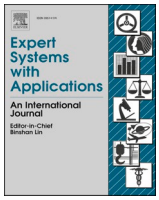
Expert Systems With Applications 244 (2024) 122931

Contents lists available at ScienceDirect

Expert Systems With Applications

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



World Hyper-Heuristic: A novel reinforcement learning approach for dynamic exploration and exploitation

Arman Daliri [a,*], Mahmoud Alimoradi [b], Mahdieh Zabihimayvan [c], Reza Sadeghi [d]

[a] Department of Computer Engineering, Karaj Branch, Islamic Azad University, Karaj, Iran
[b] Department of Computer Engineering, Lahijan Branch, Islamic Azad University, Lahijan, Iran
[c] Department of Computer Science, Central Connecticut State University, New Britain, CT, USA
[d] School of Computer Science and Mathematics, Marist College, Poughkeepsie, NY, USA

ARTICLE INFO

Keywords:
Hyper-heuristic
NP-Hard
Exploration
Exploitation
Reinforcement Learning

ABSTRACT

In the real world, there are many complex problems in engineering. Every problem has a level of computational complexity, starting from simple problems and reaching NP-hard problems. Np-hard problems do not have a definite answer. Therefore, hyper-heuristic algorithms try to optimize NP-hard problems. Hyper-heuristics optimize complex search problems using a combination plate of exploration and exploitation strategies. The current algorithms need more generalizability, handling specific data types, limitations to a particular search problem, and weak performance. We propose a World hyper-heuristic (World) to address the issues using a novel reinforcement learning method. World, in two steps of rewarding and selection, dynamically switches between exploration and exploitation strategies provided in an infinite pool of *meta*-heuristics. We evaluated the performance of our proposed method in three phases. First, we optimized the standard functions as artificial Np-hard problems. Then, we compared real engineering examples related to discrete Np-hard problems. Finally, we implemented and analyzed problems with the continuous Np-hard problems. Our extensive comparisons with the state-of-the-art algorithms demonstrate the World's outperformance in handling varied search problems with any data type. Among all the findings, World finds the shortest path of length 4.33e + 05, far shorter than the results of the state-of-the-art work, in benchmarked data of 10,000 real cities.

1. Introduction

Humans are engaged in solving a wide range of problems with different computational complexities. The problems, like sorting a list of numbers (Katajainen and Träff, 1997), are solvable using polynomial-time algorithms (Blondel and Tsitsiklis, 2000), which means the maximum amount of time required to solve is in a proportion to $C \times n^k$, where n is the number of inputs, and C and k are constant numbers. We sometimes need to solve problems that require a non-polynomial solution time. For instance, the K-means algorithm (MacQueen and others 1967) has a solution time in proportion to (input numbers)^2 in the worst case scenario. K-means take a place in a group of algorithms, which computer scientists could not find any polynomial solutions. They could not even decide upon the existence or non-existence of a polynomial solution. Such problems are known as non-deterministic polynomial-time hard (NP-hard) problems (Blondel and Tsitsiklis, 2000).

Real-world problems involve different computational complexity. The computational complexity of problems lets researchers know which methods or algorithms are suitable for which problems. Computational complexity, including P, NP, NP-complete, and NP-hard, are categorized from easy with a definite answer to hard without a definite answer, respectively (Ge et al., 2023). The class of NP-hard problems consumes a high amount of processing time as they need exhaustive search on the problem space (Shukla et al., 2019; Neumann and Witt, 2010). Meta-heuristics are remedies to solve NP-hard problems by finding a near optimal solution in a reasonable computational time. These strategies begin from a probable, randomly selected solution and use explorations and exploitations on the search space to find the near optimal solution. Exploration evaluates candidate solutions that are not neighbors to the current solution(s) while exploitation searches the neighborhood of the current solution(s) with the aim of finding the optimal solution.

The earliest and best-known *meta*-heuristic, known as the genetic algorithm, was inspired by Charles Darwin's theory of biological evolution (Srinivas and Patnaik, 1994). Following this, various *meta*-

* Corresponding author.
E-mail address: arman.daliri@kiau.ac.ir (A. Daliri).

https://doi.org/10.1016/j.eswa.2023.122931
Received 27 August 2023; Received in revised form 19 November 2023; Accepted 10 December 2023
Available online 21 December 2023

heuristics were introduced to imitate physical events, e.g., simulated annealing (Kirkpatrick, Gelatt, and Vecchi 1983); human behaviors, e.g., Tabu search (Glover, 1989); and group movements, e.g., particular swarm (Kennedy and Eberhart, 1995). Each of these *meta*-heuristics can optimize a particular type of problems according to the theory of no-free lunch (NFL) (Wolpert and Macready, 1997). A combination plate of *meta*-heuristics, which are known as hyper-heuristics (Burke et al., 2019), can provide better results by moving the search space from finding solutions to selecting a proper solution.

Some of these algorithms are designed to solve a specific search problem. The algorithms provide a near optimal problem-specific solution by combining some *meta*-heuristics and heuristics. For example, modified subpopulation teaching–learning-based optimization (MS-TLBO) reinforces TLBO to optimize truss topology optimization (TTO) (Savsani, Tejani, and Patel 2016a). On the other hand, some algorithms provide complex solutions for multi-objective optimization problems (Tejani, Pholdee, et al. 2019a; Tejani et al., 2018; Tejani, Savsani, et al. 2019b). For instance, structural optimization using multi-objective modified adaptive symbiotic organisms search (MOMASOS) can solve the TTO search problem. However, some *meta*-heuristics are slightly different. These algorithms use a source and only an idea for optimization. One of the recent *meta*-heuristics that is nature-base is the Trees Social Relations Optimization algorithm (TSR) (Alimoradi,Azgomi,and Asghari, 2022), which is inspired by the hierarchical life of trees in a jungle. In (Savsani, Tejani, and Patel 2016b), the authors propose a Truss topology optimization algorithm with static and dynamic constraints, which works based on a modified subpopulation teaching–learning-based optimization. The difference between this algorithm and the other hyper-heuristics is the source of inspiration and the exploration and exploitation methods for searching the optimal solution. In other words, a hyper-heuristic can combine multi-search methods, but a *meta*-heuristic only searches using one method.

The recent research in (Kumar et al., 2022) presented modified versions of five state-of-the-art nature-inspired algorithms, with the aim of optimizing truss design problems with static and dynamic constraints. To investigate the efficacy and feasibility of the suggested modifications, three traditional benchmark problems of truss optimization were utilized. In (Kunakote et al., 2022), the performance of twelve meta-heuristics is compared and analyzed in wind farm layout optimization problem. The experiments are based on four wind farm layout optimization problems as benchmarks, and the main design problem is to minimize wind farm cost and maximize wind farm totally produced power. In another work (Kumar et al., 2021), the researchers propose an optimizer to solve a multi-objective engineering design problem. The design problem is focused on weight minimization and maximization of nodal deflection subject to multiple constraints of trusses.

Hyper-heuristics can be as simple as running a sequence of heuristic and *meta*-heuristics (Thompson and Welker, 1963) or as complex as using reinforcement learnings (Qin et al., 2021) to select the proper solution in each step. Although these models can provide optimal solutions for some search problems, they still rely on the input data types and leverage a limited number of heuristic and *meta*-heuristics. The former issue points to the fact that some hyper-heuristics are usable for problems with a discrete search area (Walker and Keedwell, 2016) while the others can handle the search space of continuous data (Koulinas, Kotsikas,and Anagnostopoulos, 2014). The latter issue suggests that the current hyper-heuristics can use a limited number of *meta*-heuristics. For instance, (Maashi,Özcan,and Kendall, 2014) uses a reinforcement learning method to leverage only three *meta*-heuristics. As predicted by NFL theory, this limitation leads to either an equal or a weaker performance of some hyper-heuristic (Bai,Burke,and Kendall, 2008; Thomas and Chaudhari, 2014) or heuristic and *meta*-heuristics in some problems. It is the reason that most hyper-heuristics such as (Qin et al., 2021; Zhang,Zhao,and Leng, 2020; Lin,Zhu,and Gao, 2020) are limited to a particular search problem.

To fill the gaps mentioned above, this work proposes a World algorithm. World solves the issue of relying on the input's data type by dealing with both discrete and continuous search problems. World also employs a reinforcement learning (RL) method to optimally take advantage of an infinite number of *meta*-heuristics in solving NP-hard problems. RL considers more algorithm power according to the history of search strategies in an iterative process while it reserves the chance for weak algorithms in all iterations. Once World begins, all the original *meta*-heuristics have a similar chance to optimize the search problem. Then, the World method gradually drives search processes toward the best exploration and exploitation with the aim of achieving the near optimal solution. Based on the information mentioned, the main contributions and highlights of World algorithm are:

- World algorithm, utilizing a unique reinforcement learning method in two steps of rewarding and selection, dynamically switches between exploration and exploitation techniques provided in a pool containing several *meta*-heuristics algorithms.
- The proposed algorithm can optimize NP-hard problems with continuous and discrete domains and has provided competitive answers.
- The proposed algorithm, with dynamic search, avoids greedy optimization to optimize different problems and examines all modes as much as possible.
- The proposed algorithm outperforms the well-known search strategies in over 30 artificial, real, discrete, and continuous optimization problems.
- World finds the shortest path of length 4.33e + 05, which is far shorter than the results of the state-of-the-art work, in a benchmarked data of 10,000 real cities.

Trusting the described problems and examining the issue, it will be found that hyper-heuristic algorithms to cover comprehensive problems in *meta*-heuristic algorithms are an essential and fundamental need for industry and the real world. Furthermore, in the research conducted to solve this issue by providing a solution from the category of artificial intelligence for artificial intelligence, an attempt was made to interpret the work that is important for today's world and the science of artificial intelligence. Also, to evaluate the efficiency of the proposed algorithm, using real and synthetic problems, the performance of the World algorithm and other algorithms has been compared. Among the set of problems, real data has been used for the TSP problem, which shows the real efficiency of the World algorithm in the industry. Finally, the results of the world algorithm have been compared with many metaheuristic algorithms. This work evaluates the proposed algorithm's performance and proves its proper performance compared to other competitors.

The rest of this paper is organized as follows: Section 2 discusses related work on *meta*-heuristics to solve NP-hard problems. Section 3 describes the proposed algorithm and its implementation. Section 4 provides an evaluation on the proposed algorithm and compares it with other *meta*-heuristics and hyper-heuristics. Finally, Section 5 summarizes the main conclusions and discusses future work.

2. Related works

In this section, we briefly describe and compare some of the *meta*-heuristics and hyper-heuristics for optimizing NP-hard problems. We first discuss some well-known *meta*-heuristics and their applications in Subsection 2.1. Following this, we describe how common hyper-heuristics can take the advantage of *meta*-heuristics in Subsection 2.2.

2.1. Meta-heuristics

Meta-heuristics are inspired from the nature and natural rules of the world. Cognitive science, behavioral science, biology, physics, and society are some examples of the inspiration resources for the *meta*-heuristics. Such algorithms can deal with either discrete or continuous

search problems in an iterative process using both exploration and exploitation. These two strategies help finding diverse optimal solutions, avoid restraining solution found in the first iterations, and give chance to weak solutions. There is a wide variety of *meta*-heuristics to address varied NP-hard problems and their conditions (Wolpert and Macready, 1997). The *meta*-heuristics can be categorized into four classes of evolutionary-based, physics-based, human-based, and swarm-based algorithms (Mirjalili and Lewis, 2016).

Evolutionary-based algorithms are inspired from the biological evolution theory of Charles Darwin (C. A. C. Coello 1999). Such algorithms are based on the process of natural selection where the fittest individuals are chosen as parents to produce offspring of the next generation. Genetic algorithm is one of the first evolutionary-based algorithms that relies on the biologically inspired operators of mutation, crossover, and selection (Srinivas and Patnaik, 1994). The main problem of this algorithm is its high computation complexity caused by the exponential increase in search space size (Alimoradi,Azgomi,and Asghari, 2022).To solve the complexity issue, other algorithms such as invasive weed optimization algorithm (IWO) (Mehrabian and Lucas, 2006) have been proposed. IWO is inspired by the behavior of weed colonies, which aim to find the best life environment and quickly adapt to new conditions.

Algorithms in the class of physics-based algorithms rely on physical and mathematical rules to optimize NP hard search problems. One of the most well-known algorithms of this class is simulated annealing (SA) (Kirkpatrick, Gelatt, and Vecchi 1983), which uses hill climbing heuristic and is inspired by the process of cooling and annealing in metals. SA starts the search with one candidate solution and improves it over iterations to avoid greedy search. Since- Cosine algorithm (SCA) (Mirjalili, 2016) is another algorithm in the class, which uses the application of sine and cosine functions to find the optimal solution. The water optimization algorithm (WAO)(Daliri et al., 2022) is the in the class of physics-based algorithms.

Human-based algorithms are inspired by human social evolution to optimize the search problems. Imperialist competitive algorithm (ICA) (Atashpaz-Gargari and Lucas, 2007) is one of the well-known algorithms in this class and starts with an initial population, which is a set of valid solutions. Population individuals are considered as countries in two types of colonies and imperialist. The combination of counties shape empires. The algorithm aims to find the most powerful empire, which take possession of the weak colonies. Tabu search (Glover, 1989) proposed in 1986 is also placed in this class. Tabu search employs local search methods to start from a potential solution and check its immediate neighbors with the aim of finding an improved solution. Through this process, Tabu algorithm tries to jump outside of local minima by considering visited solutions as tabu and allowing worsening solutions in an iteration if there is no improving solution.

Swarm-based algorithms rely on collective behaviors of decentralized, self-organized systems to optimize a search problem. In such systems, there are simple rules followed by the individual agents and there is no centralized structure to control their behaviors. Indeed, the interactions between the agents lead to an intelligent optimal behavior in the system. Ant colony optimization (ACO) (Dorigo,Birattari,and Stutzle, 2006) is one of the well-known algorithms in this class. ACO is inspired by the behavior of ants in nature and uses a pheromone model to find the best path on a weighted graph. To reduce the high time complexity of ACO (Gutjahr, 2007), we can use Particle swarm optimization (PSO) (Kennedy and Eberhart, 1995) algorithm, which has lower time complexity compared to ACO. PSO is based on the movements of organisms in a bird flock or fish school in finding foods. (Trivedi,Varshney,and Ramteke, 2020) provides a simplified multi-objective PSO to reduces perturbation components into guided move and random move.

There are some *meta*-heuristics that are proposed in the CEC test conger. These algorithms are designed to optimize real parameter optimization problems. The Single objective real-parameter optimization: Algorithm jSO, is one of those algorithms that was published in 2017 (Brest, Maučec, and Bošković 2017). jSO were performed for CEC 2017 test. Every year some algorithms are presented with this evaluation test. In CEC 2021 test, one of the newest algorithms is the Self-adaptive Differential Evolution Algorithm with Population Size Reduction for Single Objective Bound-Constrained Optimization: Algorithm j21. This algorithm uses several mechanisms for searching, building vectors and creating population size (Brest, Maučec, and Bošković 2021, 21). Also, there are many algorithms that are not using CEC test and those are new too. For example, in 2016 the Adaptive differential evolution algorithm with novel mutation strategies in multiple sub-populations was presented. This algorithm generally uses the Differential evolution (DE) (Cui et al., 2016).

2.2. Hyper-heuristics

Individual *meta*-heuristics can find near optimal solutions on specific types of NP-hard problems as discussed in NFL theory (Wolpert and Macready, 1997). The combinations of *meta*-heuristics provide high-level approaches to optimize a wider range of NP-hard problems. These high-level approaches, known as hyper-heuristics, either select a *meta*-heuristic or generate a new algorithm using the existing components of *meta*-heuristics in each iteration of an optimization problem (Burke et al., 2019). Hyper-heuristics can be categorized into four categories: automated heuristic sequencing, automated planning system, automated parameter control, and automated learning (Ross and Marfn-Blazquez, 2005; Gratch, Chien, and DeJong 1993; C. C. Coello and Lechuga 2002; Duflo et al., 2020).

Automated heuristic sequencing (AHS) algorithms explore the space problems using a sequence of heuristic algorithms, a problem-dependent exploration algorithm, that are ordered by a *meta*-heuristic. Such algorithms shape a finite pool of heuristics and iteratively select a heuristic algorithm that provides the best outcomes in each iteration. AHS is used to solve problems of link timetabling (Ross and Marfn-Blazquez, 2005) and stock cutting (Terashima-Marín,Flores-Alvarez,and Ross, 2005).

Automated planning system (APS) algorithms extend the previous category using *meta*-heuristics to control the selection of *meta*-heuristics in each iteration. APS systems are usually used for problems that need more exploration due to the divergence of problem state spaces. As an example of algorithms in this category, we can name satellite planning systems (Gratch, Chien, and DeJong 1993) and self-learning systems (Sim,Hart,and Paechter, 2015).

Most *meta*-heuristics are adapted to specific search problem by selecting the proper tunning parameters. Automated parameter control algorithms enhance the regular *meta*-heuristics by automatically adjusting the tunning parameters. As this automation process increases the computational overhead, they are usually used for complex decision processing like multi objective optimization. In (C. C. Coello and Lechuga 2002) and (Yusoff,Ngadiman,and Zain, 2011), the authors use heuristic and *meta*-heuristics to handle tunning parameters in PSO and NSGA-II for multi-objective optimizations, respectively.

Automated learning hyper-heuristics try to generate a new algorithm from heuristic and *meta*-heuristics using machine learning algorithms, specifically reinforcement learning. For instance, (Duflo et al., 2020) utilizes a model-free reinforcement learning called Q-learning to generate new algorithms while (Qin et al., 2021) uses model-based reinforcement learning to create new algorithms. Most of these algorithms are limited to a particular search problem. For instance, (Duflo et al., 2020) and (Qin et al., 2021) are designed to solve heterogeneous vehicle routing problem and coverage of a connected unmanned aerial vehicles swarm problems. (Mazyavkina et al., 2021) provides a list of reinforcement-learning-based algorithms designed to solve discrete network search. Unfortunately, the most proposed algorithms in this category are not reusable as they fail to provide their implementations or a clear description of their initial setup. Indeed, the reproducibility and reusability of such algorithms depend on their initial setup

including several hyper-parameters used in their arithmetic computations.

All the above search algorithms, summarized in Fig. 1, suffer from lack of generalizability. They are designed to solve specific search problem and use only a limited number of heuristic and *meta*-heuristics. For instance, (Maashi,Özcan,and Kendall, 2014) can use only 3 *meta*-heuristics. Also, they are not capable of solving both discrete and continuous problem spaces. For instance, (Walker and Keedwell, 2016) and (Cruz-Duarte et al., 2021) can solve only discrete data and continuous data, respectively. On the other hand, in several cases such as (Cruz-Duarte et al., 2021), the proposed hyper-heuristic does not surpass the results of the *meta*-heuristics used to shape hyper-heuristics. In this paper, we will address all the above problems by proposing a novel hyper-heuristic, called World.

3. The proposed method

This section describes the proposed World from two perspectives. Subsection 3.1 provides an outline on the proposed algorithm by describing how we designed an extendable framework to utilize the searching power of any *meta*-heuristics in finding optimal solutions. Following this, Subsection 3.2 details the implementation steps to guarantee reproducibility of the proposed method. Furthermore, an overview of the World algorithm is presented in Fig. 2.

3.1. Outline of World algorithm

World employs a novel reinforcement learning (RL) method to optimize any search problem by vast. The dynamic exploration and exploitation in this algorithm are that different algorithms are implemented, and according to the new reinforcement learning process, with applied adjustments, it selects different exploration and exploitations to solve problems. The dynamics of this search model occur in each step of the algorithm iteration. In separate iterations, a specific algorithm is accomplished, and the way of searching for previously developed algorithms is different from each other.

The RL method empowers iterative World to use the best *meta*-heuristics in each iteration problem state while it avoids starvation of a probable effective *meta*-heuristic by blocking greedy trends of meat-heuristic algorithms. The RL method works in two steps of rewarding and selecting *meta*-heuristics. The first step provides an educated guess about the performance of each *meta*-heuristics in the next iteration. The section step does the selection of the best probable *meta*-heuristic for exploration and exploitation in the next iteration.

The rewarding *meta*-heuristics step provides a fair comparison environment for all the accessible algorithms. In the first iteration, all *meta*-heuristics search from an identical, arbitrary, initial population of the solutions to find an optimal solution, e.g., in a simple minimization problem of $f(x)$ where $x \in R, f : R \rightarrow R$ *meta*-heuristic of i starts with m

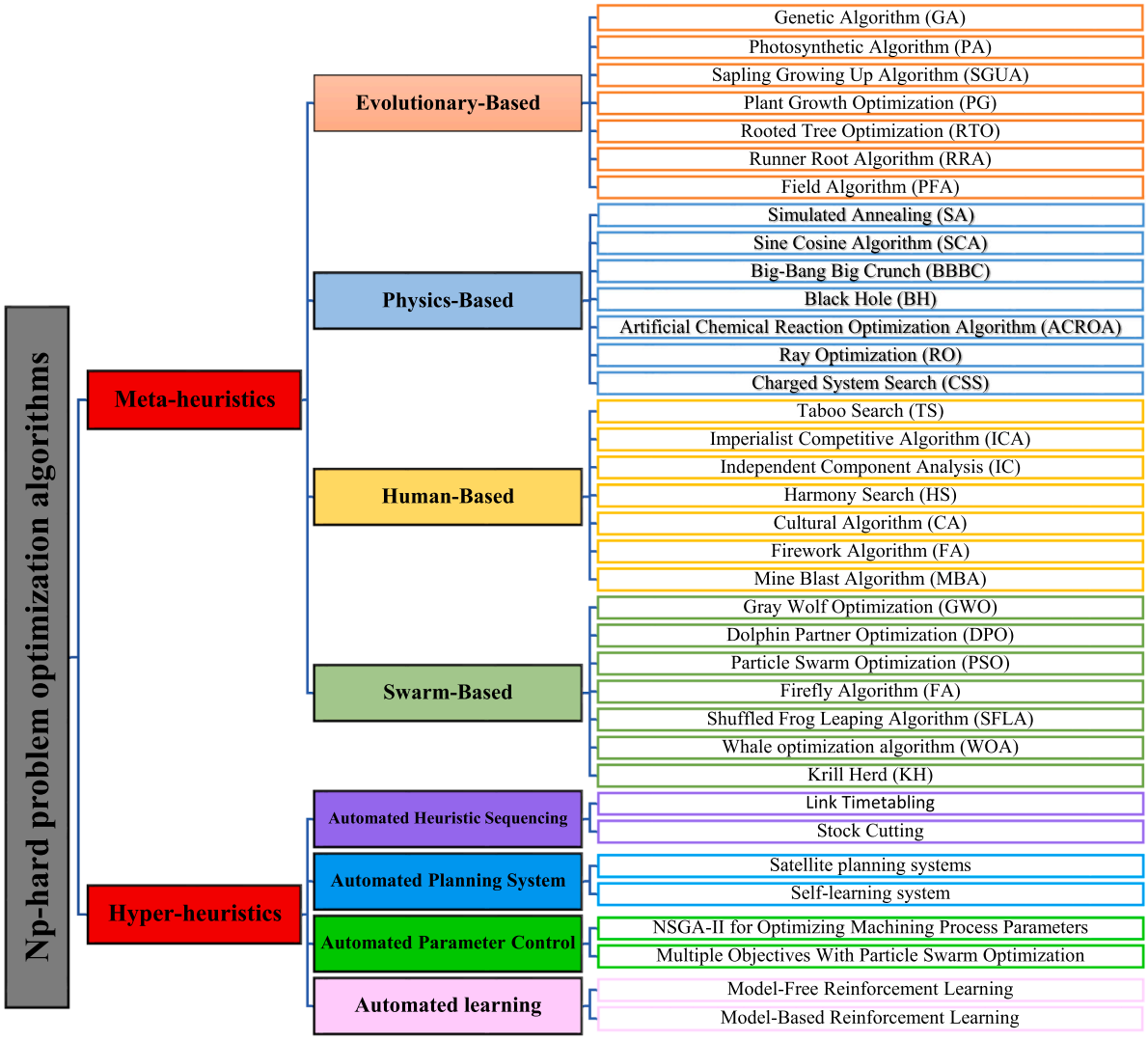


Fig. 1. Classification of Np-hard problem optimization algorithms.

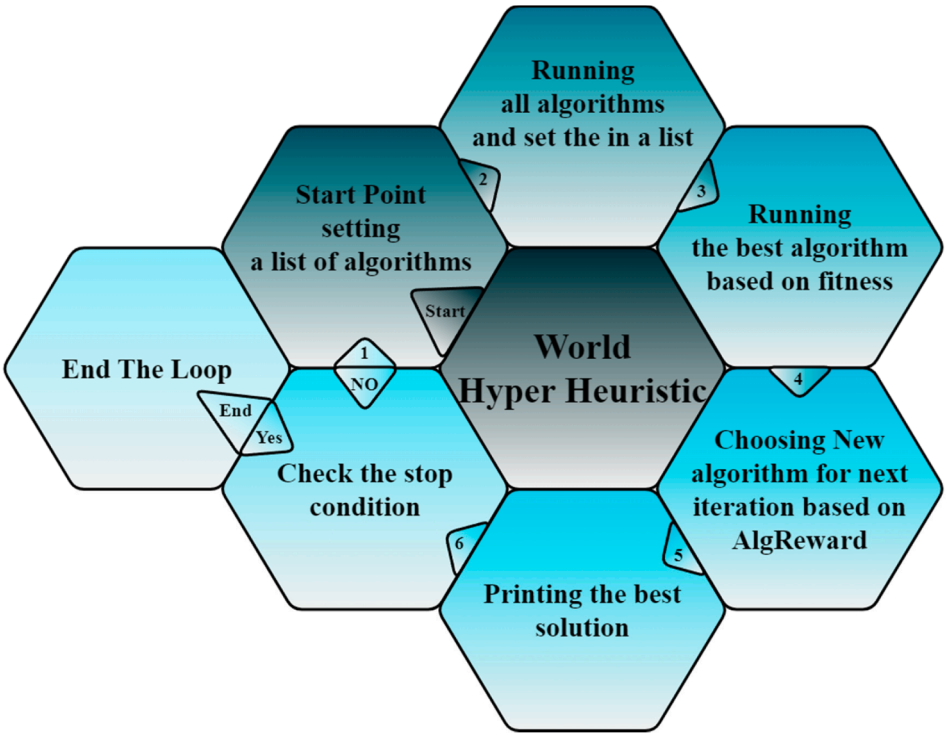


Fig. 2. overview of the World algorithm.

arbitrary population of solutions $\{Z_1, Z_2, Z_3, \cdots, Z_m\} \in R$ and reaches $L_i \in R$, as the optimal solution in the first iteration. We will group the outcomes of such a population-based algorithm into an array of $gpopCostA = \{L_1, L_2, \cdots, L_i\}$ where $i \in [1, n]$ and n is equal to the number of total *meta*-heuristics. Following this, we will prioritize the most successful algorithm in finding the minimization of $f(x)$ by forming the array of *AlgReward* as shown in Equation (1).

$$\boldsymbol{AlgReward} = 1/\boldsymbol{gpopCostA} \tag{1}$$

To achieve a fast and successful search, the selection step of the RL method follows *AlgReward* as an educated guess, explores all probable solutions in an uncertain search state space, reduces computation complexity to run only one *meta*-heuristic per iteration, and dynamically updates the educated guess based on the findings in each iteration. The selection step satisfies the mentioned requirements using a learning rate variable of *RLAlpha*, which is set to zero for the first iteration to give all the algorithms the same chance for selection. *RLAlpha* in reinforcement learning methods is the learning rate. The policy to set the learning rate value differs from one implementation to another and it can include criteria such as time and number of iterations. In our algorithm, to get the best result, according to several experiments, we set the value of *RLAlpha* in the range of 1 to 10.

As shown in Equation (2), this variable increases by $1/$ *maximumiterations* in each iteration and is bounded on a range of [0,10]. Given the growing nature of *RLAlpha*, the selection step moves toward the educated guess of roulette wheel section, which gives more chance to the algorithms with higher *AlgReward*. The trial-and-error strategy reveals World optimal performance when *RLAlpha* is set in the range of [0,10]. This trend of *RLAlpha* empowers World to follow diversify search strategy in the initial iterations and move toward intensify search strategy in the final iterations. The RL method employs *choose-random* variable to control the process of moving from exploration to exploitation in each iteration. This variable represents a random number between 0 and 1 that is generated in each iteration. In the case *choose-random* is less than *RLAlpha*, the algorithm with the higher reward has more chance to be selected since the RL method uses the roulette wheel selection to select the next iteration *meta*-heuristics. The RL method also constantly checks the optimization performance to reduce the award of the top algorithm in *AlgReward* array if the optimization performance does not change for two consecutive iterations. As shown in Equation (3), this reduction will be equal to the average of the reward of all the other algorithms. These two latest adjustments save World from getting stuck in a probable local minima and starvation in higher iterations.

$$\boldsymbol{RLAlpha} = \boldsymbol{RLAlpha} + (1/\boldsymbol{maximumiterations}) \tag{2}$$

$$\boldsymbol{AlgReward}_i = \boldsymbol{AlgReward}_i - \frac{\left(\sum_{j=1}^{j \neq i} \boldsymbol{AlgReward}_i\right)}{\boldsymbol{n}} \tag{3}$$

3.2. Details of World

World brings the RL idea of rewarding-selection, which is described in Subsection 3.1, into action through three phases. First, the hyper-parameters used in Word are set up. Then, the pool of *meta*-heuristics is managed by the rewarding phase. Finally, iterative exploration and exploitation are performed to find the optimal solution. This section describes these three phases in detail.

For the sake of simplicity, these three phases are described separately using three algorithms. In Algorithm 1, all the hyper-parameters of the World algorithm are described and initialized. The input clarifies 11 hyper-parameters, and the final output of World is expected to be an optimal solution stored in *Best Solution*. Algorithm 2 is the learning phase of the World algorithm such that *Bestsol,* the objective function value of the best solution per iteration, is used as a pivot for the conditional learning. Following this, *Bestsol* of the last iteration indicates the *Best Solution,* which is the near-optimal solution. Algorithm 3 describes the process of dynamic shifting between exploration and exploitation strategies in each iteration.

Algorithm 1 indicates all the input hyper-parameters of World. *npop* specifies the number of solution populations, which is considered the same for all the *meta*-heuristics. *maxIteration* sets the maximum number of iterations for explorations and exploitations in the third phase.

Number of *meta*-heuristics and the learning rate are indicated as *nAlg* and *RLAlpha*, respectively. The parameters *varMin* and *varMax* are used to specify the domain for the optimization problem. *BestSol* is an array to store the objective function value of the best solution per iteration. The algorithms' rewards are stored in the matrix *AlgRewards*. The variable *chN* is a random number within [0,1] that is assigned to the algorithms in each iteration to control exploration and exploitation on phase 3. *LotchList* tracks the *meta*-heuristics in each iteration by recording the identifier of each algorithm. *Lotch* also indicates the identifier of the algorithm that is executed in the current iteration, and it will be stored at *LotchList*. The counter for the main loop is called *iter* and indicates the number of iterations run from the beginning of World.

Algorithm1. The first phase of World.

Inputs:
npop // number of populations
maxIteration //number of iterations
nAlg //number of algorithms
RLAlpha = 0 //learning rate of RL- in first alpha is 0
varMin, *varMax* //domain of Problem
BestSol // the objective function value of the best solution per iteration
AlgRewards //Reward of per algorithm
chN = rand(1) // a random number between 0 and 1
lotchList // an array contain number of per iterations algorithm
lotch //number of per iteration Algorithm
iter //iteration counter
Output: ***Best Solution***

The second phase of World initiates all the parameters and data structures required for each *meta*-heuristic. Line 2 in Algorithm 2 creates the initial random solution population. In the next step, Lines 3 and 4 calculate the fitness of the initial population and run all the algorithms using the initial population. As mentioned before, algorithms experience larger steps in exploring the search area at the first iterations of their run. To provide fair conditions for all algorithms, same initial population is used at the first iteration to run the algorithms. After running the first iteration, rewards are calculated for each *meta*-heuristic. If the solution found by a *meta*-heuristic has a positive value, the reward assigned to the algorithm is calculated as $AlgRewards_i = 1/(Bestsol + 1)$, where *BestSol* indicates the objective function value of the best solution per iteration. Otherwise, the reward is computed as $AlgRewards_i = |(Bestsol + 1)|$.

Algorithm 2. The second phase of World.

1. Generate initial parameters required by all algorithms
   Generate initial population by Random
   Compute *fitness* of population
   Execute all algorithms by the initial population
   For i = 1: *nAlg*
   calculate for each algorithm based on the following formula:
   If BestSol > 0
   AlgRewards = 1/(BestSol + 1) // a number between 0,1
   else
   AlgRewards = |(BestSol + 1)| // a number between 0,1
   Sort all algorithms by *AlgRewards*

The third phase of World controls the iteration and searches through problem state space by the main loop where number of iterations is specified by *maxiter*. For a better understanding of this phase, we will first give a detailed explanation of some parameters. *chN* is a variable for exploration and exploitation and plays the role of selecting different algorithms here. In such a way that if an algorithm does not get a score or the scores are equal, exploration is done using *chN* and the operation that is executed in *LotchList*.

To better understand the state of the presented reinforcement learning algorithm, it should first be noted that the repetition of each step of the algorithm, which is called iteration, is the state of the problem. In this phase, the list of algorithms is either selected based on algorithm score (AlgRewards) or luck (Lotch). If it is based on chance, it means that the reward of the previous state (BestSol(iter-1)) and the current state (BestSol(iter)) is less than the generated random amount (Lotch).

In the first step, *chN*, which is a random number between 0 and 1, is assigned for each iteration. On Line 14, if the value of *chN* is greater than the learning rate, *RLAlpha*, the *meta*-heuristic associated with *chN*, is chosen to be run. Otherwise, the algorithm is selected using a roulette wheel that works based on the fitness values of all the *meta*-heuristics. Then, the identifier of the selected algorithm that is specified by *Lotch* is stored in *LotchList*. After selecting one algorithm per iteration, the rewards are computed, *AlgRewards* and *RLApha* are updated, and the *meta*-heuristics are sorted based on their new rewards.

Algorithm 3. The third phase of World.

For iter = 1 to *maxIteration* do
chN = Generate a random number between 0,1
if chN > RlAlpha:
Lotch = Generate a random number between 1 and *nAlg*
Else:
Lotch = Generate a random number between 1 and *nAlg* by Roulette wheel
Add *Lotch* into *LotchList* list
Execute the algorithm by number = *lotch*
Update reward of algorithm
AlgRewards = *AlgRewards*+ | *BestSol*(*iter*-1) - *BestSol*(*iter*) |If BestSol (*iter*) == *BestSol*(*iter*-1) // answer is not better
AlgRewards = *AlgRewards*- mean (*AlgRewards*)
Sort all algorithms by *AlgRewards*
Sort populations by fitness
Choose best answer == *BestSol*
RLAlpha = *RLAlpha* + (1/*maxiter*)
Output = ***Best Solution***

Fig. 3 summarizes the three phases of Word in the form of a flowchart. The first five processes describe the required procedures in phase 1 and 2. The rest summarizes the process in phase 3.Fig. 4..

4. Experimental results and discussion

We evaluated the performance of World in exploring both discrete and continuous search problems. The details of these 30 search problems and initialization of the hyper-parameters are described in Subsection 4.1. Following this, Subsection 4.2. scrutinize the trends of our novel proposed algorithm in solving 26 discrete and continuous artificial problem sets. We examined the capability of World in addressing both discrete and continuous real search problems on Subsection 4.3. For the real evaluation in the discrete field, a real database for the traveling salesman problem is used, where the results of the best answers are available. We have used this problem to evaluate the performance and show the competitiveness of the output of the proposed algorithm. Finally, in Subsection 4.4, we reviewed and discussed various issues related to hyper-heuristic algorithms.

4.1. Problems sets and initial hyper-parameters

We evaluated World's performance in solving discrete problems such as travelling salesman problem (TSP). TSP is a NP-hard problem for finding the shortest tour in which each city must be visited precisely once (Russell and Norvig, 2002). We analyzed World's trends in solving a TSP with the stochastic coordination of 1000 artificial cities where X and Y are in the range of [0,100]. Then, we extended our experiments using a real TSP, which comprises the geographical place information of 10,000 real cities around the world (World Traveling Salesman Problem." n.d. Accessed October 4, 2020). We also examined World for the problem of combinatorial-optimization-based threat evaluation and jamming allocation (COTEJA) (You,Diao,and Gao, 2019). In the context of continuous search problem, we used 25 well-known standard functions, which are listed Appendix1. These mostly non-convex functions have different global minimum values, which are listed in the third column with the name of *Fmin*. Following this, we expanded our analysis on real problems of welded beam design (Wang et al., 2012) and speed reducer (Hassan et al., 2005).

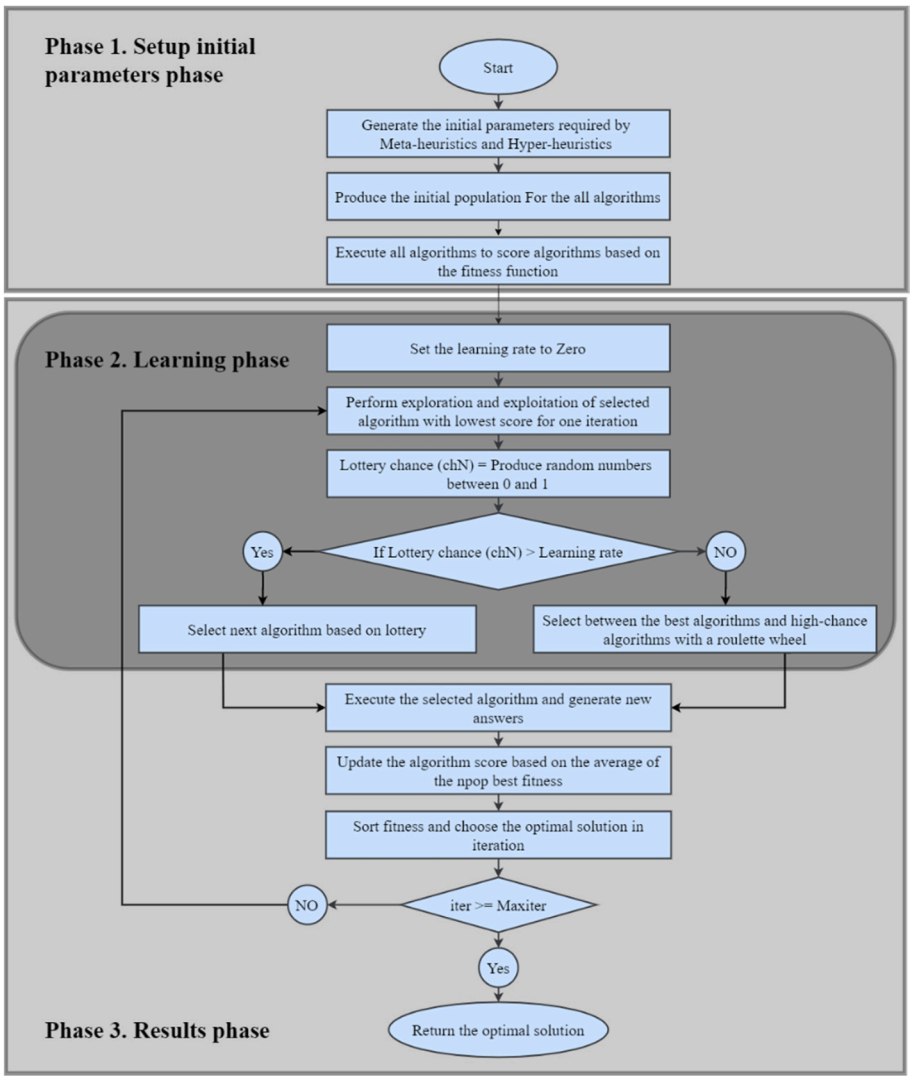


Fig. 3. the flowchart of World.

In the initial examination, all the standard test optimization functions were implemented in dimension of $n = 5$ and input variable ranges of $x_i = [-10, 10]$ to examine all search algorithms in a similar condition. Following this, we present the outcomes of search algorithms in the dimensions of 10, 20, 30, and 50. We examined the performance of World on the different search problems by the utilized *meta*-heuristics in World and the customized search algorithms for each case study. Specifically, we used Lin-Kernighan Heuristic (LKH) (Tinós,Helsgaun,and Whitley, 2018; "World Traveling Salesman Problem" n.d.), hyper-heuristics to customize *meta*-heuristics (HHCM) (Cruz-Duarte et al., 2021), and chaos great deluge algorithm (GDA) (Baykasoglu, 2012) for Real TSP, standard continuous functions, Welded beam design and Speed reducer problem, respectively.

The experiments were designed to use a version of World with 6 and 10 discrete and continuous *meta*-heuristics, respectively. In general, there are two types of algorithms for problems. Algorithms that optimize problems with discrete data and algorithms that can optimize continuous problems. Therefore, in the World algorithm, we implement two types of algorithms that can optimize both types. Table 1 presents the *meta*-heuristics used to deal with discrete data. To provide a similar starting condition for all the algorithms, World's hyper-parameters of population size, the number of generations for continuous problems, the number of generations for 10,000 cities, the number of generations for the other search problems were initialized with values of 50, 1000, 1000, and 100, respectively. The *meta*-heuristic specific hyper-parameters were set with the default values suggested by their reference paper. For instance, the two hyper-parameters of *Control Parameter* and *number of particles* in GWO algorithm were set to 50 and [0,2] according to (Mirjalili,Mirjalili,and Lewis, 2014). The similar initialization procedure were applied for the hyper-parameters of ICA, SA, and ACO algorithms based on the suggestions of (Atashpaz-Gargari and Lucas, 2007; Kirkpatrick, Gelatt, and Vecchi 1983; Dorigo,Birattari,and Stutzle, 2006).

The initial values of the selected continuous *meta*-heuristics were set to the values recommended by their original papers. Every algorithm that is built has parameters that are adjusted to make the algorithm work better either experimentally or using personalized optimization methods. In order to make the experimental results fair, we have implemented the parameters of each algorithm based on their own reference paper. These values are listed in Table 2. The hyper-parameters of *Population Size* and *Number of Generations* are common in most of the *meta*-heuristics and are set to 50 and 1000 to provide a fair search condition for all the *meta*-heuristics.

4.2. Discrete and continuous artificial experiment results

The World's performance and the individual discrete *meta*-heuristics in solving artificial TSP are reported in Table 3. The results on the best

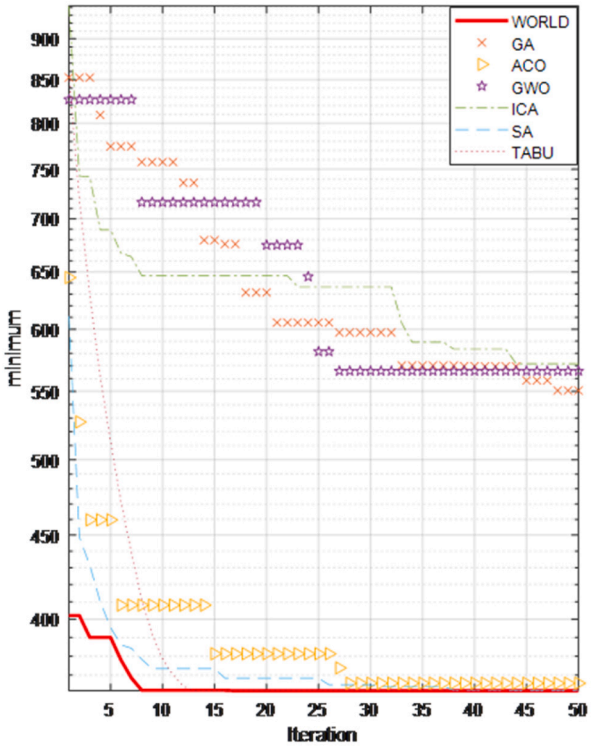


Fig. 4. The results of discrete algorithms over the TSP in all iterations.

Table 1
Hyper-parameter values of the discrete algorithms.

Algorithms	Parameters	Values
Ant Colony Optimization (ACO)	Population size	50
	Number of generations	100
	Conversion ratio fitness	100
Genetic Algorithm (GA)	Population size	50
	Number of generations of 10,000 cities	1000
	Number of generations for other	100
Grey Wolf Optimization (GWO)	Number of generations for other	[2,0]
	Number of generations of 10,000 cities	1000
	Number of generations for other	100
	Number of particles	50
Imperialist Competitive Algorithm (ICA)	Number of countries	50
	Number of generations of 10,000 cities	1000
	Number of generations for other	100
	Number of nimp	10
Simulated Annealing (SA)	Population size	50
	Number of generations of 10,000 cities	1000
	Number of generations for other	100
	Number of neighbors	10
Tabu (Taboo)	Population size	50
	Number of generations of 10,000 cities	1000
	Number of generations for other	100

column demonstrate the capability of World in finding the shortest path and outperforming over the individual *meta*-heuristics. The average of minimum solutions discovered in all iterations of World is less than the other algorithms. As a result, our proposed algorithm maintains the best performance in all the iterations and does not reach the best minima by a random trend. It is worth mentioning that the standard deviation of the results in all iterations for World is far less than the other *meta*-heuristics. The low standard deviation of World reveals the capability of our algorithm in managing the explorations and exploitations.

Table 2
Hyper-parameter values of continuous algorithms.

Algorithm	Parameters	Values
Adaptive differential evolution algorithm (ADEA)	Population size	**50**
	Number of generations	**1000**
Artificial bee colony algorithm (ABC)	Population size	**50**
	Number of generations	**1000**
Biogeography Optimization (BBO)	Alpha	**0.9**
	Number of generations	**1000**
	Keep Rate	**0.4**
Differential Evolution (DE)	Beta min	**0.2**
	Beta max	**0.8**
	PCR	**0.2**
	Number of generations	**1000**
Firefly Algorithm (FA)	Gamma	**1**
	Beta	**2**
	Number of generations	**1000**
Genetic Algorithm (GA)	Population size	**50**
	Number of generations	**1000**
	Gamma	**0.4**
Harmony Search (HS)	Number of nimp	**10**
	Neighboring value rate	**0.3**
	Discrete set and fret width	**17700,1**
	Number of generations	**1000**
Improved Invasive Weed Optimization (IWO)	Exponent	**2**
	sigma initial	**0.5**
	sigma final	**0.001**
	Number of generations	**1000**
Particle Swarm Optimization (PSO)	Inertia coefficient	**0.75**
	Cognitive & social coeff	**[1.8,2]**
	Number of generations	**1000**
	Search agents	**100**
Self-adaptive Differential Evolution Algorithm (SADE)	Population size	**50**
	Number of generations	**1000**
Shuffled Frog-Leaping Algorithm (SFLA)	Memeplex	**10**
	Alpha	**3**
	Beta	**5**
	Number of generations	**1000**
Simulated Annealing (SA)	Alpha	**0.99**
	Mu	**0.5**
	Number of generations	**1000**
Single objective real-parameter optimization (jSO)	Population size	**50**
	Number of generations	**1000**
Teaching-Learning Optimization Algorithm (TLBO)	Population size	**50**
	Number of generations	**1000**

Table 3
The evaluation results of discrete algorithms over the Travelling salesman problem (TSP).

Std	Best	Avg	Algorithms
391.77	3.65e + 02	391.77	ACO
92.41	5.50e + 02	643.69	GA
96.16	5.69e + 02	649.47	GWO
60.64	5.71e + 02	633.31	ICA
37.77	3.62e + 02	376.53	SA
97.32	3.62e + 02	397.24	Tabu
10.25	**3.52e + 02**	**365.81**	**World**

In the results section of each algorithm, in order to present the results in a fair way, comparisons are made based on the number of loop repetitions of each algorithm. This means that each algorithm performs the number of iterations of its objective function and finally the result of the last iteration, which is probably the best iteration, is presented. To scrutinize the trend of World in solving artificial TSP, the outcomes of World and *meta*-heuristics are visualized in Fig. 3. We can categorize the algorithms based on the pace of convergence into two categories of fast and slow convergent algorithms. World, TABU, ACO, and SA placed in fast category; while, ICA, GWO, and GA form the slow category. World maintains the lowest minima in each iteration and has the fastest convergence that finds the minimum value of 3.52e + 02 in iteration 7. On the other hand, TABU, ACO, and GA reached convergence in

iterations 12, 28, and 48, respectively.

To extend the previous analysis, we examined the outcomes of the World algorithm in comparison to the rest of the algorithms in the different dimensions of 10, 20, 30, and 50. As shown in Table 4, the proposed method followed the same trend in all the higher dimensions of TSP.

We also analyzed the performance of World on mostly non-convex continuous test optimization functions, introduced in Subsection 4.1 and Appendix 1. As reported in Table 5, World achieved the highest performance in finding the minimum values of all the optimization cases. World also managed to surpass the hyper-heuristic known as HHCM (Cruz-Duarte et al., 2021) in all optimization of the standard functions. Please note that World discovered (near-)global minimum values for all the artificial functions, described in Subsection 4.1 and Appendix 1. These results suggest the capability of our algorithm in finding near-optimal solutions for NP-hard problems, where global minimums are challenging to compute. The diversity in the results of the *meta*-heuristic algorithms reveals that the *meta*-heuristics have different performance in each search problem state. For instance, FA returns 3.31 and DE returns 2.98 as optimal minima, which indicates the DE algorithm has better search minima in contrast to FA in Ackley function. However, the outcomes of these two algorithms on finding the optimal minima for Schwefe 2.20 function draw reverse conclusion.

We also evaluated the World's performance in comparison to the rest of the algorithms in the different dimensions of 10, 20, 30, and 50. As shown in Table 6, the proposed method followed the same trend in almost all the higher dimensions of standard functions. The further analysis outcomes for Table 5 outcomes and higher dimension experiments are reported in Appendix 2.

To gain more insight into the World performance, we scrutinized World's trends in solving four continuous search problems of Quartic, Qing, Rastrigin, and Xin-She Yang illustrated in Figs. 5-7. As shown in Fig. 5, our proposed algorithm reached the near-optimal minimum value for Quartic function that is far better than the final results of the other algorithms. In addition, World reached the stable points in the first 50 iterations. However, the second algorithm with lowest minima, which is SFLA algorithm, used more than 900 iterations to report its final result. These outcomes demonstrate that World goes with proper exploration and exploitation such that after each iteration the gap between World results and the rest of the algorithms' results grows rapidly. This trend of World satisfies reaching optimal results in a reasonable time.

As illustrated in Fig. 6, World search in Qing function surpasses the other search algorithms in finding the minimal point similar to the previous case. However, from the other perspectives, our proposed algorithm demonstrates different trend for Qing function. In this search state space, World continues exploration and exploitation for a long time of 950 iterations to find optimal minimum points. This reveals the adaptability of World with the search space and ability to switch between exploration and exploitation. Indeed, World can jump out of local minima and perform exploration on jumps, e.g., in the iteration 300.

In two previous cases, out proposed algorithm reached far better results regardless of the number of iterations. However, there are some cases like Rastrigin and Xin-She Yang functions, illustrated in Fig. 7, that World only reached the best performance faster than the other algorithms. In another words, the rival algorithms can reach the similar results if they have adequate iterations of exploration and exploitation. SA in Rastrigin function and GA in Xin-She Yang function have slower trends to discover optimal minima rather than World. We can hence conclude that even if the other search problems can reach the same minima, World can still enhance the convergence speed in search problems. All the above case studies bring light to the outperformance of World over the other search algorithms in certain number of exploration and exploitation iterations.

Table 4
The evaluation results of discrete algorithms over the Travelling salesman problem (TSP) in the dimensions of 10, 20, 30, and 50.

Algorithms	Criteria	10	20	30	50
World	Best	3.59e + 02	3.63e + 02	3.93e + 02	3.95e + 02
	STD	10.36	10.52	10.95	10.98
	Mean	368.23	371.03	381.22	383.12
ACO	Best	3.72e + 02	3.73e + 02	3.79e + 02	3.78e + 02
	STD	391.77	391.82	392.01	391.94
	Mean	391.77	391.75	392.15	391.87
GA	Best	5.55e + 02	5.54e + 02	5.62e + 02	5.75e + 02
	STD	92.44	92.49	93.09	97.04
	Mean	643.05	642.57	646.47	648.06
GWO	Best	5.69e + 02	5.82e + 02	5.91e + 02	5.94e + 02
	STD	96.16	98.52	99.84	102.19
	Mean	649.47	658.06	654.44	656.98
ICA	Best	5.70e + 02	5.73e + 02	5.84e + 02	5.85e + 02
	STD	62.51	62.98	67.22	69.22
	Mean	635.12	636.12	642.19	644.05
SA	Best	3.62e + 02	3.69e + 02	3.71e + 02	3.74e + 02
	STD	37.82	39.20	40.15	42.79
	Mean	375.57	381.47	376.53	384.11
TABU	Best	3.64e + 02	3.68e + 02	3.72e + 02	3.78e + 02
	STD	95.78	97.25	99.12	104.87
	Mean	395.32	399.75	400.41	402.24

4.3. Discrete and continuous real experiments

Now, we report the outcomes of World in solving 4 real-world NP-hard problems. The comparisons include Real TSP and COTEJA problems to evaluate World's performance in solving discrete search problems. Following this, we explore the World's performance in solving continuous search problems using Welded beam design and Speed reducer problems. In Real TSP problem, World explored the search problem of finding the shortest path among 10,000 real cities. The number of cities is increasing over time and it may increase when this data is consulted. ("World Traveling Salesman Problem" n.d.). This path visits each city exactly once and returns to the starting point. These cities are scattered around the world and their geographical coordinates are illustrated in Fig. 8. TSP problem and the utilized dataset are detailed in Subsection 4.1.

The outcomes of World and the other search algorithms from four perspectives on Real TSP problem are reported in Table 7. The best column reports the shortest path with the length of 4.33e + 05 found by our proposed algorithm. This result is significantly better than the shortest path with the length of 7,512,218,268, which is recently reported by Lin-Kernighan heuristic (Tinós,Helsgaun,and Whitley, 2018; "World Traveling Salesman Problem" n.d.) on Feb 2021. Since World reached the lowest average of solutions in all the iterations, we conclude that World maintains its outperformance in finding the shortest path in all the iterations. Regarding the initial/worst steps, World is ranked as the 6th algorithm with value of 1.62e + 06 and has the highest standard deviation among the other search algorithms. Thus, both standard deviation and worst/initial values suggest the World's ability in performing a vast number of exploration and exploitation searches.

We magnified the trends of search algorithms in each iteration in Fig. 9 to scrutinize the exploration and exploitation behaviors of the search algorithms simultaneously. The solid red line shows the outcomes of World in each iteration. This line illustrates some big jumps with the length more than 0.15 in finding the minimum values. These jumps can represent explorations, which lead to notable changes in the optimization process for certain iterations, such as 340. These effective

Table 5
The Best minimum results of continuous algorithms over the standard functions.

Algorithm Function	SADE	jSO	ABC	ADEA	TLBO	SFLA	SA	PSO	IWO	HS	GA	FF	DE	BBO	HHCM	World
sphere	5.780e-313	4.547e-112	4.030e-101	1.560e-322	1.415e-266	5.939e-143	3.3313e-21	0	2.9454e-07	3.3485e-09	5.0967e-49	6.2241e-49	3.9758e-89	1.1321e-13	0	0
rosenbrock	0	0	0	8.780e-123	1.8315e-06	1.2106e-15	1.2106e-15	0.0153	0.0061	1.7041e + 03	2.6244	0.0275	0.0275	0.1720	0	0
quartic	1.449e-234	4.930e-145	0	0	2.3848e-04	2.7229e-05	2.9516e-04	1.1826e-04	5.1120e-04	0.0013	5.9883e-04	1.4026e-04	5.1770e-04	1.1485e-04	0	0
Schwefe220	0	0	7.053e-123	0	1.985e-135	8.6085e-73	7.9848e-11	1.222e-195	7.4293e-04	1.145e-04	1.1008e-78	1.3443e-78	2.2291e-48	2.1270e-06	0	0
Schwefel221	0	1.236e-129	0	9.390e-234	1.251e-118	1.5193e-55	4.6120e-11	1.686e-148	2.8003e-04	5.6293e-05	3.4279e-06	4.1862e-06	1.2954e-22	6.2360e-05	0	0
Schwefel222	9.204e-321	7.145e-123	3.126e-456	4.156e-321	9.767e-135	6.9791e-72	6.8785e-11	3.734e-196	9.6474e-04	1.4547e-04	2.1074e-18	2.5736e-18	9.2760e-48	1.6050e-07	0	0
Schwefel223	0	0	0	0	0	0	5.258e-105	0	7.7481e-36	9.4545e-47	3.9822e-30	4.8631e-30	0	4.7166e-57	0	0
Sum squares	0	0	0	0	1.540e-267	4.391e-144	1.6377e-21	0	1.9165e-08	1.9165e-08	1.2102e-12	1.3569e-12	2.9471e-89	1.4522e-14	0	0
brown	0	0	0	0	2.252e-270	1.556e-140	1.0455e-21	0	3.3791e-07	6.1092e-09	8.458e-130	1.025e-130	1.0991e-90	3.7793e-14	0	0
Rastrigin	0	0	0	0	0	0.9950	0	0	0.9950	2.8879e-06	9.6634e-13	1.6257e-11	0	1.6257e-11	0	0
griewank	0	0	0	0	5.2180e-15	0.0246	0.0197	0.0345	0.0271	0.0345	0.0172	0.0042	0	0.0271	0	0
ackley	0	7.265e-129	6.037e-129	0	2.9805	2.9805	2.9805	2.9805	2.9806	2.9805	2.9805	3.3149	2.9805	2.9805	0	0
exponential	−0.8555	−0.9554	−1	−1	−1	−1	−1	−1	−1.0000	−1.0000	−1.0000	−0.1212	−1	−1.0000	−1	−1.0000
periodic	1.1121	1.0121	1	1	1.0001	1	1	1.0000	1.0000	1.0000	1	1.2212	1	1.0000	1.1	1.0000
powellsum	0	0	0	0	0	1.509e-177	4.4278e-27	0	3.3729e-11	1.0475e-10	1.0675e-10	5.192e-140	4.252e-140	4.5311e-17	0	0
salomon	0	1.032e-103	0	1.125e-178	0.0999	0.0999	0.0999	0.0999	0.0999	0.1999	0.0999	0.1220	0.0999	0.1000	0	0
shubert3	−74.2314	−71.0369	−74.2310	−74.2101	−74.1898	−74.189	−74.1898	−74.1898	−62.4338	−74.1897	−74.1898	−74.1890	−74.1898	−74.1898	−74.1900	−74.1898
styblinskitank	−190.0232	−191.2340	−191.0212	−189.2013	−195.8308	−195.8308	−195.8308	−195.8308	−181.6941	−195.8308	−195.8308	−195.8289	−195.8308	−195.8308	−195.8500	−195.8308
xinsheyangn1	0	0	0	0	2.2482e-66	1.9665e-18	1.6165e-10	2.5077e-18	0.0012	2.1435e-08	5.4785e-86	2.1435e-08	1.6234e-23	1.1281e-12	0	0
Xinsheyangn2	0.0232	0.0012	0	0	0.0418	0.0418	0.0418	0.0524	0.0418	0.0418	0.0418	0.0515	0.0418	0.0418	0	0
Xinsheyangn3	0.9096	0.9036	0.9566	0.9322	0.9169	0.9187	0.9169	0.9169	0.9169	0.9169	0.9169	0.9169	0.9169	0.9169	0.9312	0.9187
Xinsheyangn4	−2.0854e-25	−2.1215e-25	−2.0321e-25	−2.0412e-25	−3.0484e-24	−3.0484e-24	2.8265e-24	−1.5391e-19	1.6691e-09	5.9587e-12	−3.0432e-24	−3.7496e-24	−3.0484e-24	5.2830e-18	−2.0942e-24	−2.0748e-25
zakharov	0	0	0	0	5.866e-216	5.0519e-64	3.1791e-20	2.789e-251	3.6550e-07	5.0941e-07	4.9182e-08	7.0434e-08	2.3274e-26	4.1266e-08	0	0
qing	1.0111e-53	1.2324e-23	1.2344e-21	1.0669e-15	1.0236e-23	1.1833e-30	2.0806e-20	1.1833e-30	6.5687e-07	3.4385e-08	6.5647e-06	6.5647e-06	1.1833e-30	2.4456e-11	1.2577e-55	1.2293e-54
Ridge	−10	−10	−10	−10	−10	−10.0000	−10.0000	−10	−9.9999	−10.0000	−10.0000	−9.9999	−10	−10.0000	−10.0000	−10.0000

Table 6
The Best minimum results of continuous algorithms over the standard functionsin the dimensions of 10, 20, 30, and 50.

Function	Criteria	10	20	30	50
Ackley	Best	0	0	0	3.98e-32
	STD	1.46	1.48	1.48	1.49
	Mean	1.84	1.85	1.86	1.85
Brown	Best	0	0	0	1.16e-22
	STD	3.04	3.04	3.06	3.06
	Mean	0.13	0.13	0.14	0.14
Exponential	Best	−1.00	−1.00	−1.00	−1.00
	STD	0.06	0.06	0.07	0.07
	Mean	0.01	0.01	0.01	0.01
Griewank	Best	0	0	0	0
	STD	0.01	0.01	0.02	0.02
	Mean	0.01	0.01	0.01	0.01
Periodic	Best	0.9	0.9	0.9	0.9
	STD	0.04	0.04	0.05	0.05
	Mean	0.97	0.97	0.97	0.97
Powell Sum	Best	0	0	0	6.32e-29
	STD	1.72	1.72	1.72	1.73
	Mean	0.07	0.07	0.07	0.07
Qing	Best	8.09e-56	7.73e-52	8.26e-51	4.04e-41
	STD	1.18	1.18	1.19	1.19
	Mean	0.06	0.06	0.06	0.07
Quartic	Best	0	0	0	1.11e-31
	STD	29.41	29.41	29.42	29.42
	Mean	1.33	1.33	1.34	1.34
Rastrigin	Best	1.96e-215	4.56e-221	7.65e-195	1.14e-14
	STD	3.07	3.07	3.08	3.09
	Mean	0.20	0.21	0.21	0.22
Ridge	Best	−10	−10	−10	−10
	STD	0.08	0.08	0.08	0.08
	Mean	−9.99	−9.99	−9.99	−9.99
Rosenbrock	Best	2.22e-63	1.98e-59	3.35e-61	6.38e-61
	STD	1.20e + 03	1.20e + 03	1.20e + 03	1.21e + 03
	Mean	52.28	52.27	52.28	52.28
Salomon	Best	0	0	0	0
	STD	0.04	0.04	0.04	0.04
	Mean	0.07	0.08	0.08	0.08
Schwefe 2.20	Best	0	0	0	7.89e-211
	STD	0.29	0.29	0.29	0.29
	Mean	0.02	0.02	0.02	0.02
Schwefel 2.21	Best	0	0	0	2.75e-39
	STD	0.14	0.16	0.14	0.19
	Mean	0.01	0.01	0.01	0.01
Schwefel 2.22	Best	0	0	0	0
	STD	0.36	0.37	0.37	0.37
	Mean	0.02	0.02	0.02	0.03
Schwefel 2.23	Best	0	0	0	0
	STD	787.88	787.88	787.88	787.89
	Mean	33.02	33.02	33.02	33.03
Schubert 3	Best	−74.18	−74.18	−73.16	−70.26
	STD	4.98	4.98	4.98	4.99
	Mean	−72.62	−72.61	−72.62	−72.62
Sphere	Best	0	0	0	5.51e-39
	STD	0.50	0.51	0.51	0.51
	Mean	0.02	0.02	0.03	0.02
Styblinskitank	Best	−195.83	−190.31	−191.87	−191.81
	STD	2.49	2.50	2.50	2.50
	Mean	−195.66	−195.67	−195.67	−195.67
Sum squares	Best	0	0	0	0
	STD	0.78	0.78	0.78	0.78
	Mean	0.03	0.03	0.03	0.04
Xin-She Yang	Best	0	0	0	0
	STD	0.29	0.29	0.29	0.29
	Mean	0.01	0.01	0.01	0.01
Xin-She Yang N. 2	Best	0	0	0	0
	STD	0.01	0.02	0.02	0.02
	Mean	0.03	0.03	0.04	0.04
Xin-She Yang N. 3	Best	0.91	0.98	1.01	1.06
	STD	0.01	0.01	0.01	0.01
	Mean	0.92	0.92	0.92	0.92
Xin-She Yang N. 4	Best	2.17e-28	2.24–26	2.16e-26	2.97e-25
	STD	4.51e-04	4.51e-04	4.51e-04	4.51e-04
	Mean	5.54e-05	5.54e-05	5.541e-05	5.54e-05
Zakharov	Best	0	0	0	0
	STD	2.79	2.79	2.79	2.79
	Mean	0.12	0.12	0.12	0.13

explorations may aid World to surpass ACO and GWO algorithms in finding the optimal path in Real TSP problem. The same pattern is observed among two search algorithms of ACO and GWO such that ACO experienced 3 jumps after iteration 600 while GWO continues exploitation around the observed minima in iteration 300.

After the above experiments, we reexamined the performance of World in higher dimensions of 10, 20, 30, and 50 in addition to dimension 5. As shown in Table 8, the results with increasing the dimension change in a little weak result but still close to the first tests. It means that the algorithms follow the same pattern in different dimensions.

The combinatorial-optimization-based threat evaluation and jamming allocation (COTEJA) (You,Diao,and Gao, 2019) is the second discrete search problem in our experiments. As shown in Fig. 10, COTEJA explores finding a network of radars to discover aircrafts. This problem requires multi-party interaction managements between radars, aircrafts, and radar-aircrafts. To this end, we examined the World's performance to allocate the radars in this multi-objective, NP-hard search problem.

As reported in Table 9, although World did not have the best start, which is reported in the worst column, it reached to the lowest cost in contrast to the other search problems. Our proposed algorithm achieved an overall successful search with the value of 0.67 in average of all iterations. As World has the second largest standard deviation, we may observe several jumps as a sign of exploration trends in consecutive results provided in all iterations of World.

As shown in Fig. 11, World outcomes exceeded the other search algorithms' results after iteration 200. In the earlier iterations, World

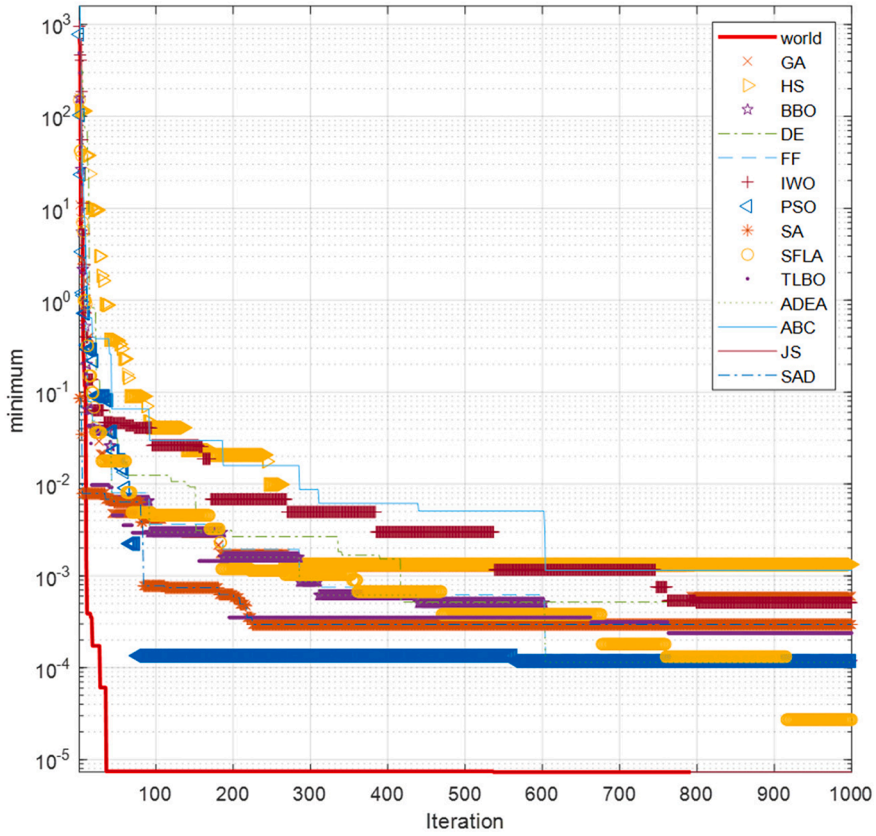


Fig. 5. The Graph evaluation results of Continuous algorithms for Quartic function.

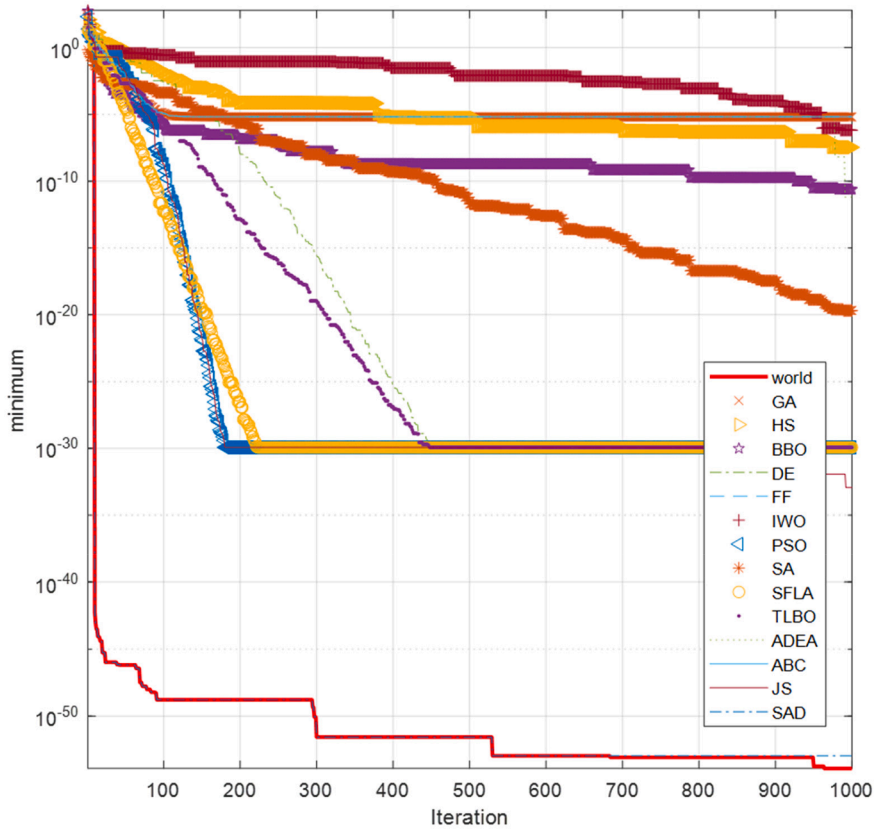
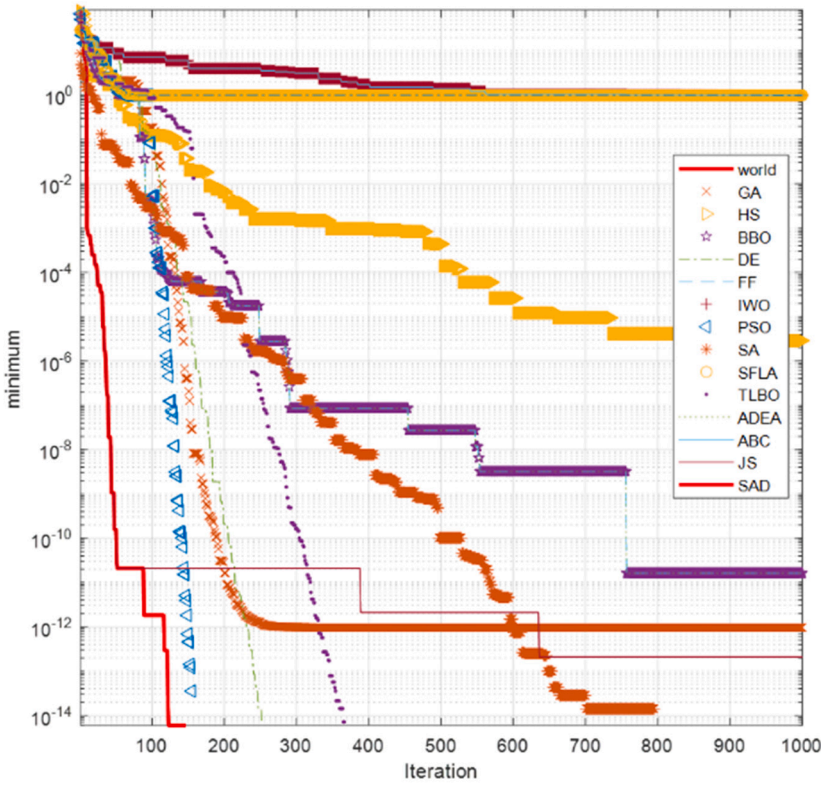
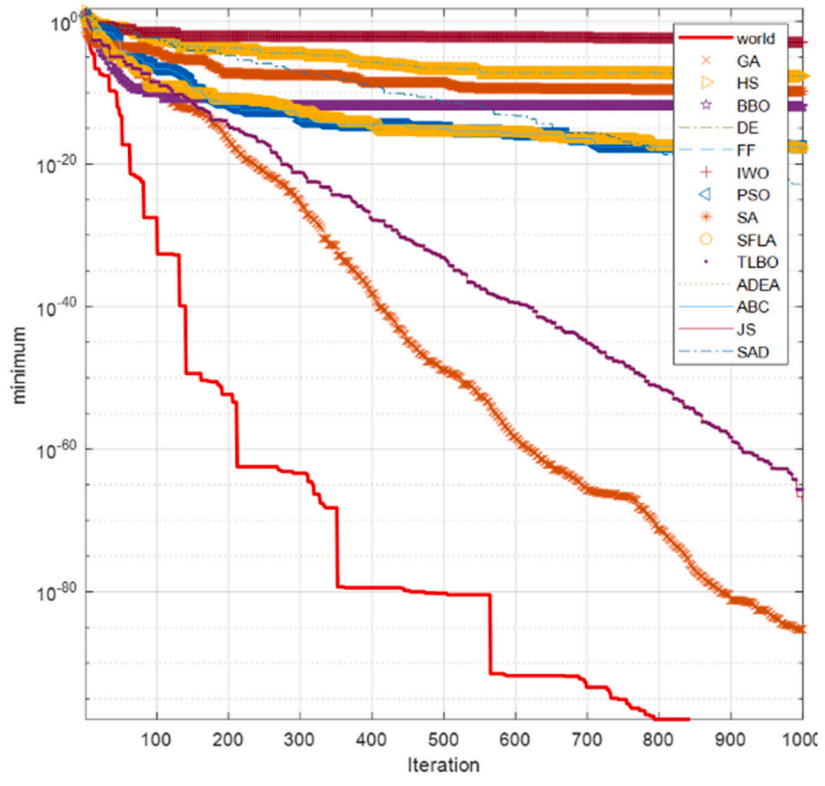


Fig. 6. The Graph evaluation results of Continuous algorithms over the Qing function.



A



B

Fig. 7. The Graph evaluation results of Continuous algorithms over (A) Rastrigin function and (B) Xin-She Yang function.

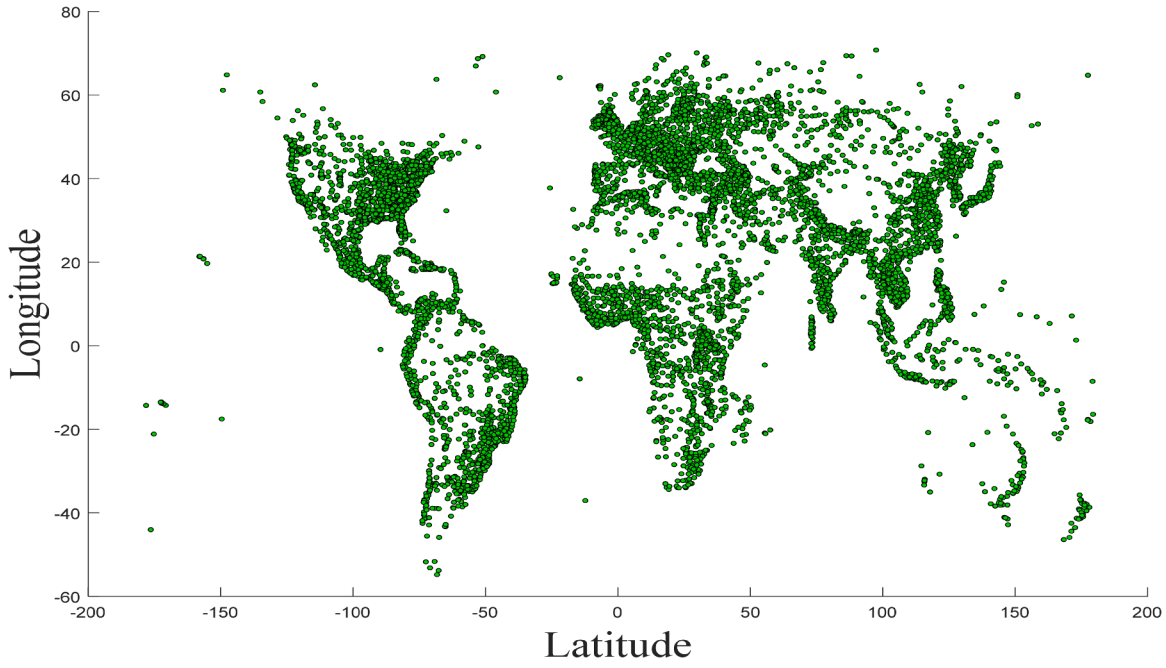


Fig. 8. Coordinates of 10,000 real cities in degrees for big transfer salesman problem (TSP).

Table 7
The evaluation results of discrete algorithms over the 10,000 cities Travelling salesman problem (TSP).

Worst	Std	Best	Avg	Algorithms
1.59e + 06	1.45e + 05	5.00e + 05	5.96e + 05	ACO
1.01e + 06	5.50e + 04	7.38e + 05	8.22e + 05	GA
1.39e + 06	**1.07e + 05**	5.79e + 05	6.14e + 05	GWO
1.06e + 06	5.21e + 04	8.46e + 05	9.02e + 05	ICA
1.06e + 06	8.45e + 04	7.68e + 05	8.99e + 05	SA
1.63e + 06	1.20e + 05	7.12e + 05	7.44e + 05	Tabu
1.62e + 06	1.88e + 05	**4.33e + 05**	**5.71e + 05**	**World**

experienced big jumps while it followed more smooth local exploitations in the latter iterations. The RL method empowers World to examine the search space by flexibility on initial explorations. This trend shifted to enhance the best observed minima in the latter iterations. As it followed a good trend of exploration at the beginning, its latter exploration rapidly increased the performance. This unique trend did not follow in the other search algorithms of Tabu, GA, and ACO. The only other algorithm that followed more improvements on the latter iterations is GWO, which suffers from weak explorations in the beginning iterations.

We extended our experiments to multi-dimensional analysis of COTEJA with higher dimensions of 10, 20, 30, and 50. The performance for the best result in World algorithm 0.1 is increased. The results in higher dimensions are similar to dimension 5 as shown in Table 10. It means that the algorithms follow the same pattern in different dimensions. The results are reported in the table below:

We need to use search optimization on real, continuous, state spaces as well as discrete ones. Welded beam design is one of the real, multi-objectives, continuous optimization problems. Welding steels surfaces provide a robust building material. Finding a low-cost production to resist on defections shapes a pareto front of solutions that need to be explored. As shown in Fig. 12, the beam with length I joins the weld with height h and length i to a member with dimensions t, b, and L to resist on a load of F. The mathematical formulas of this multi-objective minimization are detailed on (C. A. C. Coello 2002).

The results of search algorithms on welded beam design optimization are reported in Table 11 based on four evaluation criteria. The best column reports the successfulness of World in finding 1.67 cost units. This value is lower than the best value of sinusoidal chaos great deluge algorithm (GDA) (Baykasoglu, 2012) through the optimization. GDA average and worst values are equal to 1.72 while its standard deviation is equal to 0. Therefore, this search algorithm in contrast to World got stuck on a local minimum and did not manage to perform an effective exploration. The minimum average and standard deviation of *meta*-heuristics during different iterations belong to HS and SA algorithms, respectively. World may deploy some effective explorations that shifted the results in varied iterations and were represented by the standard deviation of 0.72. As this exploration behavior led to the best minimum value, the solution shifted during different iterations were controlled as it went to near-optimal solution in later iterations.

Fig. 13 brings light to the finding that all search algorithms followed a sharp decreasing trend in the process of finding the optimal minimum values in iterations 1 to 100. The unique behavior of exploration in 700th iteration distinguished the performance of World from the rest of optimization search algorithms. PSO had a small jump in the searching results in 140th iteration and continued exploitation around this data point. The rest of the algorithms did not perform a noticeable adjustment on the results in the latter iterations.

After describing the test with dimension 5, the experiment has been implemented with 10, 20, 30, and 50 dimensions to evaluate the

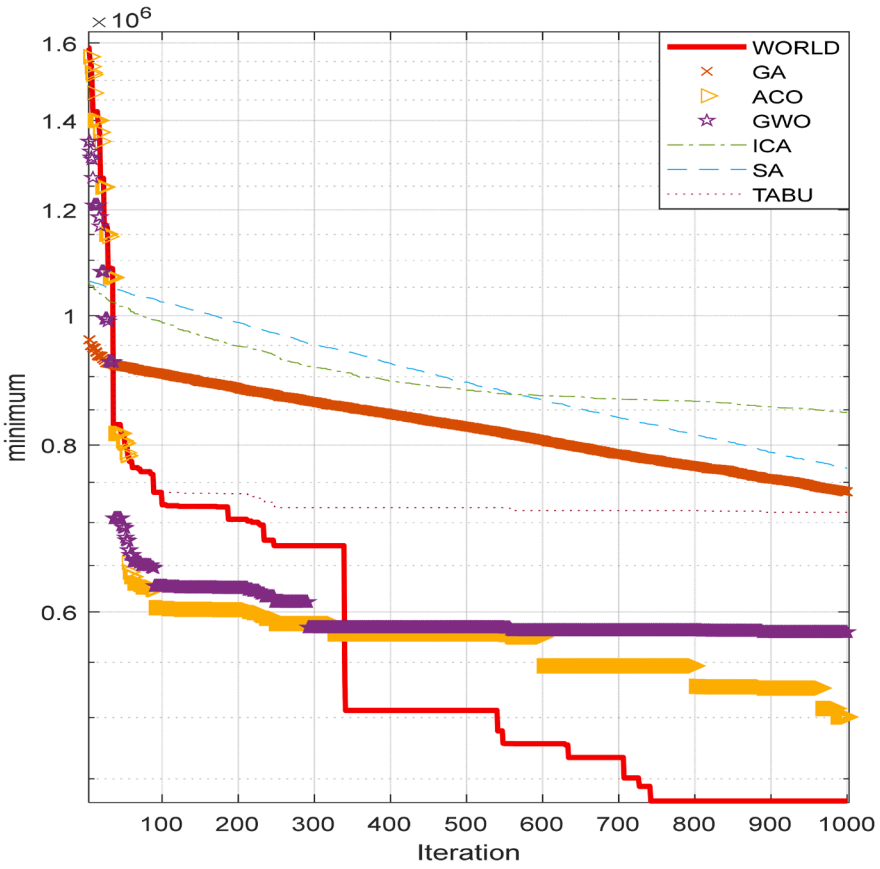


Fig. 9. The Graph evaluation results of discrete algorithms over the 10,000 cities Travelling salesman problem (TSP).

Table 8
The evaluation results of discrete algorithms over the 10,000 cities Travelling salesman problem (TSP)in the dimensions of 10, 20, 30, and 50.

Algorithms	Criteria	10	20	30	50
World	Best	4.32e + 05	4.56e + 05	4.89e + 05	4.81e + 05
	STD	1.95e + 05	2.13e + 05	1.63e + 05	2.05e + 05
	Mean	5.23e + 05	5.82e + 05	5.78e + 05	5.96e + 05
ACO	Best	5.12e + 05	5.18e + 05	5.21e + 05	5.23e + 05
	STD	1.51e + 05	1.57e + 05	1.58e + 05	1.59e + 05
	Mean	5.97e + 05	5.98e + 05	5.98e + 05	6.00e + 05
GA	Best	7.38e + 05	7.39e + 05	7.40e + 05	7.45e + 05
	STD	5.51e + 04	5.50e + 04	5.51e + 04	5.53e + 04
	Mean	8.22e + 05	8.23e + 05	8.23e + 05	8.25e + 05
GWO	Best	5.79e + 05	5.80e + 05	5.83e + 05	5.84e + 05
	STD	1.08e + 05	1.08e + 05	1.09e + 05	1.09e + 05
	Mean	6.14e + 05	6.15e + 05	6.16e + 05	6.17e + 05
ICA	Best	8.46e + 05	8.47e + 05	8.49e + 05	8.50e + 05
	STD	5.21e + 04	5.22e + 04	5.24e + 04	5.24e + 04
	Mean	9.02e + 05	9.03e + 05	9.05e + 05	9.05e + 05
SA	Best	7.69e + 05	7.69e + 05	7.70e + 05	7.72e + 05
	STD	8.45e + 04	8.45e + 04	8.46e + 04	8.47e + 04
	Mean	8.99e + 05	8.99e + 05	8.99e + 05	9.00e + 05
TABU	Best	7.12e + 05	7.13e + 05	7.14e + 05	7.16e + 05
	STD	1.20e + 05	1.20e + 05	1.21e + 05	1.22e + 05
	Mean	7.44e + 05	7.45e + 05	7.45e + 05	7.46e + 05

performance of World algorithm. Based on the evaluation results, the general trends are similar to dimension 5, but the results in dimension 5 are better than other dimensions. Table 12 reports the detailed results for welded beam design problem.

A speed reducer is used vastly in turbine generators to modify the torque and speed between the motor and the driven load. Designing a speed reducer is an optimization problem to keep the weight of speed reducer as low as possible. As shown in Fig. 14, this design includes 7 parameters of width x_1, the model of teeth x_2, teeth number in pinion x_3, length of input shaft between bearing x_4, length of output shaft x_5, diameter of input shaft x_6, and diameter of output shaft x_7.

As detailed in Table 13, World solved the speed reducer optimization by the best result of 2.04e + 03. Our algorithm similar to the previous case study followed an effective exploration alongside of exploitation behavior such that it fled from getting stuck in a local minimum. However, GDA did not manage to perform enough explorations as shown by its standard deviation and average of solution values, which are the lowest among all search algorithms' results. World's solutions of different iterations have the standard deviation of 1.21e + 03, which is the 6th lowest value among the 10 algorithms. These observations suggest effective behavior for exploration and exploitation of our proposed algorithm.

Fig. 15 confirms the conclusion mentioned above by showing several jumps, like those in iterations 20 and 540. Our proposed algorithm constantly improved the minima during iterations while HS algorithm could not enhance its outcomes after iteration 50. The rest of algorithms' search in + 100 iterations led to a small decreasing in the outcomes.

Like the previous case studies, in this section the multi dimension test of speed reducer problem is presented. The World algorithm is performing better than the other search algorithms. As Table 14 demonstrates, the optimization in lower dimension provides better solution rather than higher dimensions. In this problem, the results in each

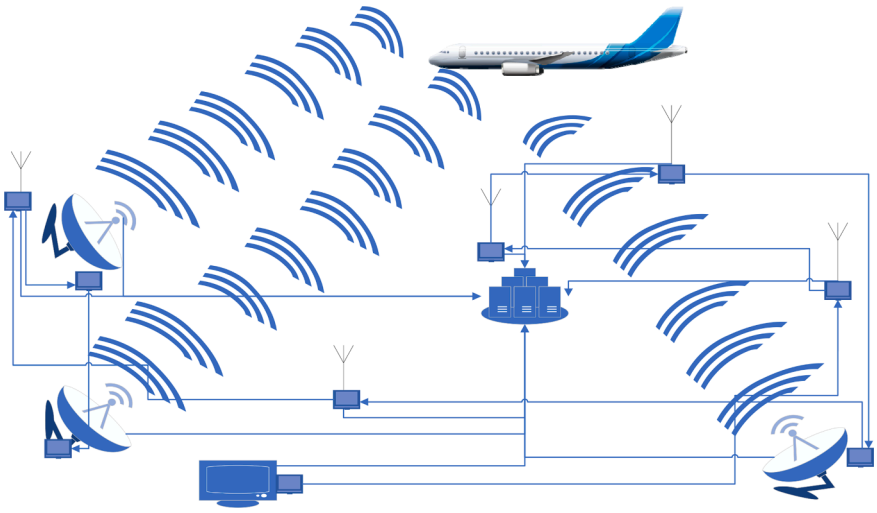


Fig. 10. Schematic view of combinatorial-optimization-based threat evaluation and jamming allocation (COTEJA) problem.

Table 9
The evaluation results of discrete algorithms over combinatorial-optimization-based threat evaluation and jamming allocation (COTEJA) problem.

Worst	Std	Best	Avg	Algorithm
1.97	0.11	0.85	0.91	ACO
1.41	0.10	0.65	0.72	GA
2.19	0.17	0.98	1.76	GWO
2.07	0.23	0.99	1.61	ICA
2.02	0.13	0.91	1.12	SA
1.07	**0.04**	0.70	0.74	Tabu
1.74	0.17	**0.59**	**0.67**	World

dimension and the best result are very similar. The best results belong to World algorithm.

5. Wilcoxon signed-rank test

All the experiments of this article compare the proposed algorithm and the other well-known search algorithms in a fair situation with identical hyper-parameters like population size. To investigate the difference between the outcomes of World and the rest of the algorithms, we used a non-parametric statistical analysis that enables us to compare all the algorithm results with the same datasets regardless of the distribution of data. Specifically, we used Wilcoxon signed-rank test (Sheskin, 2003). Wilcoxon signed-rank test rejects the null hypothesis if the difference between the World results, as compared to the others on the same dataset, are under 5 %. It means the algorithm results are statistically different from the other algorithms' outcomes. Table 15 reports the Wilcoxon signed-rank test results. Per the represented outcomes, World algorithm provides better and more distinguishable outcomes in comparison with the to all the state-of-the-art algorithms.

5.1. Discussion

Hyper heuristic algorithms are a group of optimization methods that combine heuristic algorithms with machine learning. These algorithms are used in complex and optimization problems to find better and faster

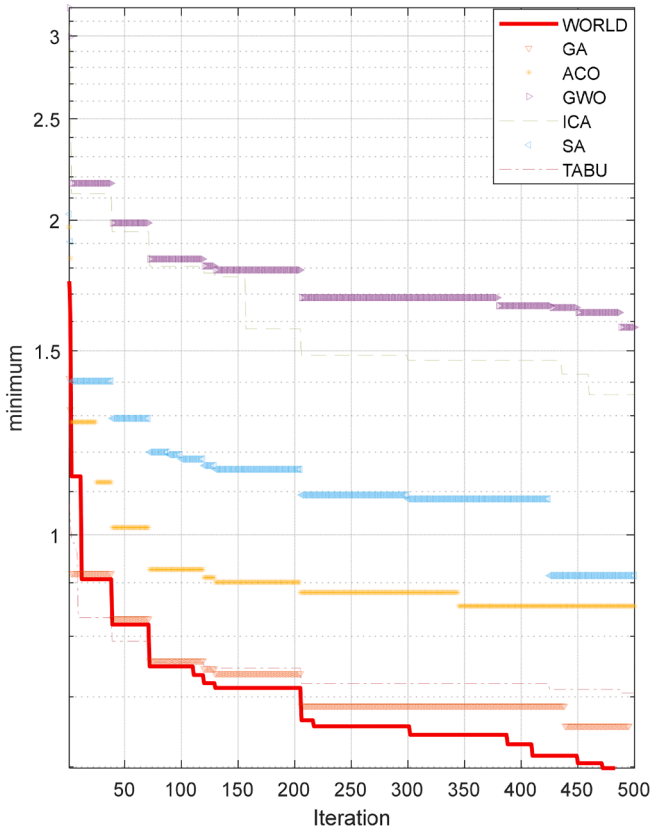


Fig. 11. The Graph evaluation results of discrete algorithms over the Combinatorial-Optimization-Based Threat Evaluation and Jamming Allocation (COTEJA) problem.

solutions. Meta-heuristics are approximate algorithms used to solve complex problems based on experience, simple rules, or inspiration from nature. These algorithms usually seek near-optimal solutions but need

Table 10
The evaluation results of discrete algorithms over combinatorial-optimization-based threat evaluation and jamming allocation (COTEJA) problem in the dimensions of 10,20, 30, and 50.

Algorithms	Criteria	10	20	30	50
World	Best	0.59	0.61	0.60	0.63
	STD	0.18	0.18	0.18	0.19
	Mean	0.69	0.67	0.71	0.73
ACO	Best	0.85	0.86	0.86	0.89
	STD	0.11	0.13	0.13	0.14
	Mean	0.91	0.92	0.92	0.95
GA	Best	0.65	0.66	0.69	0.71
	STD	0.10	0.11	0.14	0.18
	Mean	0.72	0.73	0.75	0.79
GWO	Best	0.98	0.99	1.05	1.15
	STD	0.17	0.18	0.21	0.26
	Mean	1.76	1.78	1.79	1.85
ICA	Best	0.99	1.06	1.04	1.16
	STD	0.23	0.21	0.20	0.28
	Mean	1.61	1.87	1.74	1.84
SA	Best	0.94	0.95	1.01	1.03
	STD	0.15	0.15	0.18	0.22
	Mean	1.65	1.71	1.73	1.81
TABU	Best	0.68	0.69	0.71	0.73
	STD	0.12	0.12	0.15	0.19
	Mean	0.73	0.73	0.77	0.81

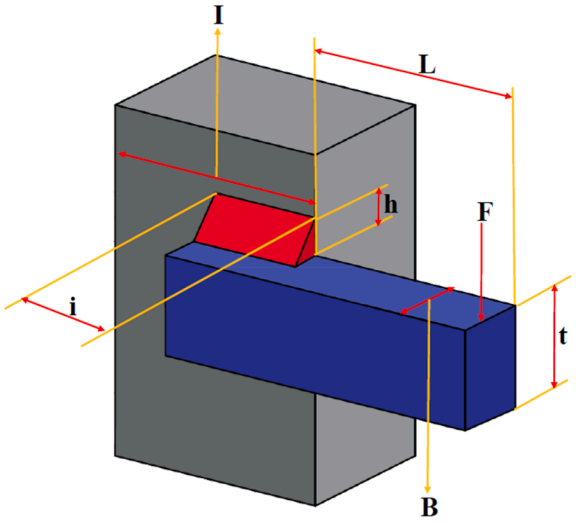


Fig. 12. Schematic view of Welded beam design.

more power to find the best solution. If Meta-heuristics algorithms are combined with machine learning, their performance and power will increase.

Machine learning allows optimization of *meta*-heuristic algorithms by extracting information and patterns in data and previous experiences.

Table 11
The evaluation results of Continuous algorithms over Welded beam design problem.

Algorithm	Avg	Best	std	worst
ABC	2.16	1.85	0.87	18.01
ADEA	2.74	1.87	0.81	15.17
BBO	1.95	1.91	0.46	10.63
DE	2.09	1.97	0.72	9.16
FF	2.05	1.93	0.98	18.02
GA	2.05	1.94	0.98	18.11
HS	1.88	1.77	0.63	15.42
IWO	2.60	2.44	0.87	21.34
jSO	2.16	1.76	0.75	17.19
PSO	2.16	1.94	0.91	19.33
SA	2.45	2.44	0.02	3.10
SADE	2.71	1.69	0.89	16.03
SFLA	2.53	2.44	0.69	10.74
TLBO	2.01	1.94	0.82	16.91
world	2.15	**1.67**	0.72	16.23
GDA- chaotic sinusoidal map [90]	1.72	1.72	**0.00**	**1.72**

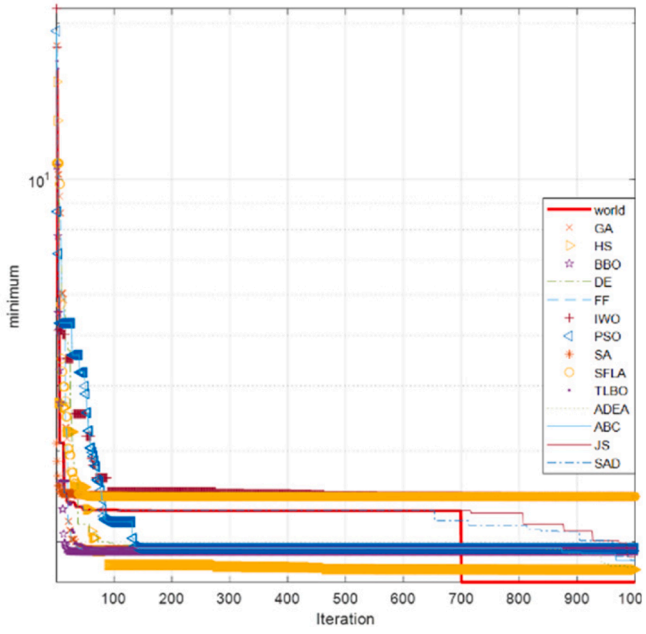


Fig. 13. The Graph evaluation results of Continuous algorithms over the Welded beam design problem.

With machine learning techniques, hyper heuristic algorithms can use their experience and knowledge in each optimization step and gradually approach better and more optimal solutions. These algorithms may use supervised learning, unsupervised learning, or a combination of these strategies to improve their performance. As a result, hyper heuristic algorithms can achieve better and faster solutions to complex problems by combining the powers of heuristic algorithms and the power of interacting with data and machine learning.

With all the interpretations in this subsection, we want to give more importance to future work. In general, the combination of reinforcement learning and deep learning is used to improve the performance of artificial intelligence systems. In this combination, the power of reinforcement learning is used to boost and improve the performance of hyper-

Table 12
The evaluation results of Continuous algorithms over Welded beam design problemin the dimensions of 10, 20, 30, and 50.

Algorithms	Criteria	10	20	30	50
World	Best	1.67	1.69	1.68	1.72
	STD	1.21e + 03	1.26e + 03	1.29e + 03	1.22e + 03
	Mean	2.17	2.18	2.17	2.19
BBO	Best	1.92	1.93	1.93	1.94
	STD	0.47	0.48	0.48	0.49
	Mean	1.95	1.96	1.96	1.96
DE	Best	1.97	1.99	1.99	2.01
	STD	0.73	0.74	0.75	0.78
	Mean	2.09	2.10	2.10	2.12
FA	Best	1.94	1.95	1.95	1.98
	STD	0.98	0.99	0.99	1.01
	Mean	2.05	2.07	2.07	2.08
GA	Best	1.95	1.97	1.98	1.99
	STD	0.99	0.99	0.99	1.01
	Mean	2.06	2.07	2.08	2.09
HS	Best	1.77	1.79	1.82	1.88
	STD	0.63	0.65	0.67	0.72
	Mean	1.88	1.89	1.91	1.93
IWO	Best	2.45	2.48	2.49	2.53
	STD	0.88	0.91	0.91	0.93
	Mean	2.61	2.63	2.64	2.65
PSO	Best	1.95	1.98	2.05	2.12
	STD	0.91	0.96	0.98	1.05
	Mean	2.16	2.18	2.19	2.26
SA	Best	2.45	2.49	2.52	2.58
	STD	0.03	0.08	0.11	0.16
	Mean	2.45	2.48	2.52	2.58
SFLA	Best	2.45	2.52	2.58	2.69
	STD	0.69	0.73	0.78	0.84
	Mean	2.53	2.55	2.59	2.63
TLBO	Best	1.95	1.98	1.99	2.05
	STD	0.82	0.85	0.83	0.88
	Mean	2.01	2.03	2.05	2.06
ADEA	Best	1.88	1.93	1.96	1.99
	STD	0.82	0.85	0.87	0.88
	Mean	2.74	2.75	2.78	2.82
ABC	Best	1.88	1.92	1.98	2.07
	STD	0.89	0.91	0.94	0.98
	Mean	2.18	2.23	2.26	2.27
JS	Best	1.77	1.84	1.92	1.97
	STD	0.76	0.82	0.85	0.88
	Mean	2.17	2.19	2.21	2.24
SADE	Best	1.70	1.75	1.76	1.82
	STD	0.92	0.94	0.95	0.98
	Mean	2.71	2.73	2.74	2.79

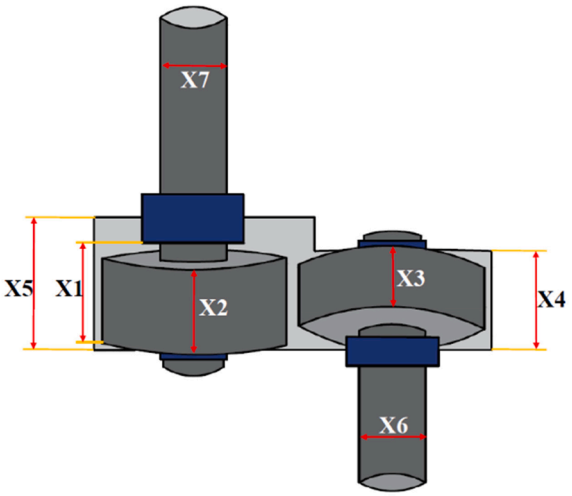


Fig. 14. Schematic view of Speed reducer Problem.

Table 13
The evaluation results of continuous algorithms over Speed reducer problem.

Algorithm	Avg	Best	std	worst
ABC	3.61e + 03	2.07e + 03	1.35e + 03	3.10e + 03
ADEA	3.31e + 03	2.52e + 03	1.89e + 03	3.12e + 04
BBO	3.33e + 03	3.13e + 03	798.51	1.81e + 04
DE	3.58e + 03	3.36e + 03	1.24e + 03	1.56e + 04
FF	3.49e + 03	3.00e + 03	1.67e + 03	3.07e + 04
GA	3.24e + 03	3.07e + 03	1.55e + 03	2.85e + 04
HS	3.20e + 03	2.63e + 03	1.07e + 03	2.63e + 04
IWO	3.43e + 03	3.65e + 03	1.48e + 03	3.64e + 04
jSO	3.45e + 03	3.05e + 03	1.45e + 03	3.08e + 04
PSO	4.18e + 03	3.39e + 03	4.15 e + 03	5.29e + 03
SA	4.06e + 03	3.37e + 03	1.11e + 03	1.72e + 04
SADE	2.01e + 03	2.70e + 03	0.82e + 03	16.94e + 03
SFLA	3.52e + 03	2.04e + 03	1.21e + 03	2.73e + 03
TLBO	3.61e + 03	2.07e + 03	1.35e + 03	3.10e + 03
world	3.31e + 03	**2.52e + 03**	**1.89e + 03**	3.12e + 04
GDA- ICMIC chaotic map [90]	**3.09e + 03**	24.48	3.00e + 03	**3.03e + 03**

heuristic algorithms. Reinforcement learning is used similarly to machine learning but concentrates on finding agents actively in the interactive environment. In reinforcement learning, an agent performs actions according to the state of the environment and learns to improve its performance based on the reward (reinforcement) or punishment it receives from the environment.

Furthermore, deep learning is a branch of machine learning that focuses on processing structured data, extensive and complex data.

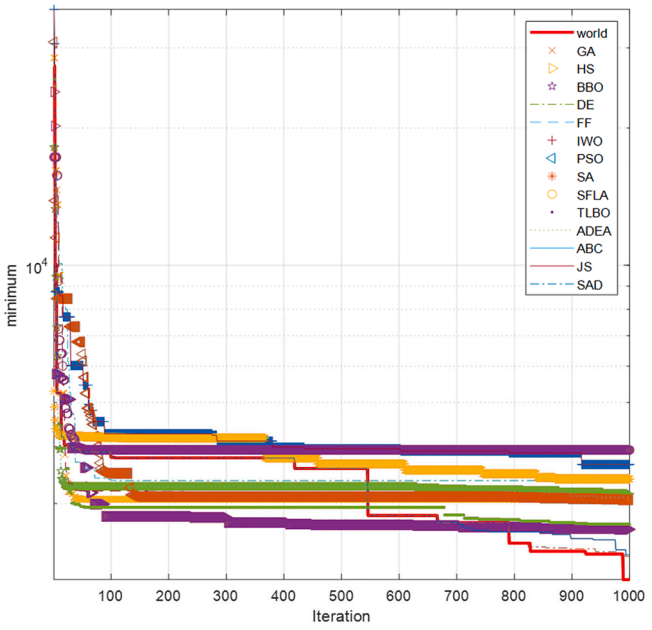


Fig. 15. The Graph evaluation results of continuous algorithms over the Speed reducer problem.

Using deep neural networks, conceptual information, and hidden features are extracted from the data, increasing the accuracy and performance of prediction and decision-making. Combining these two approaches can be chained, meaning that data features are extracted using deep learning, and then reinforcement learning algorithms are used to make optimal decisions in the environment. Again, hyper-heuristic algorithms can improve performance and find optimal points in the combination process. As a result, this combination can lead to algorithms that can extract hidden information and essential features using deep learning and improve their performance by using reinforcement learning and reach optimal points in an ample search space.

Among all the mentioned applications, one of the essential problems with which hyper-heuristics can be combined are problems such as Fast autoregressive tensor decomposition for online real-time traffic flow prediction (Xu et al., 2023). The benefit of hyper-heuristic algorithms, such as the world algorithm presented in this research, is significant in setting the parameters of prediction methods (Zhong et al., 2023). Therefore, by combining artificial intelligence methods, it is possible to increase work accuracy, speed, and correctness in any method. However, it should be noted that the proposed method has some limitations. Soft real-time problems are easily implemented with the World algorithm. However, due to automatic learning using reinforcement learning, the World algorithm cannot optimize hyperparameters in hard real-time problems. Finally, all the mentioned commodities are divided for future appointments in Table 16.

6. Conclusion & future work

We proposed a novel hyper-heuristic called World to take advantage of the exploration and exploitation of *meta*-heuristics in solving both continuous and discrete optimization problems. World employs an effective reinforcement learning to use an infinite *meta*-heuristic pool regardless of data type. Our proposed algorithm guarantees optimization convergence and provides flexible exchanging between exploration and exploitation. The experimental results over varied, artificial, real, discrete, and continuous optimization problems demonstrate the efficacy of our proposed algorithm.

This work can extend in multiple directions. We plan to examine World by a larger repository of *meta*-heuristics to optimize real search problems. In another direction, we will explore using deep reinforcement learning to dynamically update the selection procedure in each iteration. Ultimately, we will extend our model to handle dynamic state spaces, in which the outcomes of inputs are adjustable through time.

Table 14
The evaluation results of continuous algorithms over Speed reducer problem in dimensions of 10, 20, 30, and 50.

Algorithms	Criteria	10	20	30	50
World	Best	2.03e + 03	2.09e + 03	2.15e + 03	2.29e + 03
	STD	1.21e + 03	1.26e + 03	1.29e + 03	1.22e + 03
	Mean	3.15e + 03	3.13e + 03	3.31e + 03	3.41e + 03
BBO	Best	3.13e + 03	3.18e + 03	3.19e + 03	3.25e + 03
	STD	798.51	799.24	798.87	801.22
	Mean	3.33e + 03	3.35e + 03	3.36e + 03	3.38e + 03
DE	Best	3.36e + 03	3.37e + 03	3.38e + 03	3.40e + 03
	STD	1.24e + 03	1.24e + 03	1.25e + 03	1.25e + 03
	Mean	3.58e + 03	3.58e + 03	3.59e + 03	3.60e + 03
FA	Best	3.00e + 03	3.01e + 03	3.01e + 03	3.02e + 03
	STD	1.67e + 03	1.68e + 03	1.68e + 03	1.69e + 03
	Mean	3.49e + 03	3.49e + 03	3.49e + 03	3.49e + 03
GA	Best	3.07e + 03	3.08e + 03	3.08e + 03	3.10e + 03
	STD	1.55e + 03	1.56e + 03	1.56e + 03	1.59e + 03
	Mean	3.24e + 03	3.24e + 03	3.24e + 03	3.27e + 03
HS	Best	2.63e + 03	2.64e + 03	2.65e + 03	2.68e + 03
	STD	1.07e + 03	1.08e + 03	1.09e + 03	1.11e + 03
	Mean	3.20e + 03	3.20e + 03	3.21e + 03	3.23e + 03
IWO	Best	3.65e + 03	3.67e + 03	3.69e + 03	3.69e + 03
	STD	1.49e + 03	1.51e + 03	1.52e + 03	1.53e + 03
	Mean	3.44e + 03	3.46e + 03	3.46e + 03	3.46e + 03
PSO	Best	3.05e + 03	3.05e + 03	3.07e + 03	3.11e + 03
	STD	1.46e + 03	1.46e + 03	1.48e + 03	1.49e + 03
	Mean	3.46e + 03	3.46e + 03	3.47e + 03	3.51e + 03
SA	Best	3.39e + 03	3.42e + 03	3.43e + 03	3.45e + 03
	STD	4.16 e + 03	4.20 e + 03	4.21 e + 03	4.23 e + 03
	Mean	4.18e + 03	4.21e + 03	4.23e + 03	4.23e + 03
SFLA	Best	3.38e + 03	3.38e + 03	3.39e + 03	3.40e + 03
	STD	1.12e + 03	1.12e + 03	1.12e + 03	1.13e + 03
	Mean	4.07e + 03	4.07e + 03	4.07e + 03	4.07e + 03
TLBO	Best	2.70e + 03	2.70e + 03	2.73e + 03	2.76e + 03
	STD	0.82e + 03	0.82e + 03	0.84e + 03	0.85e + 03
	Mean	2.01e + 03	2.02e + 03	2.03e + 03	2.02e + 03
ADEA	Best	2.52e + 03	2.54e + 03	2.56e + 03	2.56e + 03
	STD	1.89e + 03	1.89e + 03	1.91e + 03	1.91e + 03
	Mean	3.31e + 03	3.33e + 03	3.33e + 03	3.34e + 03
ABC	Best	2.07e + 03	2.09e + 03	2.12e + 03	2.15e + 03
	STD	1.35e + 03	1.35e + 03	1.36e + 03	1.37e + 03
	Mean	3.62e + 03	3.62e + 03	3.63e + 03	3.63e + 03
JS	Best	2.07e + 03	2.07e + 03	2.08e + 03	2.11e + 03
	STD	1.31e + 03	1.31e + 03	1.31e + 03	1.32e + 03
	Mean	3.66e + 03	3.66e + 03	3.66e + 03	3.66e + 03
SADE	Best	2.56e + 03	2.58e + 03	2.61e + 03	2.63e + 03
	STD	1.89e + 03	1.89e + 03	1.91e + 03	1.92e + 03
	Mean	3.21e + 03	3.21e + 03	3.22e + 03	3.22e + 03

Table 15
Wilcoxon signed-rank test results.

Word vs.	Criteria	Better	Similar	Worse	*P-value*
BBO	Best	25	2	0	3.52e-04
	STD	25	0	2	0.56
	Mean	26	0	1	0.41
DE	Best	22	5	0	4.25e-03
	STD	25	0	2	0.36
	Mean	24	0	3	0.45
FA	Best	27	0	0	7.03e-04
	STD	27	0	0	0.31
	Mean	27	0	0	0.23
GA	Best	25	2	0	7.10e-03
	STD	26	0	1	0.37
	Mean	27	0	0	0.41
HS	Best	25	2	0	2.71e-03
	STD	27	0	0	0.37
	Mean	27	0	0	0.41
IWO	Best	25	2	0	7.70e-04
	STD	27	0	0	0.51
	Mean	27	0	0	0.48
PSO	Best	18	8	1	0.03
	STD	27	0	0	0.41
	Mean	27	0	0	0.47
SA	Best	23	4	0	6.78e-04
	STD	24	0	3	0.41
	Mean	24	0	3	0.39
SFLA	Best	23	4	0	4.21e-02
	STD	27	0	0	0.46
	Mean	27	0	0	0.37
TLBO	Best	21	6	0	4.82e-02
	STD	27	0	0	0.41
	Mean	27	0	0	0.44
ADEA	Best	10	15	2	0.34
	STD	25	0	2	0.83
	Mean	24	0	3	0.54
ABC	Best	7	17	3	0.55
	STD	25	0	3	0.63
	Mean	24	0	3	0.56
JS	Best	13	11	3	0.19
	STD	24	0	3	0.55
	Mean	23	0	4	0.45
SADE	Best	12	13	2	0.44
	STD	22	0	5	0.47
	Mean	21	0	6	0.56

Table 16
The combination of hyper heuristics with other methods in artificial intelligence in abstract form.

Type of composition	Advantage	Disadvantage	Limitations
Machine learning and hyper heuristics	Increase accuracy	–	Limited to classification and clustering algorithms
deep learning, reinforcement learning, and hyper heuristics	Increase accuracy and Automatic learning	Time-consuming problem	Delimited in the phase on explainability of neural networks
hyper heuristics and hyperparameter selection in real time classification	Increase work accuracy, speed, and correctness	lack of functionality in hard real time problems	Limited to classification and clustering algorithms

One of the studies that are hypothesized today is using explainable artificial intelligence methods. When there are limitations in the research field of complex *meta*-heuristic and hyper heuristic search algorithms, one can consider the construction of explainable algorithms for optimal search. Which part of the search space is searched, and for what reason a better result can be reached? For this reason, this field of research and operation is very faithful and has much work to do.

7. Authors' contributions

1. Dr. Arman Daliri participated in conceptualization, designing the algorithm, Implementation of the algorithm, visualization, experimental design, and experimental implementations, and writing the manuscript.
2. Mahmoud Alimoradi engaged in Implementation of the algorithm, experimental design, and experimental implementations.
3. Dr. Mahdieh Zabihimayvan was responsible for reviewing the manuscript draft.
4. Dr. Reza Sadeghi supervised the research project by assisting all co-authors in conceptualization, experimental design, writing, and reviewing the manuscript.

CRediT authorship contribution statement

Arman Daliri: Conceptualization, Visualization. **Mahmoud Alimoradi:** . **Mahdieh Zabihimayvan:** . **Reza Sadeghi:** .

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Availability of data and material: The data are available at https://tinyurl.com/WoldNew Code availability: The source code is available at https://tinyurl.com/WoldNew

Appendix 1

Standard functions used as artificial continuous problems

Function name	Formula	Fmin
Ackley	$\mathbf{f(x)} = \mathbf{f(x_1,\cdots,x_n)} = -\mathbf{a.exp}(-\mathbf{b}\sqrt{\frac{1}{\mathbf{n}}\sum_{\mathbf{i=1}}^{\mathbf{n}}\mathbf{x_i^2}}) - \mathbf{exp}(\frac{1}{\mathbf{n}}\sum_{\mathbf{i=1}}^{\mathbf{n}}\mathbf{cos(cx_i)}) + \mathbf{a} + \mathbf{exp}(1)$	0
Brown	$\mathbf{f(x)} = \sum_{\mathbf{i=1}}^{\mathbf{n-1}}(\mathbf{x_i^2})^{(\mathbf{x_{i+1}^2}+1)} + (\mathbf{x_{i+1}^2})^{(\mathbf{x_i^2}+1)}$	0
Exponential	$\mathbf{f(x)} = \mathbf{f(x_1,\cdots,x_n)} = -\mathbf{exp}(-0.5\sum_{\mathbf{i=1}}^{\mathbf{n}}\mathbf{x_i^2})$	-1
Griewank	$f(x) = f(x_1,\cdots,x_n) = 1 + \sum_{i=1}^{n}\frac{x_i^2}{4000} - \prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}})$	0
Periodic	$\mathbf{f(x)} = \mathbf{f(x_1,\cdots,x_n)} = 1 + \sum_{\mathbf{i=1}}^{\mathbf{n}}\mathbf{sin}^2(\mathbf{x_i}) - 0.1\mathbf{e}^{(\sum_{i=1}^{n}x_i^2)}$	0.9
Powell Sum	$f(x) = f(x_1,\cdots,x_n) = \sum_{i=1}^{n}\lvert x_i\rvert^{i+1}$	0
Qing	$f(x) = f(x_1,\cdots,x_n) = \sum_{i=1}^{n}(x^2 - i)^2$	0
Quartic	$\mathbf{f(x)} = \mathbf{f(x_1,\cdots,x_n)} = \sum_{\mathbf{i=1}}^{\mathbf{n}}\mathbf{ix_i^4} + \mathbf{random}[0,1)$	0
Rastrigin	$f(x,y) = 10n + \sum_{i=1}^{n}(x_i^2 - 10\cos(2\pi x_i))$	0
Ridge	$f(x) = x_1 + d(\sum_{i=2}^{n}x_i^2)^{\alpha}$	-10
Rosenbrock	$\mathbf{f(x,y)} = \sum_{\mathbf{i=1}}^{\mathbf{n}}[\mathbf{b}(\mathbf{x_{i+1}} - \mathbf{x_i^2})^2 + (\mathbf{a} - \mathbf{x_i})^2]$	0
Salomon	$f(x) = f(x_1,\cdots,x_n) = 1 - \cos(2\pi\sqrt{\sum_{i=1}^{D}x_i^2}) + 0.1\sqrt{\sum_{i=1}^{D}x_i^2}$	0
Schwefel 2.20	$f(x) = f(x_1,\cdots,x_n) = \sum_{i=1}^{n}\lvert x_i\rvert$	0
Schwefel 2.21	$f(x) = f(x_1,\cdots,x_n) = \max_{i=1,\cdots,n}\lvert x_i\rvert$	0
Schwefel 2.22	$f(x) = f(x_1,\cdots,x_n) = \sum_{i=1}^{n}\lvert x_i\rvert + \prod_{i=1}^{n}\lvert x_i\rvert$	0
Schwefel 2.23	$f(x) = f(x_1,\cdots,x_n) = \sum_{i=1}^{n}x_i^{10}$	0
shubert3	$f(x) = f(x_1,\cdots,x_n) = \sum_{i=1}^{n}\sum_{j=1}^{5}j\sin((j+1)xi + j)$	-101
Sphere	$f(x) = f(x_1,x_2,\cdots,x_n) = \sum_{i=1}^{n}x_i^2$	0
Styblinski-Tank	$\mathbf{f(x)} = \mathbf{f(x_1,x_2,\cdots,x_n)} = \frac{1}{2}\sum_{\mathbf{i=1}}^{\mathbf{n}}(\mathbf{x_i^4} - 16\mathbf{x_i^2} + 5\mathbf{x_i})$	-200
Sum Squares	$\mathbf{f(x)} = \mathbf{f(x_1,x_2,\cdots,x_n)} = \sum_{\mathbf{i=1}}^{\mathbf{n}}\mathbf{ix_i^2}$	0
Xin-She Yang	$f(x) = f(x_1,x_2,\cdots,x_n) = \sum_{i=1}^{n}\in_i\lvert x_i\rvert^i$	0
Xin-She Yang N. 2	$\mathbf{f(x)} = \mathbf{f(x_1,x_2,\cdots,x_n)} = (\sum_{\mathbf{i=1}}^{\mathbf{n}}\lvert\mathbf{x_i}\rvert)\mathbf{exp}(\sum_{\mathbf{i=1}}^{\mathbf{n}}\mathbf{sin}(\mathbf{x_i^2}))$	0
Xin-She Yang N. 3	$f(x) = f(x_1,x_2,\cdots,x_n) = \exp(-\sum_{i=1}^{n}\left(\frac{x_i}{\beta}\right)^{2m} - 2\exp(-\sum_{i=1}^{n}x_i^2)\prod_{i=1}^{n}\cos^2(x_i))$	0

(*continued on next page*)

(*continued*)

Function name	Formula	Fmin
Xin-She Yang N. 4	$f(x) = f(x_1, x_2, \cdots, x_n) = \left(\sum_{i=1}^{n} \sin^2(x_i) - e^{-\sum_{i=1}^{n} x_i^2}\right) e^{-\sum_{i=1}^{n} \sin^2 \sqrt{\lvert x_i \rvert}}$	−1
Zakharov	$f(x) = f(x_1, x_2, \cdots, x_n) = \sum_{i=1}^{n} x_i^2 + \left(\sum_{i=1}^{n} 0.5 i x_i\right)^2 + \left(\sum_{i=1}^{n} 0.5 i x_i\right)^4$	0

References

Alimoradi, M., Azgomi, H., & Asghari, A. (2022). Trees Social Relations Optimization Algorithm: A New Swarm-Based Metaheuristic Technique to Solve Continuous and Discrete Optimization Problems. *Mathematics and Computers in Simulation, 194*, 629–664.

Atashpaz-Gargari, E., & Lucas, C. (2007). Imperialist Competitive Algorithm: An Algorithm for Optimization Inspired by Imperialistic Competition. In *In 2007 IEEE Congress on Evolutionary Computation* (pp. 4661–5467). Ieee.

Bai, R., Burke, E. K., & Kendall, G. (2008). Heuristic, Meta-Heuristic and Hyper-Heuristic Approaches for Fresh Produce Inventory Control and Shelf Space Allocation. *Journal of the Operational Research Society, 59*(10), 1387–1397.

Baykasoglu, A. (2012). Design Optimization with Chaos Embedded Great Deluge Algorithm. *Applied Soft Computing, 12*(3), 1055–1067.

Blondel, V. D., & Tsitsiklis, J. N. (2000). A Survey of Computational Complexity Results in Systems and Control. *Automatica, 36*(9), 1249–1274.

Brest, J., Maučec, M. S., & Bošković, B. (2017). Single Objective Real-Parameter Optimization: Algorithm jSO. *In 2017 IEEE Congress on Evolutionary Computation (CEC), 1311–18*. IEEE.

Brest, J., Maučec, M. S., & Bošković, B. (2021). Self-Adaptive Differential Evolution Algorithm with Population Size Reduction for Single Objective Bound-Constrained Optimization: Algorithm J21. In *In 2021 IEEE Congress on Evolutionary Computation (CEC)* (pp. 817–824). IEEE.

Burke, E. K., Hyde, M. R., Kendall, G., Ochoa, G., Özcan, E., & Woodward, J. R. (2019). A Classification of Hyper-Heuristic Approaches: Revisited. In *Handbook of Metaheuristics* (pp. 453–477). Springer.

Coello, CA Coello, and M. Salazar Lechuga. 2002. "MOPSO: A Proposal for Multiple Objective Particle Swarm Optimization." In *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600)*, 2:1051–56. IEEE.

Coello, C. A., & Coello.. (1999). A Comprehensive Survey of Evolutionary-Based Multiobjective Optimization Techniques. *Knowledge and Information Systems, 1*(3), 269–308.

Coello, C. A., & Coello,. (2002). Theoretical and Numerical Constraint-Handling Techniques Used with Evolutionary Algorithms: A Survey of the State of the Art. *Computer Methods in Applied Mechanics and Engineering, 191*(11–12), 1245–1287.

Cruz-Duarte, J. M., Amaya, I., Ortiz-Bayliss, J. C., Conant-Pablos, S. E., Terashima-Marín, H., & Shi, Y. (2021). Hyper-Heuristics to Customise Metaheuristics for Continuous Optimisation. *Swarm and Evolutionary Computation, 100935*.

Cui, L., Li, G., Lin, Q., Chen, J., & Nan, L.u. (2016). Adaptive Differential Evolution Algorithm with Novel Mutation Strategies in Multiple Sub-Populations. *Computers & Operations Research, 67*, 155–173.

Daliri, A., Asghari, A., Azgomi, H., & Alimoradi, M. (2022). The Water Optimization Algorithm: A Novel Metaheuristic for Solving Optimization Problems. *Applied Intelligence*, 1–40.

Dorigo, M., Birattari, M., & Stutzle, T. (2006). Ant Colony Optimization. *IEEE Computational Intelligence Magazine, 1*(4), 28–39.

Duflo, Gabriel, Grégoire Danoy, El-Ghazali Talbi, and Pascal Bouvry. 2020. "Automated Design of Efficient Swarming Behaviours: A Q-Learning Hyper-Heuristic Approach." In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion*, 227–28.

Ge, J., Zaijun, W.u., Junjun, X.u., & Qinran, H.u. (2023). A Two-Stage Flow-Based Partition Framework for Unbalanced Distribution Networks. *CSEE Journal of Power and Energy Systems*. https://ieeexplore.ieee.org/abstract/document/10165674/.

Glover, F. (1989). Tabu Search—Part I. *ORSA Journal on Computing, 1*(3), 190–206.

Glover, F. (1990). Tabu Search—Part II. *ORSA Journal on Computing, 2*(1), 4–32.

Gratch, Jonathan, Steve Chien, and Gerald DeJong. 1993. "Learning Search Control Knowledge for Deep Space Network Scheduling." In *Proceedings of the Tenth International Conference on Machine Learning*, 135–42.

Gutjahr, W. J. (2007). Mathematical Runtime Analysis of ACO Algorithms: Survey on an Emerging Issue. *Swarm Intelligence, 1*(1), 59–79.

Hassan, Rania, Babak Cohanim, Olivier De Weck, and Gerhard Venter. 2005. "A Comparison of Particle Swarm Optimization and the Genetic Algorithm." In *46th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*, 1897.

Katajainen, J., & Träff, J. L. (1997). A Meticulous Analysis of Mergesort Programs. In *Italian Conference on Algorithms and Complexity* (pp. 217–228). Springer.

Kennedy, James, and Russell Eberhart. 1995. "Particle Swarm Optimization." In *Proceedings of ICNN'95-International Conference on Neural Networks*, 4:1942–48. IEEE.

Kirkpatrick, S., Daniel Gelatt, C., & Vecchi, M. P. (1983). Optimization by Simulated Annealing. *Science, 220*(4598), 671–680.

Koulinas, G., Kotsikas, L., & Anagnostopoulos, K. (2014). A Particle Swarm Optimization Based Hyper-Heuristic Algorithm for the Classic Resource Constrained Project Scheduling Problem. *Information Sciences, 277*, 680–693.

Kumar, S., Tejani, G. G., Pholdee, N., & Bureerat, S. (2022). Performance Enhancement of Meta-Heuristics through Random Mutation and Simulated Annealing-Based Selection for Concurrent Topology and Sizing Optimization of Truss Structures. *Soft Computing, 26*(12), 5661–5683.

Kumar, S., Tejani, G. G., Pholdee, N., Bureerat, S., & Mehta, P. (2021). Hybrid Heat Transfer Search and Passing Vehicle Search Optimizer for Multi-Objective Structural Optimization. *Knowledge-Based Systems, 212*, Article 106556.

Kunakote, T., Sabangban, N., Kumar, S., Tejani, G. G., Panagant, N., Pholdee, N., … Yildiz, A. R. (2022). Comparative Performance of Twelve Metaheuristics for Wind Farm Layout Optimisation. *Archives of Computational Methods in Engineering, 29*(1), 717–730.

Lin, J., Zhu, L., & Gao, K. (2020). A Genetic Programming Hyper-Heuristic Approach for the Multi-Skill Resource Constrained Project Scheduling Problem. *Expert Systems with Applications, 140*, Article 112915.

Maashi, M., Özcan, E., & Kendall, G. (2014). A Multi-Objective Hyper-Heuristic Based on Choice Function. *Expert Systems with Applications, 41*(9), 4475–4493.

MacQueen, James and others. 1967. "Some Methods for Classification and Analysis of Multivariate Observations." In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, 1:281–97. Oakland, CA, USA.

Mazyavkina, N., Sviridov, S., Ivanov, S., & Burnaev, E. (2021). Reinforcement Learning for Combinatorial Optimization: A Survey. *Computers & Operations Research, 105400*.

Mehrabian, A. R., & Lucas, C. (2006). A Novel Numerical Optimization Algorithm Inspired from Weed Colonization. *Ecological Informatics, 1*(4), 355–366.

Mirjalili, S. (2016). SCA: A Sine Cosine Algorithm for Solving Optimization Problems. *Knowledge-Based Systems, 96*, 120–133.

Mirjalili, S., & Lewis, A. (2016). The Whale Optimization Algorithm. *Advances in Engineering Software, 95*, 51–67.

Mirjalili, S., Mirjalili, S. M., & Lewis, A. (2014). Grey Wolf Optimizer. *Advances in Engineering Software, 69*, 46–61.

Neumann, F., & Witt, C. (2010). Combinatorial Optimization and Computational Complexity. *Bioinspired Computation in Combinatorial Optimization*, 9–19.

Qin, W., Zhuang, Z., Huang, Z., & Huang, H. (2021). A Novel Reinforcement Learning-Based Hyper-Heuristic for Heterogeneous Vehicle Routing Problem. *Computers & Industrial Engineering, 156*, Article 107252.

Ross, P., & Marfn-Blazquez, J. G. (2005). In *Constructive Hyper-Heuristics in Class Timetabling* (p. 1493). IEEE.

Russell, Stuart, and Peter Norvig. 2002. "Artificial Intelligence: A Modern Approach.".

Savsani, V. J., Tejani, G. G., & Patel, V. K. (2016a). Truss Topology Optimization with Static and Dynamic Constraints Using Modified Subpopulation Teaching–Learning-Based Optimization. *Engineering Optimization, 48*(11), 1990–2006.

Sheskin, D. J. (2003). *Handbook of Parametric and Nonparametric Statistical Procedures*. Chapman and Hall/CRC.

Shukla, A. K., Diwakar Tripathi, B., Reddy, R., & Chandramohan, D. (2019). A Study on Metaheuristics Approaches for Gene Selection in Microarray Data: Algorithms, Applications and Open Challenges. *Evolutionary Intelligence*, 1–21.

Sim, K., Hart, E., & Paechter, B. (2015). A Lifelong Learning Hyper-Heuristic Method for Bin Packing. *Evolutionary Computation, 23*(1), 37–67.

Srinivas, M., & Patnaik, L. M. (1994). Genetic Algorithms: A Survey. *Computer, 27*(6), 17–26.

Tejani, G. G., Pholdee, N., Bureerat, S., Prayogo, D., & Gandomi, A. H. (2019a). Structural Optimization Using Multi-Objective Modified Adaptive Symbiotic Organisms Search. *Expert Systems with Applications, 125*, 425–441.

Tejani, G. G., Savsani, V. J., Bureerat, S., Patel, V. K., & Savsani, P. (2019b). Topology Optimization of Truss Subjected to Static and Dynamic Constraints by Integrating Simulated Annealing into Passing Vehicle Search Algorithms. *Engineering with Computers, 35*(2), 499–517.

Tejani, G. G., Savsani, V. J., Patel, V. K., & Mirjalili, S. (2018). Truss Optimization with Natural Frequency Bounds Using Improved Symbiotic Organisms Search. *Knowledge-Based Systems, 143*, 162–178.

Terashima-Marín, H., Flores-Alvarez, E. J., & Ross, P. (2005). Hyper-Heuristics and Classifier Systems for Solving 2D-Regular Cutting Stock Problems. In *In Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation* (pp. 637–643).

Thomas, J., & Chaudhari, N. S. (2014). Design of Efficient Packing System Using Genetic Algorithm Based on Hyper Heuristic Approach. *Advances in Engineering Software, 73*, 45–52.

Thompson, R. F., & Welker, W. I. (1963). Role of Auditory Cortex in Reflex Head Orientation by Cats to Auditory Stimuli. *Journal of Comparative and Physiological Psychology, 56*(6), 996.

Tinós, R., Helsgaun, K., & Whitley, D. (2018). In *Efficient Recombination in the Lin-Kernighan-Helsgaun Traveling Salesman Heuristic* (pp. 95–107). Springer.

Trivedi, V., Varshney, P., & Ramteke, M. (2020). A Simplified Multi-Objective Particle Swarm Optimization Algorithm. *Swarm Intelligence, 14*(2), 83–116.

Walker, David J, and Ed Keedwell. 2016. "Multi-Objective Optimisation with a Sequence-Based Selection Hyper-Heuristic." In *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion*, 81–82.

Wang, Weijun, Stéphane Caro, Fouad Bennis, and Oscar Brito Augusto. 2012. "Toward the Use of Pareto Performance Solutions and Pareto Robustness Solutions for Multi-Objective Robust Optimization Problems." In *Engineering Systems Design and Analysis*, 44861:541–50. American Society of Mechanical Engineers.

Wolpert, D. H., & Macready, W. G. (1997). No Free Lunch Theorems for Optimization. *IEEE Transactions on Evolutionary Computation, 1*(1), 67–82.

"World Traveling Salesman Problem." n.d. Accessed October 4, 2020. http://www.math.uwaterloo.ca/tsp/world/index.html.

Xu, Z., Lv, Z., Chu, B., & Li, J. (2023). Fast Autoregressive Tensor Decomposition for Online Real-Time Traffic Flow Prediction. *Knowledge-Based Systems, 282*, Article 111125.

You, S., Diao, M., & Gao, L. (2019). Implementation of a Combinatorial-Optimisation-Based Threat Evaluation and Jamming Allocation System. *IET Radar, Sonar & Navigation, 13*(10), 1636–1645.

Yusoff, Y., Ngadiman, M. S., & Zain, A. M. (2011). Overview of NSGA-II for Optimizing Machining Process Parameters. *Procedia Engineering, 15*, 3978–3983.

Zhang, C., Zhao, Y., & Leng, L. (2020). A Hyper-Heuristic Algorithm for Time-Dependent Green Location Routing Problem with Time Windows. *IEEE Access, 8*, 83092–83104.

Zhong, R., Jun, Y.u., Zhang, C., & Munetomo, M. (2023). Surrogate Ensemble-Assisted Hyper-Heuristic Algorithm for Expensive Optimization Problems. *International Journal of Computational Intelligence Systems, 16*(1), 169. https://doi.org/10.1007/s44196-023-00346-y