of Engineering and Computer Science, Victoria University of Wellington, Wellington 6140, New Zealand

HIGHLIGHTS

- A systematic approach for adapting WWO metaheuristic to various problems is proposed.
- The approach is validated on FSP, MKP, and an engineering optimization problem.
- Insights for designing new or adapt existing metaheuristics are provided.

GRAPHICAL ABSTRACT



ARTICLE INFO

Article history:
Received 16 August 2018
Received in revised form 17 April 2019
Accepted 24 June 2019
Available online 6 July 2019

Keywords:
Combinatorial optimization
Metaheuristics
Water wave optimization (WWO)
Algorithm design
Machine utilization optimization

ABSTRACT

Although the community of nature-inspired computing has witnessed a wide variety of metaheuristics, it often requires considerable effort to adapt them to different combinatorial optimization problems (COPs), and few studies have been devoted to reducing this burden. This paper proposes a systematic approach that consists of a set of basic steps and strategies for adapting water wave optimization (WWO), a simple and generic metaheuristic, to concrete heuristic algorithms for different COPs. Taking advantages of the generic algorithmic framework, designers can only focus on adapting the prorogation operator and the wavelength calculation method according to the combinatorial properties of the given problem, and thus easily derive efficient problem-solving algorithms. We illustrate and test our approach on the flow-shop scheduling problem (FSP), the single-objective multidimensional knapsack problem (MKP), and the multi-objective MKP, and then present an application to a machine utilization optimization problem for a large manufacturing enterprise. The results demonstrate that our approach can derive concrete algorithms that are competitive to the state-of-the-arts. Our approach also provides insights into the adaptation of other metaheuristics and the development of new metaheuristics for COPs.

1. Introduction

Combinatorial optimization problems (COPs) are among the most common problems in computer science and operations research. Unfortunately, a lot of interesting real-life COPs are

* Corresponding author at: Institute of Service Engineering, Hangzhou Normal University, Hangzhou 311121, China.
E-mail address: yujun.zheng@computer.org (Y.-J. Zheng).

https://doi.org/10.1016/j.asoc.2019.105611

NP-hard, and sharp bounds on the optima are typically hard to derive [1], which indicates that they are likely to stay intractable for exact algorithms regardless of exponentially increasing computing power [2].

Over the last two decades, nature-inspired metaheuristic optimization algorithms have become the focus of research on combinatorial optimization. Metaheuristics usually do not guarantee to catch the exact optimal solution. Instead, they are typically designed to find one or more satisfactory solutions within an acceptable time. It is widely recognized that metaheuristics are not dedicated to the solution of a particular problem, but are rather designed with the aim of being flexible enough to handle as many different problems as possible [1]. Sörensen and Glover [3] have also defined that "a metaheuristic is a high-level problem-independent algorithmic framework that provides a set of guidelines or strategies to develop heuristic optimization algorithms". That is, a metaheuristic, in its general sense, is a framework rather than a concrete algorithm. Nevertheless, in most literature, the term "metaheuristic" refers to a specific algorithm built according to its specification [2]. In fact, a number of metaheuristics, such as particle swarm optimization (PSO) [4] and differential evolution (DE) [5], are initially proposed in a form that aims to solve high-dimensional continuous function optimization problems. Although most researchers believe that those metaheuristics can be applied to any COP, it often takes a lot of effort to adapt the ingredients of these metaheuristics to the problem at hand, and such adaptations can vary from problem to problem because different problems have different combinatorial properties. So it is unsurprising that "general" comparison of metaheuristics are mostly conducted on continuous benchmark problems [6]. Another group of metaheuristics are initially proposed for some classes of COPs, e.g., ant colony optimization (ACO) [7] is initially for path routing problems such as the traveling salesman problem (TSP), and biogeography-based optimization (BBO) [8] is initially for binary selection problems. However, adapting this group to other classes of COPs can also be difficult — often more difficult than adapting them to continuous optimization [9,10].

In recent years, a variety of novel metaheuristics, such as gravitational search algorithm (GSA) [11], fireworks algorithm (FWA) [12], bat algorithm (BA) [13], drosophila food-search optimization (DFO) [14], lion optimization algorithm (LOA) [15], to name just a few, have been proposed. Almost all of them demonstrate their performance on continuous benchmark problems such as those used in IEEE CEC competitions [16]. However, little effort has been devoted to providing guidelines or strategies for adapting these heuristics to various COPs [17,18]. In [19] Dowlatshahi et al. propose a discrete GSA, the key of which is to adapt two movement operators to a given COP based on path-relinking. But they only illustrate and test their strategy on TSP. We argue that, compared to developing novel metaheuristics, facilitating the adaptation of metaheuristics to different problems can contribute more to the acceptance of metaheuristics in the industry.

Inspired by shallow water wave models, water wave optimization (WWO) [20] is also a recent novel metaheuristic initially proposed for continuous optimization. The key idea is to assign each solution (analogous to a wave) a wavelength inversely proportional to its fitness, and make each solution propagate in a range proportional to its wavelength, i.e., make low fitness solutions explore in large spaces while high fitness solutions exploit in small regions, so as to achieve a good dynamic balance between diversification and intensification. This algorithmic framework, regardless of the metaphor used, is sufficiently generic and flexible, and thus is applicable to a broad range of COPs. In the last three years, WWO has received considerable attention in the literature (e.g., [21–34]). WWO has also been introduced as a subject of the AI curriculum in our university and has aroused great interest among students. Many colleagues and students have discussed with us on the adaptation of WWO to various COPs. However, there still lacks a systematic approach for refining the generic framework into concrete algorithms for different problems. Motivated by this, we propose a set of strategies and guidelines for handling COPs based on the WWO metaheuristic in this paper. We believe such an approach would be of great benefit for the promotion and application of WWO as well as many other metaheuristics. The main contributions of this paper can be summarized as follows:

- This is the first work that regards WWO as a generic algorithm from which concrete algorithms for various COPs can be derived based on our strategies.
- The proposed approach can not only reduce the development effort but also result in efficient problem-solving algorithms.
- This work also provides insights in adapting other metaheuristics as well as developing new metaheuristics for a wide range of COPs.

In the remainder of this paper, we briefly review the basic WWO and its recent advances in Section 2, and then propose a set of basic steps and strategies for adapting WWO to COPs in Section 3, placing emphasis on the adaptation of two key ingredients, the prorogation operator and the wavelength calculation method. We illustrate and test the approach on three typical COPs in Section 4, and present an application of the approach to a real-world engineering optimization problem in Section 5. Finally Section 6 concludes with a discussion.

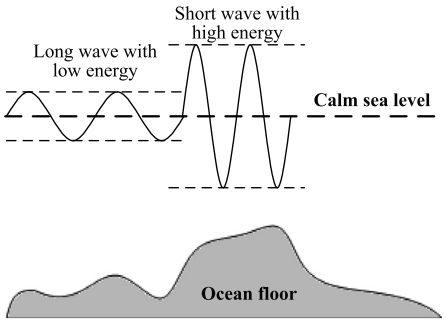


Fig. 1. Illustration of wave propagation in shallow water.

2. Review of water wave optimization

2.1. The basic WWO algorithm

WWO is a metaheuristic that takes inspiration from shallow water wave models for global optimization, where each solution $\mathbf{x}$ is analogous to a wave. The higher the energy (fitness), the smaller the wavelength $\lambda_{\mathbf{x}}$, and thus the smaller the range the wave propagates, as shown in Fig. 1. In a continuous search space, the propagation operation is done by shifting each dimension d of the solution as

$$x'(d) = x(d) + \lambda_{\mathbf{x}} \cdot rand(-1, 1) \cdot L(d) \tag{1}$$

where function *rand* produces a uniformly distributed random number, and $L(d)$ is the length of the dth dimension of the search space.

All wavelengths are initially set to 0.5, and then updated based on fitness $f(\mathbf{x})$ after each generation as

$$\lambda_{\mathbf{x}} = \lambda_{\mathbf{x}} \cdot \alpha^{-(f(\mathbf{x})-f_{\min}+\epsilon)/(f_{\max}-f_{\min}+\epsilon)} \tag{2}$$

where f_{max} and f_{min} are the maximum and minimum fitness among the population, respectively, α is the wavelength reduction coefficient which is suggested to be 1.0026, and ϵ is a very small number to avoid division by zero.

Even using only the propagation operator, WWO can efficiently solve many function optimization problems. To make it competitive with the state-of-the-arts, WWO is equipped with two other operators:

- Refraction: if a wave has not been improved after several generations, it loses its momentum and is discarded; a new wave is produced at a random position between the old wave and the best known solution $\mathbf{x}^*$.
- Breaking: whenever a new best wave $\mathbf{x}^*$ is found, it breaks into several solitary waves, each of which moves a small distance from $\mathbf{x}^*$ at a random direction.

The refraction operator, however, is discarded in an improved version of WWO [35] which gradually reduces the population size by removing the current worst solution. The breaking is essentially a local search operator. Although WWO provides a simple method for generating solitary waves (neighboring solutions), it can be replaced by any other neighborhood search method (e.g., variable and adaptive neighborhood search [36,37]). In this sense, the propagation operator can itself constitute a very simple metaheuristic based on which many powerful problem-dependent and independent algorithms can be designed, and the basic WWO can be regarded as a memetic algorithm [38]. Compared to many other metaheuristics, it can be more convenient to use WWO as "a high-level problem-independent algorithmic framework" to develop problem-dependent algorithms.

2.2. Recent advances of WWO

WWO has attracted considerable attention since it was proposed. Zhang et al. [39] improve the basic WWO by utilizing a variable population size to accelerate the search and employing a comprehensive learning mechanism in refraction to make stationary waves learn from more exemplars to increase the solution diversity. Experiments show that the improved algorithm outperforms the basic WWO and other popular metaheuristics on the CEC 2015 test set [16]. Wu et al. [23] propose an EOBWWO algorithm by incorporating the elite opposition-based learning strategy that generates an elite opposition solution for each current solution and chooses the better one between the two to enter the next iteration. Experiments show that EOBWWO has faster convergence speed. Zhang et al. [26] further improve EOBWWO by using sine and cosine curves in propagation and breaking to enhance the global search ability of WWO. There are also studies that combine WWO with other heuristics and metaheuristics including PSO [25,40], BBO [41], simulated annealing [42], firefly algorithm [22], sequential quadratic programming [33], and wind-driven optimization [34]. The WWO and its variants have been applied to many industrial problems such as neural network optimization [31,32], pattern classification and matching [43,44], feature selection [45], fuzzy clustering [46], economic dispatch [47], power dispatch [48], IIR filters design [49], supply chain design [50], partial interdiction in defensive systems [51], etc.

Another major research direction is to adapt WWO to special COPs. A class of COPs can be formulated as integer programming (IP) problems, to which metaheuristics initially proposed for continuous optimization can be easily adapted by rounding real values to integers without changing basic operators. In this manner, WWO is used to solve an IP problem in [41]. Song et al. [52] propose another WWO for IP that propagates a solution by randomly decreasing some components while increasing some others, where the decrement/increment is proportional to the wavelength. In [53] Zheng et al. propose a WWO for 0–1 programming, where the propagation is performed by randomly inverting a number of 0/1 solution components, and the number is proportional to the wavelength. For permutation-based COPs, Wu et al. [21] first adapt WWO to solve TSP, where propagation is performed by subsequence reversal and breaking is performed by local search on a swap-based neighborhood structure. Experiments show that the adapted WWO has a significant performance advantage over GA and BBO. Yun et al. [24] use a similar approach to adapt WWO to the permutation flow-shop scheduling problem (FSP), except that local search is conducted by the NEH reinsertion method [54]. In [27] Zhao et al. propose a discrete WWO for the no-wait FSP, which employs a dynamic iterated greedy method for propagation, a crossover strategy for refraction, and an insertion-based local search for local search. Experiments demonstrate that the discrete WWO outperforms a number of state-of-the-arts. Another variant of the algorithm that uses a single wave rather than a population of waves is proposed in [28]. Shao et al. [29] propose another discrete WWO for the blocking FSP by using a two-stage propagation operator that first conducts random insertions and then conduct referenced insertions. The algorithm is extended for a multi-objective blocking FSP in [30] which calculates the wavelengths based on non-dominated ranking. All these studies demonstrate the potential of WWO to solve a wide range of COPs. For the convenience of the reader, we present a brief summary of typical related work on WWO in Table 1.

3. Strategies for adapting WWO for combinatorial optimization

The basic process of using WWO to solve a given COP can be divided into four steps: (1) Defining the solution representation and the neighborhood structure(s); (2) Redefining the propagation operator based on the neighborhood structure; (3) Redefining the wavelength calculation method according to the propagation operator; (4) Adapting other aspects of the discrete optimization algorithm.

3.1. Defining the solution representation and the neighborhood structure

When solving a COP, every metaheuristic uses a solution representation, and every metaheuristic with local search uses one or more neighborhood structures. For an existing problem, there often exists definitions of solution representation and neighborhood structure in previous research, and we need only to choose an appropriate one. Taking TSP for example, the permutation representation that sequences city labels in the order in which they are visited is the mostly used representation, and an immediate neighbor of a solution can be obtained by operations such as swapping two adjacent cities, swapping a pair of cities, picking up a city and reinserting it into another place of the sequence, etc [55]. Such operations are called local search steps. For a new problem, we need to first define the representation scheme, and then define the neighborhood structure based on the representation, for which the schemes of similar problems can be helpful.

3.2. Redefining the propagation operator

The basic principle of WWO propagation is that, the better (worse) the solution, the more (less) changes made to the solution. For a COP, changes are typically made by local search steps based on neighborhood structures. We propose two basic strategies for propagating a solution $\mathbf{x}$ in a discrete search space.

Table 1
Summary of typical related work on WWO.

Category	Ref. (Pub.)	Contribution/Application
Improving basic WWO mechanisms	[35] (CEC 2015)	Variable population size
	[39] (ICIC 2015)	Comprehensive refraction
	[23] (Math. Prob. Eng.)	Opposition-based learning
	[26] (J. Intell. Fuzzy Syst.)	Enhance search with sine cosine curves
Combining with other algorithms	[25] (Neural Comput. Appl.)	Combined with PSO
	[40] (ICSI 2018)	
	[41] (Appl. Soft Comput.)	Combined with BBO
	[42] (Comput. Sci.)	Combined with SA
	[22] (Int. J. Eng. Res. Africa)	Combined with FA
	[33] (Soft Comput.)	Combined with SQP
	[34] (Appl. Intell.)	Combined with WDO
Application to specific problems	[31] (BIC-TA 2018)	Neural network optimization
	[32] (Neural Comput. Appl.)	
	[43] (ICENCO 2015)	Pattern classification/matching
	[44] (IDAACS 2015)	
	[45] (ICACI 2018)	Feature selection
	[46] (ICCSE 2018)	Fuzzy clustering
	[47] (Int. J. Intell. Eng. Syst.)	Economic dispatch
	[48] (Oper. Res.)	Power dispatch
	[49] (ICCA 2017)	IIR filters design
	[50] (Appl. Soft Comput.)	Supply chain design
	[51] (Appl. Soft Comput.)	Partial interdiction
	[53] (J. Softw.)	0-1 programming
	[52] (IEEE Trans. Ind. Inf.)	Integer programming
	[21] (ICIC 2015)	TSP
	[24] (CEC 2016)	FSP
	[27] (Expert Syst. Appl.)	No-wait FSP
	[28] (Eng. Optim)	
	[29] (Swarm Evol. Comput.)	Blocking FSP
	[30] (Knowl.-Based Syst)	Multi-objective blocking FSP

*S*1 *Changing* $\mathbf{x}$ *with a probability proportional to the wavelength* $\lambda_{\mathbf{x}}$. A typical example is the propagation proposed in [21] that modifies a solution $\mathbf{x} = \{x_1, \ldots, x_n\}$ to TSP as follows: For each i from 1 to n, with a probability of $\lambda_{\mathbf{x}}$, reverse a subsequence $\{x_i, \ldots, x_{i+l}\}$, where l is a random integer in $[1, n-i]$. In this case, $\lambda_{\mathbf{x}}$ should be a real number between [0,1].

*S*2 *Changing* $\mathbf{x}$ *by performing* k *steps of local search, where* k *is proportional to* $\lambda_{\mathbf{x}}$. For example, if we use the pair-swap neighborhood for TSP, then a propagation operation on a solution $\mathbf{x}$ can be done by randomly swapping two cities for k times, where k is a random integer in the range of $[1, \lambda_{\mathbf{x}}]$. In this case, $\lambda_{\mathbf{x}}$ should be an integer between $[1, n]$.

Strategy *S*2 is used in more studies such as [27,29,53]. Note that for some problems, the objective function of a neighbor $\mathbf{x}'$ can be evaluated based on that of the original solution $\mathbf{x}$ and the local search step(s) between them rather than evaluated from scratch. Taking TSP for example, if $\mathbf{x}'$ is obtained by swapping two adjacent cities x_i and x_{i+1}, then we have $f(\mathbf{x}') = f(\mathbf{x}) - d(x_{i-1}, x_i) - d(x_{i+1}, x_{i+2}) + d(x_{i-1}, x_{i+1}) + d(x_i, x_{i+2})$, the time complexity of which is $O(4)$, much less than $O(n)$ for re-evaluating $f(\mathbf{x}')$. Similar methods exist for neighborhoods based on city reinsertion and subsequence reversal. If $\mathbf{x}'$ is a k-step neighbor of $\mathbf{x}$, we should compare the complexity of difference-based evaluation and that of re-evaluation, and select the cheaper one (which is often the former when the problem size is large).

3.3. Redefining the wavelength calculation method

Wavelength calculation plays a key role in controlling the search process of WWO. The method shown in Eq. (2) is for searching continuous spaces, where $\alpha = 1.0026$ is a "magic number" obtained through a large number of experiments — we still do not understand why this number works so effectively. In general, Eq. (2) has two effects: the first is to gradually decrease the average wavelength of the solutions with the number of iterations, and thus makes worse solutions explore large areas in early stages and exploit small regions in later stages; the second is to make the wavelengths of better (bad) solutions decrease faster (slower) to enhance intensification (diversification). The two effects, accumulated over iterations, can achieve a very good balance between intensification and diversification.

However, for COPs we only need the second effect, because the search range of a discrete space cannot be decreased infinitely as in continuous spaces. So we do not suggest adapting Eq. (2) by only tuning the parameter α. Here we propose three strategies for calculating wavelengths for a COP.

*S*3 *Setting* $\lambda_{\mathbf{x}}$ *inversely proportional to the solution fitness* $f(\mathbf{x})$ *based on a linear model.* Two typical forms are shown in Eqs. (3) and (4), where $\mathbf{P}$ denotes the population of solutions, and $\lambda_{\min}$ and $\lambda_{\max}$ are the minimum and maximum allowable wavelength, respectively.

$$\lambda_{\mathbf{x}} = \lambda_{\max} \frac{\left(\sum_{\mathbf{x}' \in \mathbf{P}} f(\mathbf{x}')\right) - f(\mathbf{x})}{\sum_{\mathbf{x}' \in \mathbf{P}} f(\mathbf{x}')} \tag{3}$$

$$\lambda_{\mathbf{x}} = \lambda_{\min} + (\lambda_{\max} - \lambda_{\min}) \frac{f_{\max} - f(\mathbf{x}) + \epsilon}{f_{\max} - f_{\min} + \epsilon} \tag{4}$$

Studies in [21] and [56] use such a strategy.

*S*4 *Setting* $\lambda_{\mathbf{x}}$ *inversely proportional to* $f(\mathbf{x})$ *based on an exponential model.* A typical form is shown in Eq. (5):

$$\lambda_{\mathbf{x}} = \lambda_{\min} \cdot b^{\alpha \cdot (f_{\max} - f(\mathbf{x}) + \epsilon)/(f_{\max} - f_{\min} + \epsilon)} \tag{5}$$

where b and α are two control parameters. For convenience α is often set to 1, and then b is tuned as $\lambda_{\max}/\lambda_{\min}$. Study in [29] uses an equation like this. In [41] we extend it to handle a multiobjective optimization problem as follows:

$$\lambda_{\mathbf{x}} = b^{\alpha \cdot (rank(\mathbf{x})/rank_{\max})} \tag{6}$$

where $rank(\mathbf{x})$ is the non-dominated rank of $\mathbf{x}$, and $rank_{\max}$ is the total number of non-dominated fronts of the population [57].

S5 *Updating $\lambda_{\mathbf{x}}$ according to the change of $f(\mathbf{x})$.* If $f(\mathbf{x})$ increases (decreases), then $\lambda_{\mathbf{x}}$ decreases (increases) correspondingly. Yun et al. [24] use an update method as follows:

$$\lambda_{\mathbf{x}'} = \min\left(\lambda_{\mathbf{x}} + \alpha\frac{f(\mathbf{x}) - f(\mathbf{x}')}{f_{\max}}, \lambda_{\max}\right) \tag{7}$$

In [27] Zhao et al. use another form that compares the new solution with its historical personal best (*pbest*) rather than its parent.

We should be careful of the range of $\lambda_{\mathbf{x}}$ and, if needed, set the lower and upper limits explicitly. When propagation is based on strategy *S*2, $\lambda_{\mathbf{x}}$ is a positive real number and $\lambda_{\max}$ is typically set to 1; when propagation is based on *S*3, $\lambda_{\mathbf{x}}$ is a positive integer, $\lambda_{\min}$ is typically set to 1, and $\lambda_{\max}$ is often set according to the problem size.

3.4. Adapting other aspects of the discrete algorithm

Our strategies for adapting the breaking operator for COPs is also based on neighborhood search:

S6 *Generating a set of n_b immediate neighbors of the best solution newly found.* In general, using a constant n_b is sufficiently good for most problems, although it can be dynamically adjusted for further performance improvement.

Note that the neighborhood structure used for breaking is not necessarily the same as that for propagation: An immediate neighbor in terms of breaking can be "closer" to the original solution than that of propagation, because breaking focuses on intensification while propagation emphasizes the balance between diversification and intensification. For example, for TSP we can use swapping two adjacent cities for breaking while use swapping two random cities for propagation.

For solution update, we propose the following three alternative strategies:

S7 *Replace a solution $\mathbf{x}$ with the propagated solution $\mathbf{x}'$ only if $\mathbf{x}'$ is better.* This is recommended in most cases.

S8 *Replace a solution $\mathbf{x}$ with the propagated solution $\mathbf{x}'$ if $\mathbf{x}'$ is better or $\exp\left(\left(f(\mathbf{x}') - f(\mathbf{x})\right)/T\right)$ is larger than a random number in [0, 1] (as the acceptance condition used in simulated annealing [42]).* This can sometimes slightly improve the algorithm performance, but includes an additional control parameter and thus increases the design burden.

S9 *Replace a solution $\mathbf{x}$ with the propagated solution $\mathbf{x}'$ if $\mathbf{x}'$ is better or $\mathbf{x}'$ can improve the population diversity (e.g., the distance between $\mathbf{x}'$ and its closest individual is larger than that of $\mathbf{x}$).* This is recommended for multimodal or multiobjective optimization where the solution diversity is of critical importance.

In order to remove low-quality or stagnant solutions, we suggest using population size reduction as the simplified WWO [35], i.e.:

S10 *Reducing the population size NP from $NP_{\max}$ to $NP_{\min}$ by iteratively removing the current worst solution.* Depending on the problem size, $NP_{\max}$ is typically set between 20–100, and $NP_{\min}$ is typically 1/4 to 1/2 of *NP*.

Using population size reduction, the refraction operator is not necessary for most problems we studied, although users can design an elaborate refraction operator to pursue further performance improvement.

The above steps and strategies constitute a general framework that is applicable to a majority of COPs. There are many additional strategies that require much more design effort to further enhance the algorithm performance, including using special solution initialization methods, using adaptive neighborhood structures in breaking, integrating with evolutionary operators of other metaheuristics, etc. In this paper we will not discuss them in detail.

4. Illustration and evaluation on three typical problems

We select three COPs, including two single-objective COPs, FSP and the multidimensional knapsack problem (MKP), and a multiobjective MKP (MOMKP), to illustrate and test our approach for designing discrete WWO algorithms. Note that we focus on the easy adaptation of the WWO metaheuristic to the problems, and thus we do not intend to propose "specially crafted" algorithms to all state-of-the-art algorithms.

4.1. Adapting WWO for FSP

FSP is to schedule n independent jobs, each of which contains exactly m operations to be sequentially executed on m machines. The processing time t_{ij} of each jth operation of each job i is known. A solution $\mathbf{x} = \{x_1, \ldots, x_n\}$ to the problem is a sequence (permutation) of the n jobs, and the completion time $C(x_i, j)$ of each jth operation of each job x_i can be iteratively calculated as:

$$C(x_1, 1) = t_{x_1,1} \tag{8}$$

$$C(x_i, 1) = C(x_{i-1}, 1) + t_{x_i,1}, \quad 2 \le i \le n \tag{9}$$

$$C(x_1, j) = C(x_1, j-1) + t_{x_1,j}, \quad 2 \le j \le m \tag{10}$$

$$C(x_i, j) = \max\left(C(x_{i-1}, j), C(x_i, j-1)\right) + t_{x_i,j}, \quad 2 \le i \le n; 2 \le j \le m \tag{11}$$

The objective is to minimize the completion time of last job:

$$\min f(\mathbf{x}) = C(x_n, m) \tag{12}$$

We use the following steps to design discrete WWO for FSP.

(1) Select the basic permutation-based representation and the subsequence reversal neighborhood.
(2) Use strategy *S*2 for the propagation operator based on subsequence reversal.
(3) Use strategy *S*3 for wavelength calculation.
(4) Use strategy *S*6 to design the breaking operator.
(5) Use strategy *S*7 for solution update.
(6) Use strategy *S*10 to linearly decrease the population size.

We implement two versions of WWO. The first version focuses on using the least effort to adapt WWO to the problem, and its breaking operator generates an immediate neighbor by reinserting a random job with waiting time larger than the average waiting time of all jobs. The second version (denoted by WWO-M) aims to improve the algorithm performance by using a self-adaptive breaking operator, which dynamically selects among three candidate operations including re-inserting a job, swapping two adjacent jobs, and the NEH heuristic [54] based on their past performance. That is, initially the three operations have the same selection probability; after the first *LP* generations (where *LP* is a parameter called learning period), the selection probability of each operation is set proportional to its success rate (i.e., the percentage of the number of invocations that produce better solutions) during the previous *LP* generations. The resulting algorithm is presented in Alg. 1 (the two versions use different local search operators in Line 11).

Algorithm 1: The WWO algorithm for FSP.

```
Randomly initialize a population of NP solutions to the problem;
Let x* be the best among the solutions;
while the stopping condition is not met do
    Calculate the wavelengths of the solutions;
    foreach x in the population do
        Let k = rand(1, λ_x);
        Propagate x to a new x' by performing k random reversals;
        if f(x') < f(x) then
            Replace x with x' in the population;
            if f(x) < f(x*) then
                Perform a local search around x;
                Replace x* with the best among x and its neighbors;
    Update the population size;
return x*.
```

As wavelength plays a key role in WWO, here we first test the different wavelength calculation methods, i.e., Eqs. (3), (4), (5), and (7), as well as different values of parameter $\lambda_{\max}$ for WWO. We use 12 test instances of FSP from [58]. The stopping condition is that the number of objective function evaluations (NFE) reaches $100mn$, $NP_{\max}$ is set to $\min(3n, 100)$, $NP_{\min}$ is set to 18, and the learning period of WWO-M is set to 30. The performance is evaluated in terms of the relative percentage deviation (RPD) of the objective function values obtained by the algorithms from the known best one. Fig. 2 presents the median RPD (over 30 random runs) of the algorithm versions with $\lambda_{\max}$ ranging from $n/2$ to n. The results show that, on the test set, WWO using Eq. (7) that updates wavelengths according to fitness changes exhibits the worst performance. The two versions using linear equations (3) and (4) exhibit similar performance, and both of them achieve the best performance when $\lambda_{\max}$ is set around $0.9n$. In comparison, the version using exponential equation (5) is more sensitive to $\lambda_{\max}$, and if its $\lambda_{\max}$ value is fine-tuned to around $0.75n \sim 0.8n$, it exhibits better performance than the linear versions. Moreover, by using adaptive local search, WWO-M versions are less sensitive to $\lambda_{\max}$ than the corresponding WWO versions, and almost all algorithm versions are less sensitive to $\lambda_{\max}$ on large-size instances than that on small-size instances.

After the sensitivity test, we employ Eq. (4) with $\lambda_{\max} = 0.9n$ for WWO, and employ Eq. (5) with $\lambda_{\max} = 0.8n$ for WWO-M. Next we compare the two WWO versions with the following nine classical or state-of-the-art metaheuristics for FSP:

- An ordered-based GA [59].
- A discrete PSO algorithm [60].
- A discrete DE algorithm [61], for which we implement two versions, a basic version without enhanced local search and a version enhanced with NEH and referenced local search (denoted by DE-M).
- A discrete cuckoo search algorithm (CSA) [62].
- A teaching–learning-based optimization (TLBO) algorithm [63], for which we implement two versions, a basic version without enhanced local search and a version enhanced with insert local search based on simulated annealing (denoted by TLBO-M).
- A BBO algorithm using ecogeography-based migration [64] which outperforms other BBO variants [65].
- A hybrid ACO (HACO) algorithm [66].

All algorithms use the same stopping condition that NFE reaches $100mn$. Each algorithm is run 30 times on each instance. Table 2 presents the medians (med) and standard deviations (std) of the RPD values of the algorithms, and the box plots in Fig. 3 also show the maximum, minimum, the first quartile (Q1) and the third quartile (Q3) values. The comparison can be divided into two groups:

- In the first group, we compare WWO with the first six algorithms without dedicated local search operators, and mark a superscript † before a median value of a comparative algorithm to indicate that the result of the algorithm is significantly different from that of WWO (at a confidence level of 95%, according to the nonparametric Wilcoxon rank sum test). The results show that WWO obtains the best median RPD on all 12 instances. On the smallest-size instance 1, except DE, all other algorithms always obtain the exact optimal solution (RPD = 0). Among the remaining 11 instances, WWO achieves significant performance improvements over GA on all 11 instances, over PSO, DE, and CSA on 10 instances, over TLBO on 8 instances, and over BBO on 6 instances. In summary, the overall performance of the WWO derived by the proposed approach is significantly better than the other six well-known metaheuristics on the test set.
- In the second group, we compare WWO-M with the other three algorithms with dedicated local search, and use a superscript † before a median value to indicate that the result of the corresponding algorithm is significantly different from that of WWO-M. The results show that the algorithm performances are much improved by equipping with dedicated local search operators, and WWO-M also obtains the best median RPD on all 12 instances. In this group, all four algorithms obtain the exact optimal solutions on instances 4 and 7, and the three algorithms (except DE-M) also do so on instances 1, 2, and 3. On the remaining 7 instances, WWO-M always achieves significant performance improvements over DE-M and TLBO-M. HACO exhibits competitive performance on the test set, but WWO-M still significantly outperforms HACO on three instances. This also demonstrates that the WWO framework can be used to derive very efficient concrete algorithms by using dedicated mechanisms such as adaptive local search.

4.2. Adapting WWO for single-objective MKP

As a generalization of the basic knapsack problem, MKP is to choose a subset of n items whose profit sum is maximized while m knapsack constraints are satisfied:

$$\max f(\mathbf{x}) = \sum_{i=1}^{n} p_i x_i \tag{13}$$

$$\text{s.t.} \sum_{i=1}^{n} w_{ij} x_i \le c_j, \quad j = 1, \ldots, m \tag{14}$$

$$x_i \in \{0, 1\}, \quad i = 1, \ldots, n \tag{15}$$

where p_i, w_{ij} and c_j are all non-negative numbers.

We use the following steps to design discrete WWO for MKP.

(1) Select the basic n-dimensional 0–1 vector representation and the bit inversion neighborhood.
(2) Use strategy $S2$ for the propagation operator based on bit reversal (i.e., changing a component $x_i = 1 - x_i$).
(3) Use strategy $S3$ for wavelength calculation.
(4) Use strategy $S6$ to design the breaking operator.

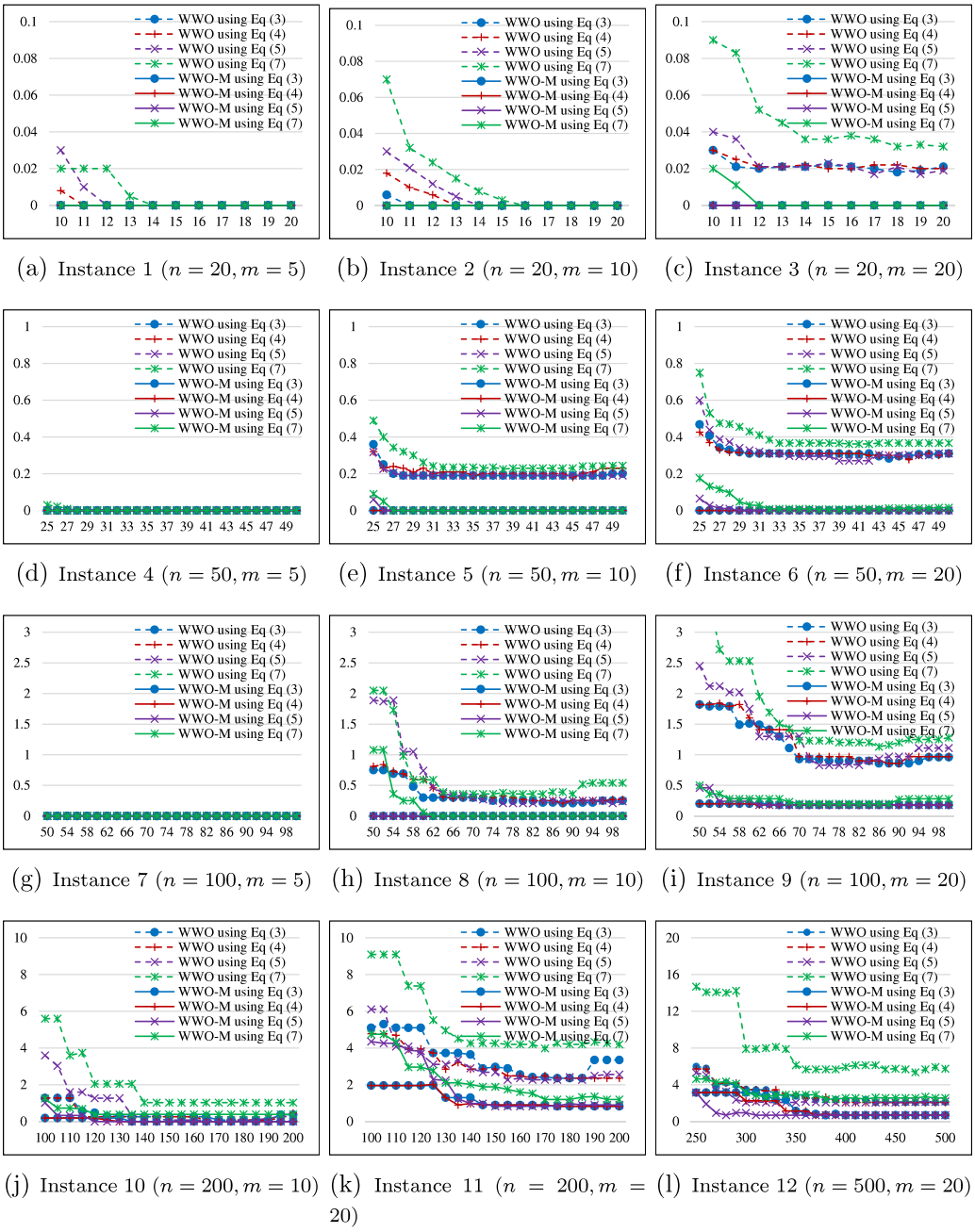


Fig. 2. Comparison of WWO algorithms using different wave calculation methods and λ_{max} values on FSP test instances. The x-axis denotes the λ_{max} value, and the y-axis denotes the RPD (%) value.

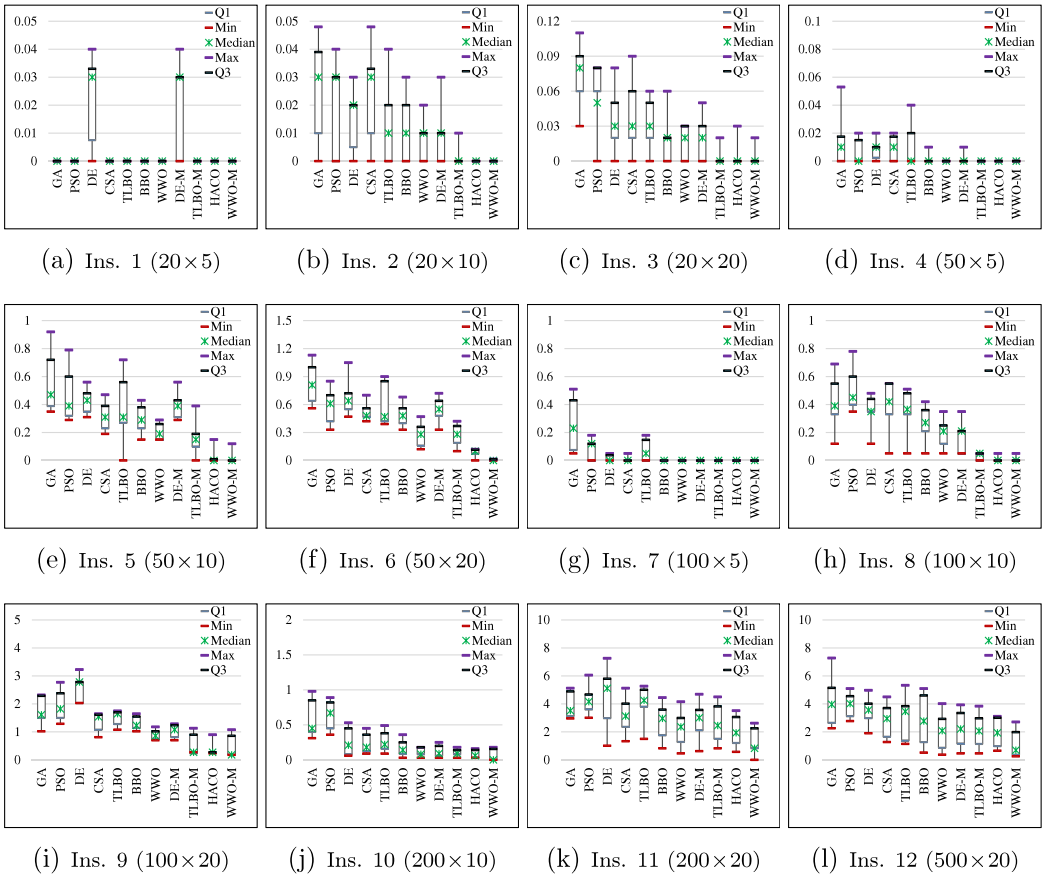
(5) Use strategy $S7$ for solution update.
(6) Use strategy $S10$ to linearly decrease the population size.

We also implement two WWO versions, one version using a single breaking operator that produces a neighbor by temporarily removing the item of the kth smallest profit ($1 \leq k \leq n_b$) and then trying to add other items in decreasing order of profit until there is no item can be added to the knapsack, and another version denoted by WWO-M that dynamically selects among three candidate breaking operators including replacing the kth smallest profit item, reversing a component for a higher profit, and interchanging two opposite components for a higher profit. The resulting algorithm is presented in Alg. 2 (the two versions use different local search operators in Line 11).

Similarly, we first test the effects of different wavelength calculation methods, i.e., Eqs. (3), (4), (5), and (7), as well as different values of λ_{max} (ranging from $n/2$ to n) on WWO. We use 10 test instances from OR-Library [67] and [68]. The constraints are handled by the penalty function method [68]. The stopping condition is that NFE reaches $50mn$. We set NP_{max} to $5m\ln(n/2)$, NP_{min} is set to 12, and LP of WWO-M to 30. On the smallest-size instance 1, all versions always achieve RPD $= 0$. Fig. 4 presents the median RPD (over 30 random runs) of the different versions on the remaining nine instances. On the MKP instances, WWO using Eq. (7) that updates wavelengths according to fitness changes also exhibits the worst performance. On most instances, the two versions using linear equations (3) and (4) and the version using exponential equation (5) exhibit similar performance, and they obtain the best RPD when λ_{max} is set around $0.7n \sim 0.9n$. The exponential version occasionally achieves slightly better results than the linear versions on instances 9 and 10, but it requires more tuning efforts. Comparatively, WWO-M versions are much

Table 2
The results of the comparative algorithms on FSP test instances.

Ins. ($n \times m$)		GA	PSO	DE	CSA	TLBO	BBO	WWO	DE-M	TLBO-M	HACO	WWO-M
1 (20 × 5)	med	**0**	**0**	†0.03	**0**	**0**	**0**	**0**	†0.03	**0**	**0**	**0**
	std	0	0	0.02	0	0	0	0	0.02	0	0	0
2 (20 × 10)	med	†0.03	†0.03	†0.02	†0.03	0.01	0.01	0.01	†0.01	**0**	**0**	**0**
	std	0.02	0.02	0.01	0.02	0.01	0.01	0.01	0.01	0.00	0	0
3 (20 × 20)	med	†0.08	†0.05	†0.03	†0.03	†0.03	0.02	0.02	†0.02	**0**	**0**	**0**
	std	0.02	0.03	0.02	0.03	0.02	0.01	0.01	0.01	0.00	0.01	0.00
4 (50 × 5)	med	†0.01	**0**	†0.01	†0.01	**0**	**0**	**0**	**0**	**0**	**0**	**0**
	std	0.02	0.01	0.01	0.01	0.01	0.00	0	0.00	0	0	0
5 (50 × 10)	med	†0.47	†0.39	†0.43	†0.31	0.31	0.29	0.19	†0.39	†0.15	**0**	**0**
	std	0.30	0.26	0.13	0.14	0.27	0.16	0.03	0.12	0.11	0.07	0.04
6 (50 × 20)	med	†0.81	†0.61	†0.64	†0.48	†0.47	†0.48	0.28	†0.55	†0.28	†0.1	**0**
	std	0.29	0.25	0.23	0.08	0.31	0.13	0.15	0.09	0.11	0.05	0.01
7 (100 × 5)	med	†0.23	†0.12	**0**	**0**	†0.05	**0**	**0**	**0**	**0**	**0**	**0**
	std	0.2	0.07	0.02	0.02	0.08	0	0	0	0	0	0
8 (100 × 10)	med	†0.39	†0.45	†0.35	†0.42	†0.37	†0.27	0.21	†0.21	†0.05	**0**	**0**
	std	0.21	0.18	0.09	0.19	0.15	0.13	0.09	0.11	0.02	0.02	0.01
9 (100 × 20)	med	†1.61	†1.82	†2.78	†1.54	†1.65	†1.23	0.86	†1.08	†0.27	0.27	**0.18**
	std	0.49	0.55	0.35	0.49	0.32	0.31	0.16	0.22	0.37	0.194	0.41
10 (200 × 10)	med	†0.45	†0.67	†0.21	†0.18	†0.22	†0.14	0.08	†0.09	†0.09	0.06	**0**
	std	0.24	0.22	0.27	0.18	0.16	0.11	0.06	0.1	0.07	0.05	0.08
11 (200 × 20)	med	†3.52	†4.16	†5.11	†3.13	†4.25	†2.96	2.36	†3.01	†2.45	†1.93	**0.83**
	std	1.05	0.68	2.26	1.18	0.89	1.06	1.39	1.35	1.15	1.28	1.23
12 (500 × 20)	med	†3.96	†4.02	†3.56	†2.95	†3.46	†2.77	2.08	†2.21	†2.06	†1.93	**0.69**
	std	1.79	1.35	0.91	1.20	1.44	1.63	1.12	1.26	1.12	1.35	0.98



(a) Ins. 1 (20×5) (b) Ins. 2 (20×10) (c) Ins. 3 (20×20) (d) Ins. 4 (50×5)

(e) Ins. 5 (50×10) (f) Ins. 6 (50×20) (g) Ins. 7 (100×5) (h) Ins. 8 (100×10)

(i) Ins. 9 (100×20) (j) Ins. 10 (200×10) (k) Ins. 11 (200×20) (l) Ins. 12 (500×20)

Fig. 3. Box plots of the results of the comparative algorithms on FSP test instances.

less sensitive to λ_{max}, and they obtain the best RPD when λ_{max} is set around $0.6n \sim n$ on most instances.

After the sensitivity test, we employ Eq. (4) with $\lambda_{max} = 0.9n$ for WWO, and employ Eq. (5) with $\lambda_{max} = 0.75n$ for WWO-M. Next we compare the two WWO versions with following seven metaheuristic algorithms for MKP:

- A basic GA [68].
- A binary PSO algorithm [69].
- A binary fruit fly optimization (FFO) algorithm [70].
- An ACO algorithm combined with local search [71].
- A hybrid harmony search (HHS) algorithm with local search [72].

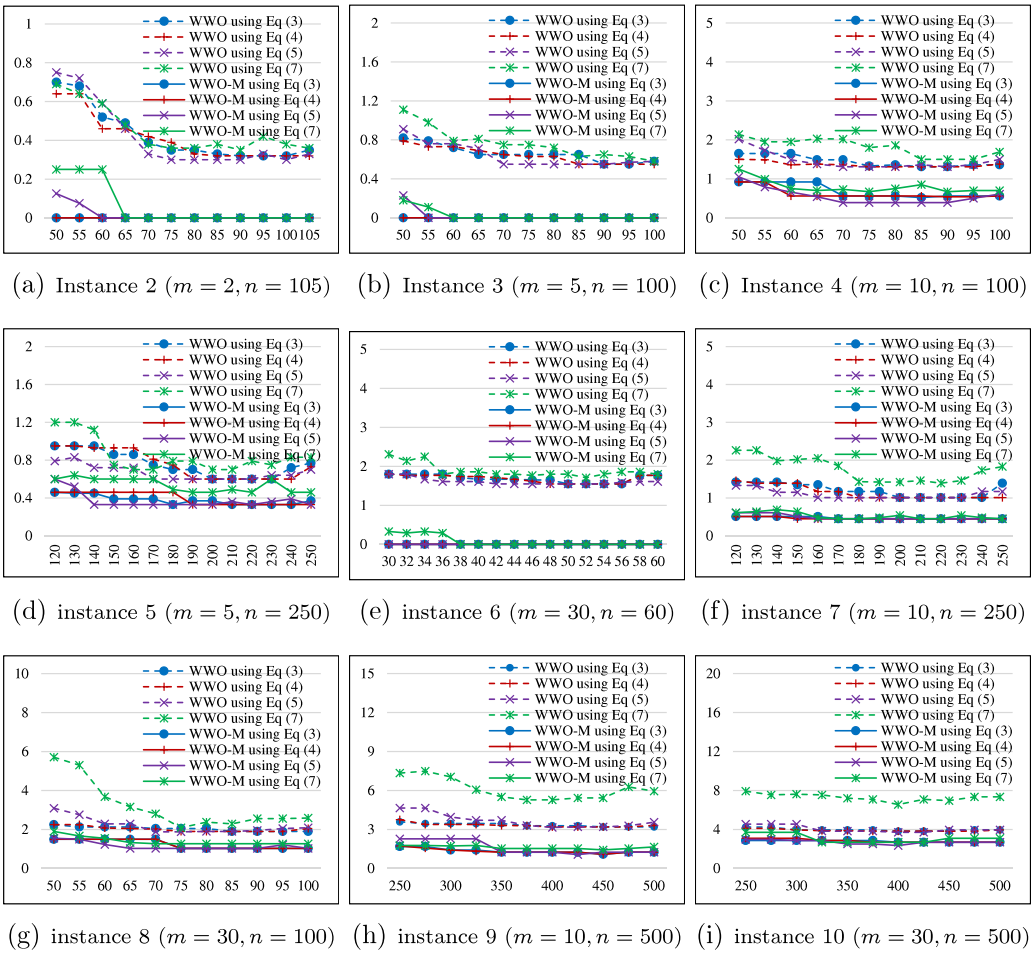


Fig. 4. Comparison of WWO algorithms using different wave calculation methods and λ_{max} values on MKP test instances. The *x*-axis denotes the λ_{max} value, and the *y*-axis denotes the RPD (%) value.

Algorithm 2: The WWO algorithm for MKP.

```
Randomly initialize a population of NP solutions to the problem;
Let x* be the best among the solutions;
while the stopping condition is not met do
    Calculate the wavelengths of the solutions;
    foreach x in the population do
        Let k = rand(1, λ_x);
        Propagate x to a new x' by randomly reversing k components;
        if f(x') > f(x) then
            Replace x with x' in the population;
            if f(x) > f(x*) then
                Perform a local search around x;
                Replace x* with the best among x and its neighbors;
    Update the population size;
return x*.
```

- A binary artificial algae algorithm (BAAA) with elite local search [73].
- A hybrid GA (HGA) with local search based on simulated annealing [74].

All algorithms use the same stopping condition that NFE reaches 50*mn*. Each algorithm is run 30 times on each instance. The experimental results are given in Table 3 and the box plots in Fig. 5. The comparison is also divided into two groups:

- The first four algorithms are without dedicated local search operators. On instance 1, PSO, FFO, and WWO always obtain the exact optimal solution. On the remaining nine instances, FFO and WWO both obtain the best median value on instances 2 and 5, PSO uniquely obtains the best median value on instance 6, FFO uniquely obtains the best median value on instances 3 and 4, and WWO uniquely obtains the best median value on instances 7, 8, 9, and 10. That is, FFO performs better on small- and medium-size instances, WWO performs better on large-size instances, and the overall performance of WWO is the best among the four algorithms on the test set. According to the nonparametric Wilcoxon rank sum test, WWO performs significantly better than GA on all ten instances, better than PSO on six instances, and better than FFO on four instances. On the contrary, there is only one case that WWO performs significantly worse than PSO on instance 6. The results demonstrate the competitive performance of the WWO derived by the proposed approach.
- The last five algorithms are equipped with dedicated local search. On instances 1 and 6, all five algorithms always obtain the exact optimal solutions. ACO exhibits the worst performance, which is significantly worse than WWO-M on the remaining eight instances. On instance 2, the four

algorithms (except ACO) always obtain the exact optimal solution. On instance 3, BAAA, HGA, and WWO-M always obtain the exact optimal solution. WWO performs significantly better than HGA on all the remaining six instances and better than HHS on three instances. There is only one case that WWO performs significantly worse than HHS on instance 5. The overall performances of BAAA and WWO-M are the best among all comparative algorithms, and there is no significant difference between the results of BAAA and WWO-M. This also demonstrates that the promising performance of WWO combined with adaptive local search for the MKP.

4.3. Adapting WWO for MOMKP

MOMKP extends its single-objective MKP by using M objectives [75]:

$$\max f(\mathbf{x}) = \sum_{i=1}^{n} p_{ik} x_i, \quad k = 1, \ldots, M \tag{16}$$

$$\text{s.t.} \sum_{i=1}^{n} w_{ij} x_i \leq c_j, \quad j = 1, \ldots, m \tag{17}$$

$$x_i \in \{0, 1\}, \quad i = 1, \ldots, n \tag{18}$$

where p_{ik} is the profit of item i relative to the objective k.

To design discrete WWO for MOMKP, we use the same steps as in Section 4.2, except that

- We use Eq. (6) to calculate wavelength calculation based on non-dominated sorting.
- We perform breaking on each new solution in the first non-dominated front.
- We use strategy S9 for solution update.

Similarly, we implement two WWO versions, one version (denoted by MOWWO) using a single breaking operator that produces a neighbor by temporarily removing the item of the smallest ratio of total profit to total weight and then trying to add other items in decreasing order of this ratio to the knapsack, and another version (denoted by MOWWO-M) dynamically selecting among three candidate breaking operators including replacing the item of the smallest ratio of total profit to total weight, reversing a component for a higher total profit, and interchanging two opposite components for a higher total profit. The resulting algorithm is presented in Alg. 3 (the two versions use different local search operators in Line 12).

Algorithm 3: The MOWWO algorithm for MOMKP.

```
1  Randomly initialize a population of NP solutions to the problem;
2  Let x* be the best among the solutions;
3  while the stopping condition is not met do
4      Perform non-dominated sorting on the solutions;
5      Calculate the wavelengths of the solutions according to Eq. (6);
6      foreach x in the population do
7          Let k = rand(1, λ_x);
8          Propagate x to a new x' by randomly reversing k components;
9          if rank(x') < rank(x) or (rank(x') = rank(x) and D(x') > rank(x)) then
10             Replace x with x' in the population;
11             if rank(x) = 1 then
12                 Perform a local search around x;
13                 Replace x* with the best among x and its neighbors;
14     Update the population size;
15 return x*.
```

We select three bi-objective test instances from [76], which use 250, 500 and 750 items, respectively. The following well-known multi-objective optimization algorithms are used for comparison:

- A multiobjective evolutionary algorithm based on decomposition (MOEA/D) [77].
- A two-phase Pareto local search (2PPLS) algorithm [75].
- A multi-population iterated local search (MILS) algorithm [78].
- A multi-population based cooperative algorithm Wϵ-CMOLS [79].
- A gradual weight-based ACO (Gw-ACO) algorithm [80].

For MOWWO, we set $NP_{\max} = 120$, $NP_{\min} = 12$, $b = m$, and $\alpha = 0.36$. The learning period of MOWWO-M is set to 30. All algorithms use the same stopping condition that NFE reaches $20m$, and each algorithm is run 30 times on each instance.

The following three metrics are used for comparing the multi-objective algorithms:

- The hypervolume $\mathcal{H}$.
- The average distance $\mathcal{D}$ between the reference set and the first non-dominated set obtained.
- The proportion $\mathcal{P}_N$ (in percentage) of non-dominated points obtained.

Table 4 presents the results of the algorithms on the test instances, where a † before a mean value indicates that the metric of the corresponding algorithm is significantly different from that of MOWWO-M (at a confidence level of 95%). From the different metrics, we can observe that:

- In terms of hypervolume, 2PPLS obtains the best mean values on the first two instances, and Wϵ-CMOLS obtains the best mean value on the third instance. MOWWO-M exhibits relatively good performance, as it ranks third on the first two instances and ranks the second on the third instance. There is also no statistically significant difference between MOWWO-M and the top algorithm on each instance. The performance of MOWWO-M is significantly better than MOEA/D on all instances and better than MOWWO on the third instance, which validates the effectiveness of the self-adaptive breaking operator in improving solution accuracies.
- In terms of average distance, MILS obtains the best mean values on the first and third instances, and Wϵ-CMOLS obtains the best mean value on the second instance. MOWWO-M ranks fourth, third, and third on the three instances, respectively.
- In terms of proportion $\mathcal{P}_N$, GwACO obtains the best mean values on the first two instances, where there is no significant difference between GwACO and MOWWO-M. MOWWO-M obtains the best mean value on the third instance.

In summary, no algorithm can exhibit the best performance in terms of all three metrics. 2PPLS obtains the best hypervolume results, but obtains the worst average distances. Comparatively, the overall performance of Wϵ-CMOLS, GwACO, and MOWWO-M are promising. The results demonstrate the effectiveness of the proposed approach for adapting WWO to multi-objective COP.

Table 3
The results of the comparative algorithms on MKP test instances.

Ins. ($n \times m$)		GA	PSO	FFO	WWO	ACO	HHS	BAAA	HGA	WWO-M
1 (2 × 28)	med	†0.39	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
	std	0.24	0	0	0	0	0	0	0	0
2 (2 × 105)	med	†1.50	0.36	0.32	0.32	†0.25	**0**	**0**	**0**	**0**
	std	0.39	0.25	0.30	0.44	0.22	0.1	0	0	0
3 (5 × 100)	med	†0.85	†0.74	0.37	0.55	†0.02	**0**	**0**	**0**	**0**
	std	0.53	0.17	0.26	0.33	0.14	0.12	0	0	0
4 (10 × 100)	med	†1.93	†1.33	1.27	1.31	†1.18	0.37	**0.29**	†1.05	0.39
	std	0.37	0.27	0.45	0.6	0.52	0.38	0.39	0.51	0.22
5 (5 × 250)	med	†0.86	†0.95	0.6	0.6	†0.42	−**0.23**	0.33	†0.55	0.33
	std	0.31	0.57	0.51	0.26	0.31	0.23	0.15	0.2	0.18
6 (30 × 60)	med	†5.14	−0.65	†1.83	1.55	**0**	**0**	**0**	**0**	**0**
	std	1.25	0.68	0.6	0.71	0.12	0	0	0.26	0
7 (10 × 250)	med	†1.57	†1.89	†1.15	1.01	†1.08	†0.61	0.48	†1.14	**0.45**
	std	0.82	0.7	0.72	0.51	0.36	0.33	0.28	0.35	0.25
8 (30 × 100)	med	†3.61	2.07	†2.29	1.9	†1.88	†1.23	**1.03**	†1.67	**1.03**
	std	2.21	1.57	1.43	0.99	1.02	0.35	0.24	0.4	0.21
9 (10 × 500)	med	†3.73	†3.46	3.31	3.2	†3.09	1.41	1.25	†1.73	**1.07**
	std	1.44	1.35	1.31	1.29	0.98	0.41	0.36	0.95	0.41
10 (30 × 500)	med	†5.27	†4.66	†3.96	3.83	†3.71	†2.69	2.43	†2.8	**2.35**
	std	1.98	1.03	0.89	0.7	0.69	0.6	0.46	0.68	0.48

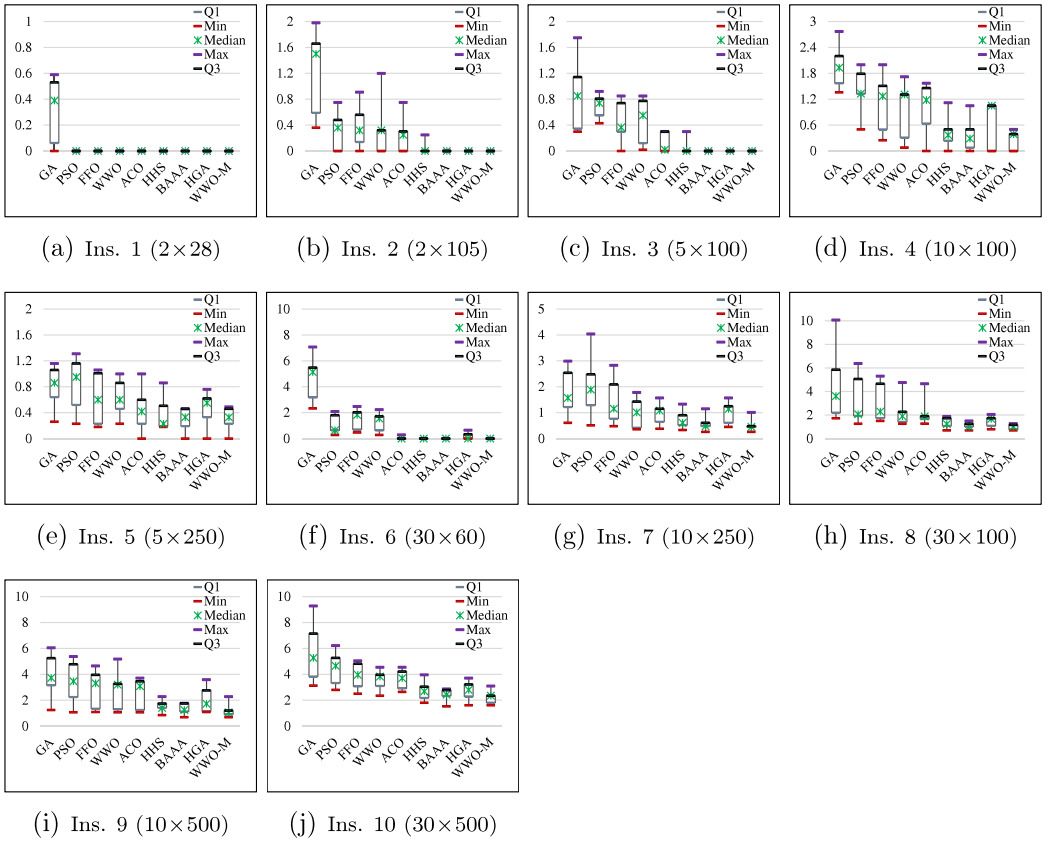


Fig. 5. Box plots of the results of the comparative algorithms on MKP test instances.

5. Application to a practical machine utilization optimization problem

5.1. Problem description

We apply WWO to solve a practical engineering optimization problem, which is to distribute a predefined workload W to a set of n available machines of the same type, such that the total expected maintenance time of the machines is minimized. The problem is critical to strategic planning of many large manufacturing enterprises.

Assume that the machine life follows an exponential distribution, the mean-time-between-maintenance (MTBM) can be estimated based on its failure rate function, denoted by F_r. That is, if the use time reaches the MTBM, the machine must be maintained to keep it in good working condition [81]. For each machine i, let t_i denote its total use time and $t_i(0)$ denote its use time since the last maintenance, then its expected use time from now to the next (1st round) maintenance is estimated as follows ($1 \le i \le n$):

$$t_i(1) = \frac{1}{F_r(t_i)} - t_i(0) \tag{19}$$

Table 4
The results of the comparative algorithms on MOMKP test instances.

Ins. (m)	Algorithms	$\mathcal{H}$		$\mathcal{D}$		$\mathcal{P}_N$	
		Mean	std	Mean	std	Mean	std
250	MOEA/D	†9.55E+07	(6.67E+06)	†3.79E−03	(4.28E−04)	67.23	(3.53)
	2PPLS	**9.73E+07**	(5.75E+06)	2.48E−02	(9.66E−04)	70.15	(2.97)
	MILS	9.58E+07	(4.29E+06)	†**1.00E−03**	(1.00E−04)	65.69	(6.48)
	Wϵ-CMOLS	9.65E+07	(3.69E+06)	2.28E−03	(5.06E−04)	68.93	(7.33)
	GwACO	9.60E+07	(6.25E+06)	†6.50E−03	(7.44E−04)	**72.60**	(9.90)
	MOWWO	9.58E+07	(6.72E+06)	†4.90E−03	(6.20E−04)	67.09	(8.22)
	MOWWO-M	9.63E+07	(5.08E+06)	2.62E−03	(2.28E−04)	69.81	(5.38)
500	MOEA/D	†3.90E+08	(2.27E+07)	8.51E−03	(1.32E−03)	†40.09	(3.81)
	2PPLS	**4.04E+08**	(1.30E+07)	†1.57E−02	(2.30E−03)	45.86	(2.36)
	MILS	3.95E+08	(1.60E+07)	†2.94E−03	(1.98E−04)	†40.99	(5.02)
	Wϵ-CMOLS	4.00E+08	(1.95E+07)	†**2.69E−03**	(3.30E−04)	46.43	(5.89)
	GwACO	3.99E+08	(9.84E+06)	8.99E−03	(2.05E−03)	**47.80**	(6.27)
	MOWWO	3.94E+08	(2.05E+07)	9.69E−03	(1.63E−03)	42.62	(5.35)
	MOWWO-M	3.99E+08	(1.37E+07)	7.21E−03	(1.12E−03)	45.00	(3.62)
750	MOEA/D	†8.56E+08	(6.49E+07)	†1.80E−02	(3.14E−03)	†6.91	(0.68)
	2PPLS	8.86E+08	(3.03E+07)	†4.56E−02	(7.22E−03)	12.12	(0.60)
	MILS	†8.59E+08	(5.11E+07)	†**5.31E−03**	(3.93E−04)	10.79	(1.98)
	Wϵ-CMOLS	**8.91E+08**	(5.85E+07)	7.33E−03	(1.09E−03)	11.03	(0.98)
	GwACO	8.66E+08	(2.79E+07)	†1.71E−02	(3.60E−03)	12.78	(1.51)
	MOWWO	†8.54E+08	(3.56E+07)	1.15E−02	(1.77E−03)	†9.66	(1.16)
	MOWWO-M	8.88E+08	(1.89E+07)	9.59E−03	(1.25E−03)	**13.03**	(1.26)

After the next maintenance, the failure rate of the machine becomes $F_r(t_i + t_i(1))$, and thus the expected use time from the next maintenance to the 2nd round maintenance is estimated as:

$$t_i(2) = \frac{1}{F_r(t_i + t_i(1))} \tag{20}$$

By analogy, the expected use time from the kth round maintenance to the $(k+1)$-th round maintenance is:

$$t_i(k+1) = \frac{1}{F_r(t_i + t_i(1) + \cdots + t_i(k))} \tag{21}$$

Let x_i be the workload (decision variable) allocated to machine i, then the number of maintenance of the machine during the planning cycle is:

$$\psi(x_i) = \max\{k \in \mathbb{Z}^+ | t_i(1) + t_i(2) + \cdots + t_i(k) \le x_i\} \tag{22}$$

Given the mean maintenance time τ, the accumulated maintenance time of machine i during the cycle is $T_i = \tau\psi(x_i)$. Let $T = \sum_{i=1}^{n} T_i$ be the total maintenance time of all the machines, our objective is to maximize the machine utilization in terms of $W/(W+T)$, which is identical to minimizing $\sum_{i=1}^{n} \psi(x_i)$. Thereby, the problem can be formulated as:

$$\min\ \Psi(\mathbf{x}) = \sum_{i=1}^{n} \psi(x_i) \tag{23}$$

$$\text{s.t.} \sum_{i=1}^{n} x_i = W \tag{24}$$

$$\text{Eqs. (19)–(22)}, \quad 1 \le i \le n$$

$$x_i \in \mathbb{Z}^+, \quad 1 \le i \le n$$

The above formulation is an integer program. The calculation of its objective function based on Eqs. (19)–(23) is of high complexity, and thus it is quite difficult to solve the problem when n and W are relatively large.

5.2. WWO for the machine utilization optimization problem

We are required to solve the considered problem for a large manufacturer in China, which now has thousands of machines and millions of working hours to be distributed in annual planning. The manufacturer had used a branch-and-bound (BnB) algorithm to solve the problem in the early 2000s. However, it became less and less efficient with increasing problem size. Therefore, the manufacturer had employed some metaheuristic algorithms including those based on PSO [82] and GA [83] for the problem. Recently, we use our approach to derive a discrete WWO algorithm for the problem based on the following steps:

(1) Use the basic n-dimensional integer vector representation.
(2) Define a neighborhood structure based on transferring a partial of workload from one machine to another.
(3) Use strategy *S2* to design the propagation operator as changing a solution $\mathbf{x}$ to a k-step neighbor, where k is proportional to $\lambda_{\mathbf{x}}$.
(4) Use strategy *S4* to calculate the wavelength for each solution $\mathbf{x}$ as

$$\lambda_{\mathbf{x}} = n^{\alpha(f_{\max} - f(\mathbf{x}) + \epsilon)/(f_{\max} - f_{\min} + \epsilon)} \tag{25}$$

where α linearly decreases from 0.68 to 0.32 with iteration.
(5) Use strategy *S6* to design the breaking operator.
(6) Use strategy *S7* for solution update.
(7) Use strategy *S10* to linearly reduce the population size *NP* from 60 to 12.

Algorithm 4 presents the WWO algorithm derived for the machine utilization optimization problem.

5.3. Application results

In each of the last three years (2016–2018), the manufacturer simultaneously run PSO, GA, and WWO to generate the annual machine utilization plans. Here, for comparison, we further include three state-of-the-art algorithms, including a hybrid BBO (HBBO) algorithm [84], an improved cuckoo search algorithm (CSA) [85], and a new boundary swarm optimization (BSO) algorithm [86]. Each algorithm is independently run 30 times, and the stopping condition is that maximum CPU time reaches four hours.

Table 5 presents the best, median and worst machine utilization rates obtained by each algorithm over the 30 runs, and Fig. 6 gives the distribution of the results. In Table 5, the maximum median and best utilization rates on each instance are shown in boldface, and a superscript † before a median value indicates that

Fig. 6. Box plots of the results of the algorithms for machine utilization optimization.

Algorithm 4: The WWO algorithm for the machine utilization optimization problem.

```
1  Randomly initialize a population of NP solutions to the problem;
2  Let x* be the best among the solutions;
3  while the stop condition is not met do
4      Calculate the wavelengths of the solutions according to Eq. (25);
5      foreach x in the population do
6          for k = 1 to rand(1, λ_x) do
7              Randomly select two distinct i, j ∈ [1, n];
8              Let c = rand(1, min(x_i, x_j));
9              Set x_i ← x_i − c, x_j ← x_j + c;
10         if the resulting x' is better than x then
11             Replace x with x' in the population;
12             if x is better than x* then
13                 for k = 1 to n_b do
14                     Create a neighbor according to Lines 7–9;
15                 Replace x* with the best among x and the neighbors;
16     Update the population size;
17 return x*.
```

the result of the algorithm is statistically significantly different from that of WWO. The results show that among the three algorithms, WWO always obtains the maximum best utilization rate in each year, and obtains the maximum median utilization rates in 2016 and 2017. BSO obtains the maximum median utilization rate in 2018. In terms of statistical tests, the results of WWO are significantly better than PSO and GA on all three instances, better than BSO in 2016 and 2017, and better than HBBO in 2017 and 2018. None of the other algorithms can obtain significantly better result than that of WWO on any instance. Therefore, we can conclude that WWO exhibits the best overall performance on the three instances.

Table 5
The resulting machine utilization rates (in percents) of the algorithms for solving the problem during 2016–2018..

Year	Metric	PSO	GA	HBBO	CSA	BSO	WWO
2016	Best	83.66	84.12	85.23	85.07	84.49	**85.68**
	Median	†79.91	†82.26	82.69	82.87	†82.48	**83.08**
	Worst	77.29	79.35	79.51	79.93	79.20	79.97
	Std	1.62	1.31	2.38	1.56	1.27	1.53
2017	Best	80.02	79.97	80.69	80.31	80.92	**81.10**
	Median	†76.71	†78.25	†78.46	78.77	†78.55	**79.33**
	Worst	73.50	74.13	75.10	75.31	75.27	74.08
	Std	2.37	2.20	2.59	1.58	1.97	2.86
2018	Best	74.66	75.51	76.25	76.22	76.07	**76.36**
	Median	†72.70	†74.22	†73.96	74.60	**74.91**	74.78
	Worst	71.05	72.80	71.86	71.33	73.15	73.63
	Std	1.82	1.33	2.22	1.79	1.09	1.52

During the three years, the manufacturers always select the best solution of WWO for practical implementation. Note that the increase of utilization rate will result in both the decrease of maintenance cost and the increase of machine use time. Thus, when the total workload and the number of machines are large, a slight increase of utilization rate will often bring considerable profits to the manufacturer. To evaluate this, we compare the best solutions obtained by the algorithms with a benchmark solution that divides all machines into several classes, roughly distributes the total workload to these classes, and for each class equally distributes the workload to the machines (which was the traditional manual method used by the manufacturer in its beginning stages). For each metaheuristic solution, we respectively calculate the decrease of maintenance cost and the increase of expected profits in comparison with the benchmark solution, and present the results in Fig. 7. Compared with PSO and GA solutions, using WWO solutions increases the profits of the manufacturer by approximately 13 and 10 million RMB yuan (2.2 and 1.7 million dollars) per year, respectively, which demonstrates the

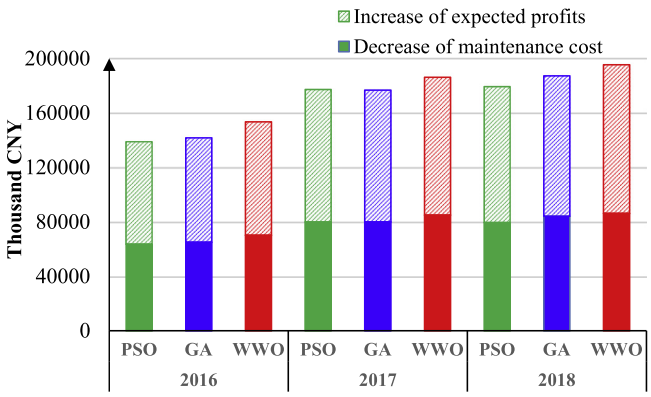


Fig. 7. The cost reduced and profits increased (in thousand RMB yuan) by the metaheuristic solutions compared to the benchmark solution.

significant economic benefits brought by our WWO solutions to the organization.

6. Conclusions

Adapting a generic metaheuristic to a special COP is a common but challenging task, because there lacks systematic strategies for guiding the process of algorithm refinement. This paper proposes a set of design strategies for adapting the WWO metaheuristic to different COPs. The tests on three typical COPs demonstrate that the proposed approach can derive concrete algorithms that are competitive to the state-of-the-art metaheuristics. The comparative results also demonstrate that using a self-adaptive breaking operator always exhibits much better performance than using a single fixed breaking operator. Moreover, we use the proposed approach to adapt WWO to a real-world engineering optimization problem, and the derived algorithm produces efficient solutions that increase the profits of the company by over 10 million RMB yuan (1.7 million dollars) compared to the old solutions.

We argue that, to promote the development of metaheuristics and facilitate their application in industry, the convenience of adapting a metaheuristic for different problems is as important as its tuned performance on benchmark problems. In particular, when proposing a new metaheuristic, the designer is expected to provide a deep insight into the role that each ingredient of the metaheuristic plays in optimization, and thus work out a set of guidelines and strategies on how to refine the general algorithmic framework to different concrete problems.

Currently we are applying our approach to more real-world engineering problems. A preliminary observation is that our approach can lead to not only higher development efficiency but also shorter training cycle. However, we need to conduct a more quantitative comparison of our systematic approach and traditional approaches for adapting different metaheuristics to the problems, and such quantitative results would be very useful to managers in deciding which metaheuristic and which design approach are the most appropriate for the problems to be solved. Our ongoing work also includes providing strategies for combining different metaheuristics to derive more efficient problem-solving algorithms.

Acknowledgment

This work was supported by National Natural Science Foundation of China (Grant No. 61473263, 61573316, and 61872123).

Declaration of competing interest

No author associated with this paper has disclosed any potential or pertinent conflicts which may be perceived to have impending conflict with this work. For full disclosure statements refer to https://doi.org/10.1016/j.asoc.2019.105611.

References

[1] A. Hertz, M. Widmer, Guidelines for the use of meta-heuristics in combinatorial optimization, European J. Oper. Res. 151 (2) (2003) 247–252, http://dx.doi.org/10.1016/S0377-2217(02)00823-8, meta-heuristics in combinatorial optimization.
[2] K. Sörensen, Metaheuristics – the metaphor exposed, Int. Trans. Oper. Res. 22 (1) (2015) 3–18, http://dx.doi.org/10.1111/itor.12001.
[3] K. Sörensen, F.W. Glover, Metaheuristics, Springer US, Boston, MA, 2013, pp. 960–970, http://dx.doi.org/10.1007/978-1-4419-1153-7_1167.
[4] J. Kennedy, R. Eberhart, Particle swarm optimization, in: IEEE International Conference on Neural Networks, Vol. 4, 1995, pp. 1942–1948, http://dx.doi.org/10.1109/ICNN.1995.488968.
[5] R. Storn, K. Price, Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces, J. Global Optim. 11 (4) (1997) 341–359, http://dx.doi.org/10.1023/A:1008202821328.
[6] D. Whitley, S. Rana, J. Dzubera, K.E. Mathias, Evaluating evolutionary algorithms, Artificial Intelligence 85 (1) (1996) 245–276, http://dx.doi.org/10.1016/0004-3702(95)00124-7.
[7] M. Dorigo, G. Di Caro, Ant colony optimization: a new meta-heuristic, in: IEEE Congress on Evolutionary Computation, Vol. 2, IEEE, 1999, pp. 1470–1477.
[8] D. Simon, Biogeography-based optimization, IEEE Trans. Evol. Comput. 12 (6) (2008) 702–713, http://dx.doi.org/10.1109/TEVC.2008.919004.
[9] K. Socha, M. Dorigo, Ant colony optimization for continuous domains, European J. Oper. Res. 185 (3) (2008) 1155–1173, http://dx.doi.org/10.1016/j.ejor.2006.06.046.
[10] H. Ma, D. Simon, Blended biogeography-based optimization for constrained optimization, Eng. Appl. Artif. Intell. 24 (3) (2011) 517–525, http://dx.doi.org/10.1016/j.engappai.2010.08.005.
[11] E. Rashedi, H. Nezamabadi-pour, S. Saryazdi, GSA: A gravitational search algorithm, Inform. Sci. 179 (13) (2009) 2232–2248, http://dx.doi.org/10.1016/j.ins.2009.03.004.
[12] Y. Tan, Y. Zhu, Fireworks algorithm for optimization, in: Y. Tan, Y. Shi, K. Tan (Eds.), Advances in Swarm Intelligence, in: Lect. Note. Comput. Sci., vol. 6145, Springer, 2010, pp. 355–364, http://dx.doi.org/10.1007/978-3-642-13495-1_44.
[13] X.-S. Yang, A.H. Gandomi, Bat algorithm: A novel approach for global engineering optimization, Eng. Comput. 29 (5) (2012) 464–483, http://dx.doi.org/10.1108/02644401211235834.
[14] K.N. Das, T.K. Singh, Drosophila food-search optimization, Appl. Math. Comput. 231 (2014) 566–580, http://dx.doi.org/10.1016/j.amc.2014.01.040.
[15] M. Yazdani, F. Jolai, Lion optimization algorithm (LOA): A nature-inspired metaheuristic algorithm, J. Comput. Des. Eng. 3 (1) (2016) 24–36, http://dx.doi.org/10.1016/j.jcde.2015.06.003.
[16] J.J. Liang, B.Y. Qu, P.N. Suganthan, Q. Chen, Problem definitions and evaluation criteria for the CEC 2015 competition on learning-based real-parameter single objective optimization, Tech. rep. Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou, China (2014).
[17] T. Alexandros, D. Georgios, Nature inspired optimization algorithms related to physical phenomena and laws of science: A survey, Int. J. Artif. Intell. Tools 26 (6) (2017) 1750022, http://dx.doi.org/10.1142/S0218213017500221.
[18] G. Jaradat, M. Ayob, I. Almarashdeh, The effect of elite pool in hybrid population-based meta-heuristics for solving combinatorial optimization problems, Appl. Soft Comput. 44 (2016) 45–56, http://dx.doi.org/10.1016/j.asoc.2016.01.002.
[19] M.B. Dowlatshahi, H. Nezamabadi-pour, M. Mashinchi, A discrete gravitational search algorithm for solving combinatorial optimization problems, Inform. Sci. 258 (2014) 94–107, http://dx.doi.org/10.1016/j.ins.2013.09.034.
[20] Y.-J. Zheng, Water wave optimization: A new nature-inspired metaheuristic, Comput. Oper. Res. 55 (1) (2015) 1–11, http://dx.doi.org/10.1016/j.cor.2014.10.008.
[21] X.-B. Wu, J. Liao, Z.-C. Wang, Water wave optimization for the traveling salesman problem, in: D.-S. Huang, V. Bevilacqua, P. Premaratne (Eds.), Intelligent Computing Theories and Methodologies, Springer, Cham, 2015, pp. 137–146, http://dx.doi.org/10.1007/978-3-319-22180-9_14.
[22] K. Lenin, B.R. Reddy, M. Suryakalavathi, Hybridization of firefly and water wave algorithm for solving reactive power problem, Int. J. Eng. Res. Afr. 21 (2016) 165–171.

[23] X. Wu, Y. Zhou, Y. Lu, Elite opposition-based water wave optimization algorithm for global optimization, Math. Probl. Eng. 2017 (2017) 25, http://dx.doi.org/10.1155/2017/3498363.

[24] X. Yun, X. Feng, X. Lyu, S. Wang, B. Liu, A novel water wave optimization based memetic algorithm for flow-shop scheduling, in: IEEE Congress on Evolutionary Computation, 2016, pp. 1971–1976, http://dx.doi.org/10.1109/CEC.2016.7744029.

[25] A. Azadi Hematabadi, A. Akbari Foroud, Optimizing the multi-objective bidding strategy using min–max technique and modified water wave optimization method, Neural Compu. Appl. (2018) http://dx.doi.org/10.1007/s00521-018-3361-0, (online first).

[26] J. Zhang, Y. Zhou, Q. Luo, An improved sine cosine water wave optimization algorithm for global optimization, J. Intell. Fuzzy Syst. 34 (4) (2018) 2129–2141, http://dx.doi.org/10.3233/JIFS-171001.

[27] F. Zhao, H. Liu, Y. Zhang, W. Ma, C. Zhang, A discrete water wave optimization algorithm for no-wait flow shop scheduling problem, Expert Syst. Appl. 91 (2018) 347–363, http://dx.doi.org/10.1016/j.eswa.2017.09.028.

[28] F. Zhao, L. Zhang, H. Liu, Y. Zhang, W. Ma, C. Zhang, H. Song, An improved water wave optimization algorithm with the single wave mechanism for the no-wait flow-shop scheduling problem, Eng. Optim. (2018) http://dx.doi.org/10.1080/0305215X.2018.1542693, (online first).

[29] Z. Shao, D. Pi, W. Shao, A novel discrete water wave optimization algorithm for blocking flow-shop scheduling problem with sequence-dependent setup times, Swarm Evol. Comput. 40 (1) (2018) 53–75, http://dx.doi.org/10.1016/j.swevo.2017.12.005.

[30] Z. Shao, D. Pi, W. Shao, A novel multi-objective discrete water wave optimization for solving multi-objective blocking flow-shop scheduling problem, Knowl.-Based Syst. 165 (1) (2019) 110–131, http://dx.doi.org/10.1016/j.knosys.2018.11.021.

[31] X.-H. Zhou, Z.-G. Xu, M.-X. Zhang, Y.-J. Zheng, Water wave optimization for articial neural network parameter and structure optimization, in: International Conference on Bio-Inspired Computing: Theories and Applications, in: Commun Comput Inf Sci, vol. 951, Springer, 2018, pp. 343–354, http://dx.doi.org/10.1007/978-981-13-2826-8_30.

[32] A. Liu, P. Li, W. Sun, X. Deng, W. Li, Y. Zhao, B. Liu, Prediction of mechanical properties of micro-alloyed steels via neural networks learned by water wave optimization, Neural Comput. Appl. (2019) http://dx.doi.org/10.1007/s00521-019-04149-1, (online first).

[33] G. Singh, M. Rattan, S.S. Gill, N. Mittal, Hybridization of water wave optimization and sequential quadratic programming for cognitive radio system, Soft Comput. (2018) http://dx.doi.org/10.1007/s00500-018-3437-x, (online first).

[34] J. Zhang, Y. Zhou, Q. Luo, Nature-inspired approach: a wind-driven water wave optimization algorithm, Appl. Intell. 49 (1) (2019) 233–252, http://dx.doi.org/10.1007/s10489-018-1265-4.

[35] Y.-J. Zheng, B. Zhang, A simplified water wave optimization algorithm, in: IEEE Congress on Evolutionary Computation, 2015, pp. 807–813, http://dx.doi.org/10.1109/CEC.2015.7256974.

[36] N. Mladenović, P. Hansen, Variable neighborhood search, Comput. Oper. Res. 24 (11) (1997) 1097–1100, http://dx.doi.org/10.1016/S0305-0548(97)00031-2.

[37] W. Gao, Z. Wei, Y. Luo, J. Cao, Artificial bee colony algorithm based on Parzen window method, Appl. Soft Comput. 74 (2019) 679–692, http://dx.doi.org/10.1016/j.asoc.2018.10.024.

[38] P. Moscato, C. Cotta, Memetic Algorithms, Oxford University Press, 2002.

[39] B. Zhang, M.-X. Zhang, J.-F. Zhang, Y.-J. Zheng, A water wave optimization algorithm with variable population size and comprehensive learning, in: D.-S. Huang, V. Bevilacqua, P. Premaratne (Eds.), Intelligent Computing Theories and Methodologies, Springer, Cham, 2015, pp. 124–136.

[40] Z.-Y. Rong, M.-X. Zhang, Y.-C. Du, Y.-J. Zheng, A hybrid evolutionary algorithm for combined road-rail emergency transportation planning, in: Y. Tan, Y. Shi, Q. Tang (Eds.), Advances in Swarm Intelligence, Springer, Cham, 2018, pp. 465–476, http://dx.doi.org/10.1007/978-3-319-93815-8_44.

[41] Y.-J. Zheng, Y. Wang, H.-F. Ling, Y. Xue, S.-Y. Chen, Integrated civilian-military pre-positioning of emergency supplies: A multiobjective optimization approach, Appl. Soft Comput. 58 (2017) 732–741, http://dx.doi.org/10.1016/j.asoc.2017.05.016.

[42] W.L. Wang, C. Chen, L. Li, W.K. Li, Adaptive water wave optimization algorithm based on simulated annealing, Comput. Sci. 44 (10) (2017) 216–227, (in Chinese).

[43] M. Kilany, A.E. Hassanien, A. Badr, Accelerometer-based human activity classification using water wave optimization approach, in: 11th International Computer Engineering Conference, 2015, pp. 175–180, http://dx.doi.org/10.1109/ICENCO.2015.7416344.

[44] W. Xu, Z. Ye, Y. Hou, A fast image match method based on water wave optimization and gray relational analysis, in: 9th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems, Vol. 2, 2017, pp. 771–776, http://dx.doi.org/10.1109/IDAACS.2017.8095193.

[45] H. Chen, Y. Hou, Q. Luo, Z. Hu, L. Yan, Text feature selection based on water wave optimization algorithm, in: 10th International Conference on Advanced Computational Intelligence, 2018, pp. 546–551, http://dx.doi.org/10.1109/ICACI.2018.8377518.

[46] Z. Ren, C. Wang, X. Fan, Z. Ye, Fuzzy Clustering based on water wave optimization, in: 13th International Conference on Computer Science Education, 2018, pp. 1–5, http://dx.doi.org/10.1109/ICCSE.2018.8468778.

[47] M. Siva, R. Balamurugan, L. Lakshminarasimman, Water wave optimization algorithm for solving economic dispatch problems with generator constraints, Int. J. Intell. Eng. Syst. 9 (4) (2016) 31–40.

[48] Y. Zhou, J. Zhang, X. Yang, Y. Ling, Optimal reactive power dispatch using water wave optimization algorithm, Oper. Res.(online first 2018). http://dx.doi.org/10.1007/s12351-018-0420-3.

[49] S. Zhao, Z. Li, X. Yun, K. Wang, X. Lyu, B. Liu, IIR filters designing by water wave optimization, in: 13th IEEE International Conference on Control Automation, 2017, pp. 347–352, http://dx.doi.org/10.1109/ICCA.2017.8003085.

[50] A.M.F. Fard, M. Hajaghaei-Keshteli, A tri-level location–allocation model for forward/reverse supply chain, Appl. Soft Comput. 62 (2018) 328–346, http://dx.doi.org/10.1016/j.asoc.2017.11.004.

[51] A.M.F. Fard, M. Hajiaghaei-Keshteli, A bi-objective partial interdiction problem considering different defensive systems with capacity expansion of facilities under imminent attacks, Appl. Soft Comput. 68 (2018) 343–359, http://dx.doi.org/10.1016/j.asoc.2018.04.011.

[52] Q. Song, Y. Zheng, Y. Huang, Z. Xu, W. Sheng, J. Yang, Emergency drug procurement planning based on big-data driven morbidity prediction, IEEE Trans. Ind. Informat.(online first) http://dx.doi.org/10.1016/j.ecolind.2017.06.037.

[53] Y. Zheng, B. Zhang, J. Xue, Selection of key software components for formal development using water wave optimization, J. Softw. 27 (4) (2016) 933–942, http://dx.doi.org/10.13328/j.cnki.jos.004964, (in Chinese).

[54] M. Nawaz, E.E. Enscore, I. Ham, A heuristic algorithm for the *m*-machine, *n*-job flow-shop sequencing problem, Omega 11 (1) (1983) 91–95, http://dx.doi.org/10.1016/0305-0483(83)90088-9.

[55] F. Glover, Multilevel tabu search and embedded search neighborhoods for the traveling salesman problem, Working paper, University of Colorado (1991).

[56] Y. Wang, M.-X. Zhang, Y.-J. Zheng, A hyper-heuristic method for UAV search planning, in: Y. Tan, H. Takagi, Y. Shi, B. Niu (Eds.), Advances in Swarm Intelligence, Part II, Springer, 2017, pp. 454–464, http://dx.doi.org/10.1007/978-3-319-61833-3_48.

[57] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Trans. Evol. Comput. 6 (2) (2002) 182–197, http://dx.doi.org/10.1109/4235.996017.

[58] E. Taillard, Benchmarks for basic scheduling problems, European J. Oper. Res. 64 (2) (1993) 278–285, http://dx.doi.org/10.1016/0377-2217(93)90182-M.

[59] L. Wang, L. Zhang, D.-Z. Zheng, A class of order-based genetic algorithm for flow shop scheduling, Int. J. Adv. Manuf. Technol. 22 (11) (2003) 828–835, http://dx.doi.org/10.1007/s00170-003-1689-8.

[60] C.-J. Liao, C.-T. Tseng, P. Luarn, A discrete version of particle swarm optimization for flowshop scheduling problems, Comput. Oper. Res. 34 (10) (2007) 3099–3111, http://dx.doi.org/10.1016/j.cor.2005.11.017.

[61] Q.-K. Pan, M.F. Tasgetiren, Y.-C. Liang, A discrete differential evolution algorithm for the permutation flowshop scheduling problem, Comput. Ind. Eng. 55 (4) (2008) 795–816, http://dx.doi.org/10.1016/j.cie.2008.03.003.

[62] P. Dasgupta, S. Das, A discrete inter-species cuckoo search for flowshop scheduling problems, Comput. Oper. Res. 60 (2015) 111–120, http://dx.doi.org/10.1016/j.cor.2015.01.005.

[63] W. Shao, D. Pi, Z. Shao, An extended teaching-learning based optimization algorithm for solving no-wait flow shop scheduling problem, Appl. Soft Comput. 61 (2017) 193–210, http://dx.doi.org/10.1016/j.asoc.2017.08.020.

[64] Y.-J. Zheng, H.-F. Ling, H.-H. Shi, H.-S. Chen, S.-Y. Chen, Emergency railway wagon scheduling by hybrid biogeography-based optimization, Comput. Oper. Res. 43 (3) (2014) 1–8, http://dx.doi.org/10.1016/j.cor.2013.09.002.

[65] Y.-C. Du, M.-X. Zhang, C.-Y. Cai, Y.-J. Zheng, Enhanced biogeography-based optimization for flow-shop scheduling, in: J. Qiao, X. Zhao, L. Pan, X. Zuo, X. Zhang, Q. Zhang, S. Huang (Eds.), Bio-Inspired Computing: Theories and Applications, Commun Comput Inf Sci, Springer, Singapore, 2018, pp. 295–306.

[66] O. Engin, A. Güçlü, A new hybrid ant colony optimization algorithm for solving the no-wait flow shop scheduling problems, Appl. Soft Comput. 72 (2018) 166–176, http://dx.doi.org/10.1016/j.asoc.2018.08.002.

[67] J.E. Beasley, Or-library: Distributing test problems by electronic mail, J. Oper. Res. Soc. 41 (11) (1990) 1069–1072, http://dx.doi.org/10.1057/jors.1990.166.

[68] P. Chu, J. Beasley, A genetic algorithm for the multidimensional knapsack problem, J. Heuristics 4 (1) (1998) 63–86, http://dx.doi.org/10.1023/A:1009642405419.

[69] J.C. Bansal, K. Deep, A modified binary particle swarm optimization for knapsack problems, Appl. Math. Comput. 218 (22) (2012) 11042–11061, http://dx.doi.org/10.1016/j.amc.2012.05.001.
[70] L. Wang, X.-L. Zheng, S.-Y. Wang, A novel binary fruit fly optimization algorithm for solving the multidimensional knapsack problem, Knowl.-Based Syst. 48 (2013) 17–23, http://dx.doi.org/10.1016/j.knosys.2013.04.003.
[71] L. Ke, Z. Feng, Z. Ren, X. Wei, An ant colony optimization approach for the multidimensional knapsack problem, J. Heuristics 16 (1) (2010) 65–83, http://dx.doi.org/10.1007/s10732-008-9087-x.
[72] B. Zhang, Q.-K. Pan, X.-L. Zhang, P.-Y. Duan, An effective hybrid harmony search-based algorithm for solving multidimensional knapsack problems, Appl. Soft Comput. 29 (2015) 288–297, http://dx.doi.org/10.1016/j.asoc.2015.01.022.
[73] X. Zhang, C. Wu, J. Li, X. Wang, Z. Yang, J.-M. Lee, K.-H. Jung, Binary artificial algae algorithm for multidimensional knapsack problems, Appl. Soft Comput. 43 (2016) 583–595, http://dx.doi.org/10.1016/j.asoc.2016.02.027.
[74] A. Rezoug, M. Bader-El-Den, D. Boughaci, Hybrid Genetic Algorithms To Solve the Multidimensional Knapsack Problem, Springer, Cham, 2019, pp. 235–250, http://dx.doi.org/10.1007/978-3-319-95104-1_15.
[75] T. Lust, J. Teghem, The multiobjective multidimensional knapsack problem: a survey and a new approach, Int. Trans. Oper. Res. 19 (4) (2012) 495–520, http://dx.doi.org/10.1111/j.1475-3995.2011.00840.x.
[76] E. Zitzler, L. Thiele, Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach, IEEE Trans. Evol. Comput. 3 (4) (1999) 257–271, http://dx.doi.org/10.1109/4235.797969.
[77] Q. Zhang, H. Li, MOEA/D: A multiobjective evolutionary algorithm based on decomposition, IEEE Trans. Evol. Comput. 11 (6) (2007) 712–731, http://dx.doi.org/10.1109/TEVC.2007.892759.
[78] D.S. Vianna, M. d. F. A. D. Vianna, Local search-based heuristics for the multiobjective multidimensional knapsack problem, Production 23 (3) (2013) 478–487, http://dx.doi.org/10.1590/S0103-65132012005000081.
[79] I.B. Mansour, M. Basseur, F. Saubion, A multi-population algorithm for multi-objective knapsack problem, Appl. Soft Comput. 70 (2018) 814–825, http://dx.doi.org/10.1016/j.asoc.2018.06.024.
[80] I. Ben Mansour, I. Alaya, M. Tagina, A gradual weight-based ant colony approach for solving the multiobjective multidimensional knapsack problem, Evol. Intell. (2019) http://dx.doi.org/10.1007/s12065-019-00222-9, (online first).
[81] J. Moubray, Reliability-Centered Maintenance, Industrial Press, 1997.
[82] P.-Y. Yin, J.-Y. Wang, A particle swarm optimization approach to the nonlinear resource allocation problem, Appl. Math. Comput. 183 (1) (2006) 232–242, http://dx.doi.org/10.1016/j.amc.2006.05.051.
[83] H.-F. Ling, Y.-J. Zheng, Y.-H. Xiao, Intelligent Optimization Decision Methods and Applications for Material Support, National Defense Press, Beijing, 2015.
[84] Z.-C. Wang, X.-B. Wu, Hybrid biogeography-based optimization for integer programming, Sci. World J. 2014 (2014) 9, http://dx.doi.org/10.1155/2014/672983.
[85] M. Abdel-Baset, Y. Zhou, M. Ismail, An improved cuckoo search algorithm for integer programming problems, Int. J. Comput. Sci. Math. 9 (1) (2018) 66–81, http://dx.doi.org/10.1504/IJCSM.2018.090710.
[86] W.-C. Yeh, A novel boundary swarm optimization method for reliability redundancy allocation problems, Reliab. Eng. Syst. Safety (2018) http://dx.doi.org/10.1016/j.ress.2018.02.002, (online first).