Available online at www.sciencedirect.com
ScienceDirect
Mathematics and Computers in Simulation 194 (2022) 629–664
MATHEMATICS AND COMPUTERS IN SIMULATION
www.elsevier.com/locate/matcom

Original articles

Trees Social Relations Optimization Algorithm: A new Swarm-Based metaheuristic technique to solve continuous and discrete optimization problems

Mahmoud Alimoradi[a], Hossein Azgomi[b,*], Ali Asghari[a]

[a] *Department of Computer Engineering, Shafagh Institute of Higher Education, Tonekabon, Iran*
[b] *Department of Computer Engineering, Rasht Branch, Islamic Azad University, Rasht, Iran*

Received 16 April 2021; received in revised form 10 November 2021; accepted 9 December 2021
Available online 17 December 2021

Abstract

This paper presents a new metaheuristic algorithm called Trees Social Relations Optimization Algorithm (TSR). TSR inspired by the hierarchical and collective life of trees in the jungle. The main priority of the collective consciousness of the trees is the survival of the woods. The trees try to reduce the damage in various ways so that the forest can develop. Organizing trees, protecting young seedlings, and their communication mechanism create a complex structure based on swarm intelligence that is the inspiration for designing an algorithm to solve existing problems. In TSR, each answer considered as a tree and a set of solutions defined as a sub-jungle. Sub-jungles are interconnected and help each other to get the right answer. The use of parallel and synchronized sub-jungles with its dedicated operators will increase the accuracy and shorten the time to reach an acceptable response. The TSR algorithm can use in continuous and discrete problems and, therefore, can use in a wide range of issues. Numerous experiments on standard and various benchmarks, as well as some classic and new issues, show that our proposed algorithm provides appropriate and acceptable answers in both time and accuracy to some similar algorithms.

Keywords: Optimization; Metaheuristic algorithms; Continuous optimization; Discrete optimization; Trees Social Relations

1. Introduction

With the increasing growth of science in various fields, the need for metaheuristic algorithms is becoming more and more evident. The primary source of metaheuristic can be found in artificial intelligence and operation research [6,51]. Increasing the data in the problem makes it more challenging to solve. Problems for which there is no solution in polynomial time are called NP-HARD [92]. So far, no algorithm that can solve the definitive answer of NP-hard at the right time and reach the final solution has been introduced [56]. Optimization methods are an excellent way to solve these problems [36]. These methods can find optimal or near-optimal answers to severe problems in a short time [52].

* Corresponding author.
E-mail addresses: mahmoud.alimoradi@shafagh.ac.ir (M. Alimoradi), Azgomi@iaurasht.ac.ir (H. Azgomi), asghari_ali@aut.ac.ir (A. Asghari).

https://doi.org/10.1016/j.matcom.2021.12.010

Optimization methods have an essential role in the industry, science development, management, and problem-solving that can model in this field. Multidimensional, discontinuous models and noise-containing data cannot solve by traditional methods [118]. In a general division, optimization methods are divided into two categories, single-objective and multi-objective. Usually, in multi-objective methods, there is a trade-off between goals. In other words, by improving an objective, at least one more objective is likely to be weakened [41].

LSGO[1] refers to optimization issues that have a large number of variables and many decision-making conditions, many of the problems that exist today in various fields of engineering and planning can classify as LSGOs that require new solutions [82]. For example, inversion problems in biological systems and industrial design as a decision-making problem are among the most time-consuming and significant issues that fall into the category of very complex problems [48,82]. Metaheuristics are a type of algorithm used to solve a wide range of problems, including engineering, basic sciences, medicine, and other disciplines that are involved in solving their problems. Metaheuristics have more capabilities than heuristics, which is the most crucial distinguishing feature of avoiding falling into the local optimal [50]. The main problem with heuristic algorithms is local and non-optimal (weak) responses [40].

Historically, metaheuristic algorithms can be classified into classic and modern. The algorithms of the 1990s and earlier were classic and later called modern [69]. Also, metaheuristic algorithms are divided into two types, population-based and trajectory-based, based on the type of response they produce [24,38]. Implementing these algorithms and extending them to different problems from different categories of sciences is an essential advantage of this type of algorithm. If the right strategy and parameters are selected, these algorithms will achieve the desired answer. The value of this becomes clear when we know that NP-hard problems, as the number of parameters increases numerically, are accompanied by an exponential increase in response time, which even with the most powerful computers, it may take several years to obtain an answer. Metaheuristic algorithms try to solve this problem and are an excellent alternative to traditional techniques. Metaheuristic algorithms are the right choice for optimizing issues [113].

Four fundamental features make metaheuristic algorithms very popular. First, they have a simple idea and implementation. Second, they are flexible, meaning they can easily change shape for all Problems. Third, metaheuristic works like a black box, so just input and output are essential for metaheuristic. Fourth, they do not fall into the trap of local optimism [88]. The fourth case, which has the advantage of metaheuristic over heuristic, is due to the possibility of proper adjustment between two fundamental factors in them. Exploration or Diversification and Exploitation or Intensification are two critical parameters in metaheuristic [113]. The exploitation phase seeks the optimal answer in existing candidates, and the exploration phase seeks the answer in spaces that have not searched so far. Metaheuristic involves iterative and production processes that find the optimal solution through these two phases [18,19].

The metaheuristic power comes from abstracting the features of nature in the simplest possible way. It implements millions of years of nature tests and Fittest and adaptability to the problem conditions [19]. The primary source of inspiration for metaheuristic algorithms is nature. Evolutionary theory, the phenomena of physics, politics, and biology are among the things inspired by them that will explain in the next section. These algorithms follow anything that can turn into a rule, including music, agriculture, and even ray ideas [1,26,65,75].

Today, metaheuristics have been implemented and thriving in a wide range of sciences, including the implementation of the Whale algorithm for error control in wind generators [100], the interpretation of PPG signals for the interpretation of cardiovascular diseases [99], and the control of automated production with Manufacturer of fuzzy PID [97], store planning [16], CR-Based IoT [10] and many engineering applications [13,61,119,121].

Each of the metaheuristic algorithms is suitable for solving a specific set of problems, and each has its advantages and disadvantages. For example, noise tolerance can be one of the advantages that does not exist in everyone. According to the NFL[2] theory, it is not possible to solve all problems with a specific algorithm [116]. For example, evolutionary algorithms are most commonly used to solve global search optimization problems in non-convex [34]. In addition to new metaheuristic methods, their combination and taking advantage of each is considered [81,95,96]. This strategy has given rise to a variety of algorithms. The emergence of new issues and the performance of new metaheuristic features highlight the importance of designing new algorithms.

[1] Large-scale Global optimization.
[2] No Free lunch.

According to the NFL theory, to get out of the future and solve today's problems and meet the needs, it is necessary to follow the example of nature and reach for new metaheuristics. In this paper, a new metaheuristic algorithm called the Trees social Relations Optimization Algorithm (TSR) is introduced, which is inspired by the social life and relationships of trees. The hierarchical nature of tree life, attention to other trees, how they relate to each other, and the structure of the sub-jungle, attention to young seedlings, and efforts to survive all the trees are the main features of the jungle. The collection of trees under the leadership of a particular tree called the mother tries to balance and maintain order in the jungle. The TSR algorithm is inspired by the collective life of trees in the jungle, with these features designed operators tailored to the behavior of the trees. In this algorithm, each tree represents an answer, and the jungle represents all possible answers. For parallelization, the division of the jungle into sub-jungles has used. The sub-jungles operate in parallel and exchange their excellent achievements to achieve Global optimization. The nature of the jungle works in parallel, and the mothers interact with each other. Of course, sub-jungle has not used to observe the principle of equality in comparing algorithms in implementations and comparisons of this article. TSR algorithm has important features that distinguish it from other similar methods. First, our proposed method works well in both discrete and continuous problem areas, Compared to other similar methods. Second, TSR algorithm has inherent parallelization capability and the tasks of each sub-jungle can be delegated to assigned processor core. The third feature of this algorithm is its better performance in large-scale problems. In fact, the bigger the problem, the better the performance of TSR, compared to competitors. The following are list of the main features of our algorithm:

- A new metaheuristic algorithm called TSR inspired by the social behavior of jungle trees
- Ability to parallelize by dividing the problem into smaller sub-jungles
- Connect sub-jungles to exchange experiences
- Ability to solve discrete and continuous problems using the proposed algorithm
- Introducing new operators to increase the convergence speed and achieve more appropriate answers
- Evaluating the proposed algorithm with classic and modern benchmarks and compare it with some standard techniques
- The proposed method is more efficient than some related and compared methods, especially when the parallelization feature is deployed
- Although the proposed method has been evaluated on various problems, its performance is more efficient on large-scale issues

Organizing the other sections of the article is as follows:

Section 2 contains a comprehensive overview of metaheuristic algorithms defined in four steps. Section 3 describes the basics of the TSR algorithm and describes the details of this algorithm completely. In Section 4, the evaluation of the proposed algorithm performed using classic and new standard benchmarks and the performance of the proposed method compared with some standard algorithms. Section 5 also concludes the work.

2. Literature review

The world around us is a source of inspiration for the production of metaheuristic algorithms. Nature and biology are some of these sources. In addition to biology, the laws of nature and physics, the cognitive and behavioral sciences of man, and the laws of sport and politics, the collective life of animals and plants, are the sources of inspiration for the production of metaheuristic algorithms. In all these cases, there is intelligence to solve problems and prevent ruin. In general, metaheuristic algorithms can be divided into four broad categories. The first category is Evolutionary-Based Algorithms. The second category is called Physics-Based algorithms. The third category is Human-Based algorithms, and finally, the fourth category is Swarm-Based algorithms. Fig. 1 provides a general classification of metaheuristic algorithms. Tables 1–4 show the four categories of related papers, i.e.: Evolutionary-Based, Physics-Based, Human-Based, and Swarm-Based algorithms in more detail.

Evolutionary-Based Algorithms include algorithms based on Charles Darwin's theory of evolution. These algorithms based on the generation of successive generations. In most cases, the first generation emerges by chance, and subsequent generations emerge from the process of parent integration and the birth of new offspring. Genetic Algorithm is the first metaheuristic algorithm and belongs in this category was proposed in 1960 by John Holland. This algorithm uses two principal operators, crossover and mutation, to generate new responses. In most cases, the population size of each generation is constant [12]. Other algorithms inspired by this category include

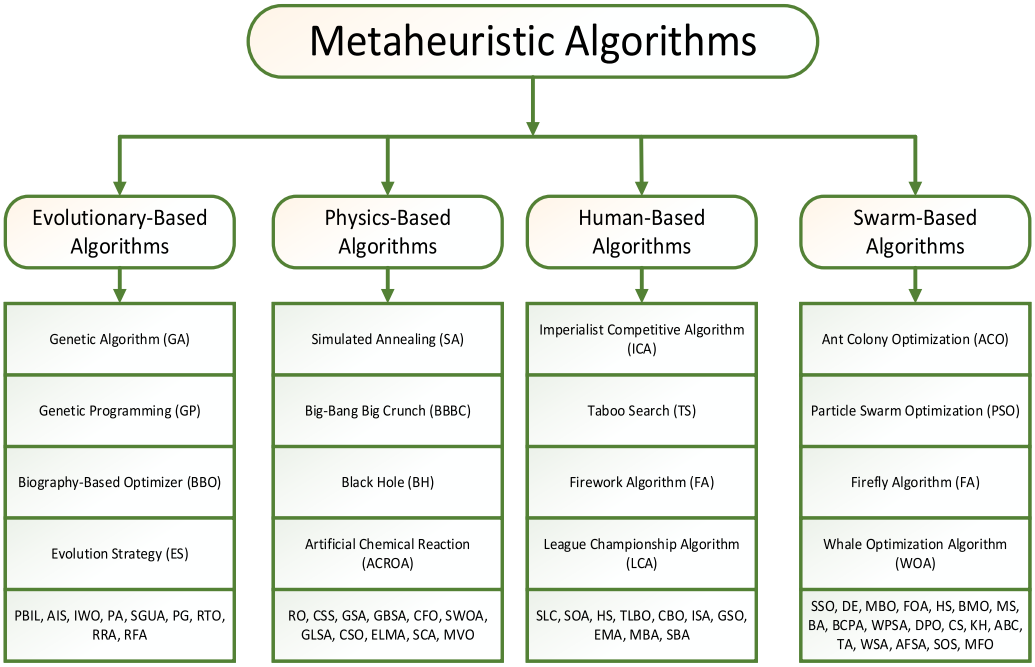

Fig. 1. The metaheuristic classification.

Table 1
Evolutionary-Based Algorithms summary details.

Algorithm	Published year	Cited count	Type
GA [12]	1992	4525	Discrete
GP [78]	1992	21907	Continuous
PBIL [85]	1994	1674	Continuous
PA [91]	2000	29	Continuous
ES [7]	2002	2845	Continuous
AIS [4]	2002	2981	Discrete
IWO [83]	2006	1325	Continuous
SGUA [43]	2006	30	Continuous
BBO [5]	2008	3264	Continuous
PFA [26]	2009	74	Continuous
RRA [84]	2015	93	Continuous
RTO [79]	2016	83	Discrete
PG [33]	2017	23	Continuous

Genetic Programming (GP). This algorithm used in autonomy applications [78]. The Biogeography-Based Optimizer (BBO) algorithm is another metaheuristic algorithm that improves the responses of a unit by repetition over many generations and also uses mutations [5]. Evolution strategy (ES) is an evolutionary algorithm developed by Schwefel et al. In this algorithm, the answers are agents in the environment. Instead of mutation, the components assigned a randomly distributed value [7]. Probability-Based Incremental Learning (PBIL) is an estimating algorithm and, in some cases, has performed better than the classical genetic algorithm. Creating answers in this algorithm follows a probability vector and, like other algorithms in this category, uses mutation [85] Other algorithms such as Artificial Immune System (AIS) [4], Invasive Weed Optimization Algorithm (IWO) [83], Photosynthetic Algorithm (PA) [91], Sapling Growing Up Algorithm (SGUA) [43], Plant Growth Optimization (PG) [33], Rooted Tree Optimization (RTO) [79], Runner Root Algorithm (RRA) [84], Paddy Field Algorithm (PFA) [26] belong to this group.

The second category of metaheuristic algorithms are algorithms based on physics. The most famous physical algorithm is the Simulated Annealing (SA) algorithm. In this algorithm, the process of crystallization of metal particles simulated when the molten metal cools slowly. This algorithm is the metaheuristic mode of the Hill

Table 2
Physics-Based Algorithms summary details.

Algorithm	Published year	Cited count	Type
SA [42]	1983,1985	51107,4671	Discrete
GLSA [115]	2003	81	Continuous
BBBC [59]	2006	1189	Continuous
SWOA [58]	2006	116	Continuous
CFO [63]	2007	409	Continuous
GSA [15]	2009	5200	Continuous
CSS [31]	2010	1060	Continuous
GBSA [37]	2011	201	Continuous
ACROA [47]	2011	250	Continuous
RO [35]	2012	515	Continuous
CSO [89]	2012	94	Continuous
BH [70]	2013	786	Continuous
ELMA [62]	2013	18	Continuous

Table 3
Human-Based Algorithms summary details.

Algorithm	Published year	Cited count	Type
TS [67,68]	1989, 1990	9228, 5677	Discrete
ICA [49]	2007	5291	Discrete
SOA [39]	2009	410	Continuous
GSO [14]	2009	740	Continuous
FA [11]	2010	859	Discrete
LCA [73]	2011	187	Continuous
MBA [105]	2013	528	Continuous
SBA [101]	2013	84	Continuous
EMA [66]	2014	148	Continuous
CBO [76]	2014	488	Continuous
ISA [64]	2014	305	Continuous
SLC [90]	2014	101	Continuous
TLBO [32]	2018	33	Continuous

climbing algorithm and is applied in the search space to achieve the optimal answer. Movement between neighbors takes place with the function of acceptance and avoids purely greedy choices [35,42]. Other algorithms such as Big-Bang Big Crunch (BBBC) [59], Black Hole (BH) [70], Artificial Chemical Reaction Optimization Algorithm (ACROA) [47], Ray Optimization (RO) [75], Charged System Search (CSS) [31], Gravitational Search Algorithm (GSA) [15], Galaxy-Based Search Algorithm (GBSA) [37], Central Force Optimization (CFO) [63], Small-World Optimization Algorithm (SWOA) [58], Gravitational Local Search Algorithm (GLSA) [115], Curved Space Optimization (CSO) [89], Electromagnetism-like mechanism algorithm (ELMA) [62] also belong to the category of Physics-Based algorithms.

The third category algorithms are designed based on social and human laws. The Imperialist Competitive Algorithm (ICA) is the most common in this category. This algorithm depicts the approach of the imperialist countries and their colonies in the form of mathematical laws. In this algorithm, the answers are the same countries. The best solution for any region is colonial, and the rest of the responses around it are colonies. The imperialists compete with each other until they get the right answer [49]. Firework Algorithm (FA) is an algorithm based on random initial answers that are followed by a metric search around the same starting points to find the optimal answer [11]. The League Championship Algorithm (LCA) simulates sports matches. This algorithm analyzes them by simulating a competition between teams. The winner of this competition transmits the most reliable answer to the output [65]. Other algorithms such as Taboo Search (TS) [67,68], Soccer League Competition Algorithm (SLC) [90], Seeker Optimization Algorithm (SOA) [39], Harmony Search(HS) [1], Teaching Learning Based Optimization (TLBO) [32], Colliding Bodies Optimization (CBO) [76], Interior Search Algorithm(ISA) [64], Group Search Optimizer (GSO) [14], Exchange Market Algorithm (EMA) [66], Mine Blast Algorithm (MBA) [105], Social-Based Algorithm(SBA) [101], also belong to the category of Human-Based algorithms.

Table 4
Swarm-Based Algorithms summary details.

Algorithm	Published year	Cited count	Type
PSO [77]	1995	67564	Discrete
MBO [17]	2001	512	Discrete
TA [104]	2005	48	Continuous
ACO [57]	2006	13876	Discrete
MS [20]	2007	266	Continuous
WPSA [2]	2007	150	Continuous
ABC [3]	2007	1355	Discrete
WSA [98]	2007	66	Continuous
BCPA [30]	2008	83	Continuous
DPO [107]	2009	88	Continuous
CS [120]	2009	5899	Continuous
AFSA [60]	2009	74	Continuous
FA [9]	2010	2167	Continuous
BA [28]	2010	4525	Continuous
HS [93]	2010	245	Discrete
FOA [94]	2012	1326	Continuous
KH [65]	2012	1487	Continuous
DE [74]	2013	363	Continuous
BMO [27]	2013	115	Continuous
SOS [53]	2014	940	Continuous
GWO [88]	2014	6165	Continuous
MFO [21]	2015	1708	Continuous
MVO [23]	2016	991	Continuous
SCA [86]	2016	1592	Continuous
WOA [87]	2016	3731	Continuous
SSO [122]	2017	44	Continuous

Fourth-class algorithms are algorithms based on the collective intelligence of living things and particles that alone do not have intelligence and even have limitations. One of these algorithms is the Ant Colony Optimization (ACO) algorithm, proposed by Dorigo, inspired by the cooperation of ants to reach the food source. This algorithm is used to solve discrete problems and achieve the path with the least cost [57]. Particle Swarm Optimization (PSO) is another algorithm inspired by the mass movement of birds to reach food. In this algorithm, the best personal experience and the best group experience are the two principles for achieving the answer [77]. Other algorithms such as Whale optimization algorithm (WOA) [87], Firefly (FA) [9], Social spider optimization (SSO) [122], Dolphin Echolocation (DE) [74], Marriage in Honey Bees Optimization (MBO) [17], Fruit Fly Optimization Algorithm(FOA) [94], Hunting Search (HS) [93], Bird Mating Optimizer(BMO) [27], Monkey Search(MS) [20], Bat Inspired Algorithm (BA) [28], Bee Collecting Pollen Algorithm (BCPA) [30], Wolf Pack Search Algorithm (WPSA) [2], Dolphin Partner Optimization (DPO) [107], Cuckoo Search(CS) [120], Krill Herd KH) [65], Artificial Bee Colony(ABC) [3], Termite Algorithm (TA) [104], Wasp Swarm Algorithm (WSA) [98], Artificial Fish-Swarm Algorithm(AFSA) [60], Symbiotic organisms search (SOS) [53] belong to this group.

Metaheuristic algorithms try to pave the way to the optimal solution by generating random answers, optimizing existing answers, chances of weak answers, avoiding deadlocks and breaking duplicate space, and at the same time complete these tasks in the shortest possible time. Exploration and Exploitation are two important principles in metaheuristic algorithms that make it possible to reach the desired answer and, at the same time, not be satisfied with the quasi-desirable answers. According to the NFL theory introduced in the introduction, metaheuristic algorithms can solve a specific range of problems. Therefore, the variety of these types of algorithms is wide [117].

3. Trees social relations optimization algorithm

In this section, the basics of the Trees social Relationship Optimization (TSR) algorithm, which is inspired by the social life of trees, are first described. The TSR algorithm is then fully described in detail.

Fig. 2. The root of the plants, the path of communication and sending messages to each other.

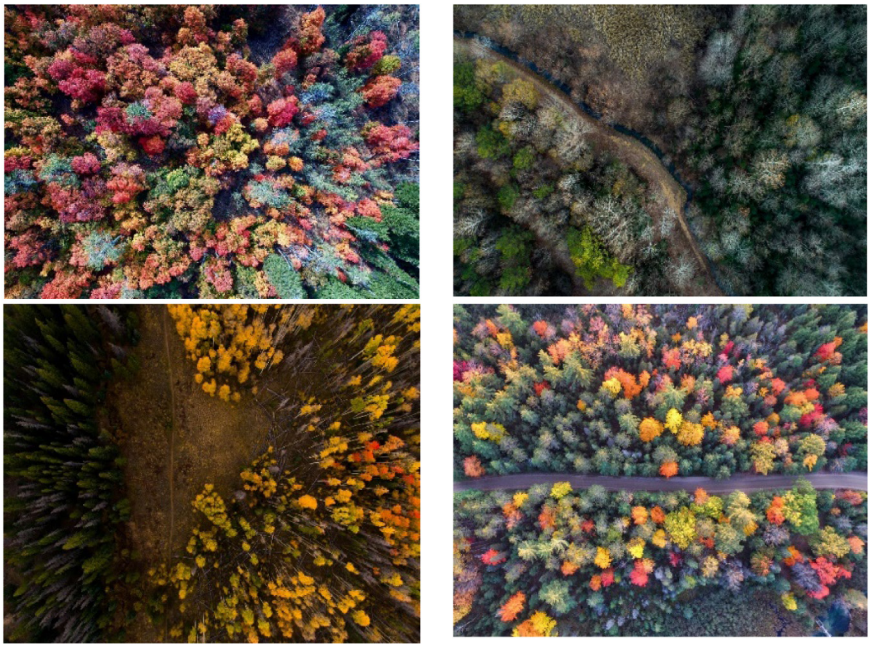

Fig. 3. Tree islands side-by-side, working together and competing with each other.

3.1. Inspiration

For the first time, Ms. Susan Simard proved the connection between trees. Communication that involved exchanging chemical messages through leaflets and talking to their roots [106]. Trees have a brain called the root that has the power to learn and memory and to extend it to future decisions [109]. Forests are one of the most dynamic creatures around us. In addition to delivering water and nutrients to different parts of the tree, the roots of the tree are responsible for decision-making and command [71]. Fig. 2 shows an example of the root of a tree and its communication path with other trees. Plants exchange information, learn about what is happening around them, and ask for help from nearby trees if there is a shortage of food around them. If something happens to the trees in an area mile away, they can report their condition through the air and the release of chemical odors, or through roots in the soil. Trees understand the changes in the chemicals around them. They recognize even the slightest change in the climate. All these changes and their exact senses cause them to react. The number of senses of trees is more than other creatures. They understand temperature, electromagnetic, chemical, etc. changes much better than other living things. Hill and Carbon demonstrated this in their 2009 studies [71,110]. In any area where several trees grow side by side, an island of trees usually forms. In fact, in the jungle, there are islands of trees next to each other. Fig. 3 shows four examples of island formation of trees. A communication channel connects the islands. According to the researchers, in the jungle, per islands are managed by one tree. This tree is called the pole. Ms. Simard called the poles the mother trees. The mother tree has the task of coordinating and sharing energy resources [108]. The

Fig. 4. The fungal network, network of tree connector.

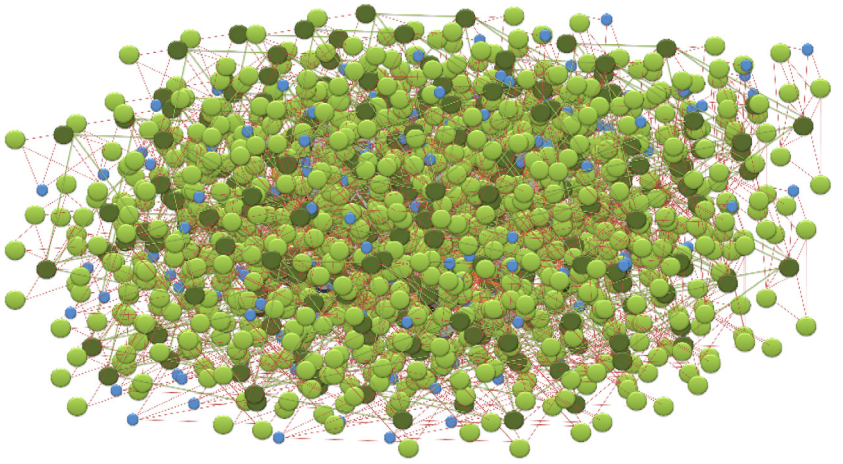

Fig. 5. Tree network, an extensive network within the jungle, including mother trees, seedlings and other trees.

priority is to feed with seedlings. If food resources are scarce, seedlings are at the forefront of the food queue, and if they are in demand, they are taken care of out of turn and with special priority. Researchers' experiments show that the decision-making power of trees lies in their roots. The pole tree (mother) makes decisions according to the environmental conditions and taking into account the demand of the trees in the subset. It is the mother who decides how much food should provide to the group. The mother tree ensures that the seedlings do not die. The role of mother trees is vital to jungle survival. If the mother trees destroyed in any way, the woods in the area covered by them will destroy. Plants are connected from the roots by an extensive network like the Internet in the soil. This network is made up of mycelium fungi, is called mycorrhiza. Fig. 4 shows examples of mycorrhiza fungal networks in different cases. These networks are continually growing and updating [108]. Fig. 5 shows a fake example of a tree

Fig. 6. Relation of mother trees together and relationship of mother trees with other trees.

communication network in a forest. In this form, the bold knots are the mother trees, and the lighter knots are the trees around them, as well as the small blue knots of the young trees that are important for the survival of the jungle. The forest regenerates through its network and makes this dynamic environment resistant to destructive factors and its subsequent reconstruction. Transmissions within this network cause the systematic induction of defense genes and security warnings and awareness of the danger [25,111]. As mentioned, the management of the trees on each island is with the mothers. The priority of responsibility in the jungle is generally to first look at the condition of the seedling trees. After that, the trees of the same species, then the neighbors and then other trees receive the answers to their requests. The preservation and survival of the woods, depending on the management of the mother trees [108]. Fig. 6 shows an example of the relationship between mother trees and their relationship to trees in their area. Studies show that when large and medium-sized trees destroyed, Norse and small trees die faster, which shows the critical and vital role of parents. Studies in the Birch Douglas jungle have also demonstrated that the quality of the region's commercial timber and its growth rate has a direct impact on the presence of older trees [112]. Various studies show that there is competition among tall trees for access to resources. In one of these studies, conducted on coniferous trees, it demonstrated that mother trees compete with each other. Where there are large trees, growth is higher among tall trees. Here large trees try to absorb more carbon and nutrients. This competition is also for food for the trees under their management [103]. Among trees, protoplasmic currents transmit warning signals through a fungal network. These warnings include messages from food sources, predators, fires, and climate change [112]. Accurate hierarchy and prioritization of forest conservation over the individual survival of each tree have made the forest a successful model for problem-solving. It seems that following this system will lead to solving problems.

3.2. TSR algorithm

TSR modeled on the collective living behavior of trees in the jungle. This algorithm based on the collective life characteristics of the trees that make it unique. The first and foremost feature of this algorithm is its ability to parallelize, which can split an issue into smaller sections called sub-jungles, and each sub-jungle can simultaneously handle assigned tasks. Working in parallel is part of the nature of the forest, different trees in the forest interact with each other to do their work. Another feature of this algorithm is its ability to solve both continuous and discrete problems. Therefore, it can use in a wide range of issues. The third feature of the TSR algorithm is to prioritize growing responses. By defining the growth rate (GR) parameter and determining the GR, the answers that, although they did not score well but had good growth compared to the previous generation, are allowed to participate in the next iteration. This work increases the likelihood of achieving optimal solutions. Increasing the GR is probably a sign of moving towards better answers. In the TSR algorithm, in addition to the best answers and answers that have a high GR, some weak answers also have a chance to reappear. This number of reuses is a percentage parameter. This value is randomly selected from the remaining answers in each iteration and sent for the next iteration. The TSR algorithm guarantees that the optimal response achieved at an acceptable time. The TSR algorithm, like other

metaheuristic algorithms, starts with an initial random population. In this algorithm, each of the answers considered as a tree. If we consider TSR as a vast jungle, it must first divide into several sub-jungles. The number of sub-jungles is a hyperparameter, meaning it must be determined before the algorithm can run at the input. In each main iteration of the algorithm, the sub-jungles can be executed simultaneously with multiple cores. The presence of sub-jungles increases population diversity with each repetition. Therefore, the answer obtained with more sub-forest is more optimal. Of course, too many sub-forests are not useful. The number of trees in all sub-forests is equal. Also, the sub-forests do not overlap with each other, and each tree belongs to only one sub-jungle. After creating the sub-jungles, the algorithm enters the main iteration stage. In the iteration phase, the algorithm applies two general processes to the answers. The first process is related to the operation of sub-jungles, which can do jobs in parallel and non-parallel. Of course, in the reviews and experiments of this article, parallelism has been avoided. The second process involves the management of the entire jungle. All jungle trees are responsible for a particular mother tree, which we call the king. In the following, we will examine these processes.

- **Sub-jungle process**

Sub-jungle process Operations are part of a set of operations that can be performed simultaneously in each iteration of the algorithm. At this stage, in each sub-jungle, p_s percent of high-fit trees are determined as seedling trees. This value is achieved by Eq. (2). In this relation, n is the total number of trees, and Z_j determines the number of trees in each sub-jungle when the number of sub-jungles is considered k. In sub-jungle operations, new trees produced by operators. The Proliferation and Proliferation-Seedling operators create two or more new trees from a parent pair. There are several types of parental choices available to the new population. The most commonly used methods are the roulette wheel and random selection. The Proliferation operator generally uses a roulette wheel to select parents. The ProliferationSeedling operator uses a combination of well-fitness trees with lower-fitness trees. For this purpose, a random tree from seedlings and a random tree from non-seedlings from the sub-jungle is selected. In discrete problems, these operators use different types of parent combinations to maximize population diversity. In continuous problems, an attempt made to use the parents' location to determine points for Newly produced trees. The number of parents sent to Proliferation and ProliferationSeedling operators determined by the input parameters p_p and p_{sp}, respectively. These two parameters are the percentage of sub-jungle trees assigned to operators. The Layering operator creates new trees with just one origin tree. This operator increases exploration. These operators move away from local answers by generating very different responses from their mother. The number of parents sent to the Layering operator is determined based on the percentage parameter p_l. For the new offspring of these three operators, in addition to calculating fitness, a parameter called growth rate (GR) is calculated and stored, which will use in the algorithm. GR shows the amount of change in the fitness of new trees relative to the parents. The GR at the beginning of work is zero for all trees. It calculated according to the positive or negative growth of the new population concerning the parent(s). The GR is obtained for new two-parent trees with Eq. (3) and single-parent trees with Eq. (4). In these relationships, is the value of new answers fitness, and is the value of parent(s) fitness. When the GR is more than one, it means that the answers are getting better. In both discrete and continuous problems, when the GR is above one, it means that the answers are improving. A value of less than one also means moving away from the right answers. GR can identify potential answers in the response space to give these cases more opportunity to generate new answers. In both discrete and continuous forms, GR means the size of the growth of the answers. This parameter determines the new seedlings. GR sorts the trees of each sub-jungle, and the list of seedlings is updated.

$$Z_j = \frac{n}{k} \tag{1}$$

$$SD_j = Z_j * p_s \tag{2}$$

$$GR = \frac{fitness_{new}}{fitness_{p1} + fitness_{p2}} \tag{3}$$

$$GR = \frac{fitness_{new}}{fitness_{p}} \tag{4}$$

Next, to determine the population sent from this stage, in each sub-jungle, the total initial and produced population is sorted according to the amount of fitness. The number of trees in each sub-jungle should be equal to $(\frac{n}{k})$, according to Eq. (1). In the proposed method, a percentage of weak responses also reused. This random selection is made

Table 5
TSR parameters.

Parameters of TSR	Definition
n	*population size*
k	*Sub-jungle count*
p_s	*percent of seedling trees*
p_p	*proliferation percent*
p_{sp}	*Seedlings-Proliferation percent*
p_l	*layering percent*
p_e	*transformation percent*
maxiter	*maximum iteration*

to reuse these answers in the production cycle. This mechanism, which reinforces the principle of exploration, is inspired by trees and dead trees components that provide nutrients to the soil surface of trees. The population that each sub-jungle produces for subsequent iterations is z_j, and the rest eliminate. The seedlings are updated after the new population identifies for the next iteration. For this purpose, the population in the sub-jungle is sorted by GR. Then p_s percent of the beginning of the list is selected as seedlings.

- **Global process**

The goal of the Global process is to balance sub-jungles. That is, production should continue in such a way that all sub-forests have a chance to reach an optimal answer. For this purpose, in the Global process, fitness average is calculated for all sub-jungles. This average fit is obtained according to Eq. (5).

$$Zf_j = \frac{k * \sum_{u=1}^{n} fitness_u}{n} \tag{5}$$

According to the number of sub-jungles and the total population, the value of zf_j obtain. After calculating zf_j, all sub-jungles are sorted by fitness average, meaning the first sub-jungle of the list has the best fitness average and is ranked best. The upper half of the sorted list is considered as strong sub-jungles, and the lower half is considered as weak sub-jungles. In the following, some high-fitness answers copy from the strongest sub-jungles to the weakest, respectively. The number of submitted trees determine from strong sub-forests with the parameter Pe. In fact, (p_e) specifies the percentage of trees sent. For the algorithm not to work greedily, this response should not always send. For this purpose, the normalized value for each sub-jungle is first calculated according to Eq. (6).

$$PT = 1 - \frac{zf_j}{\sum_{w=1}^{allzone} zf_w} \tag{6}$$

After calculating the PT for each sub-jungle, a random number between zero and one generates, and if this random number is bigger than the PT, then, best trees are transferred to the destination sub-jungle. If the random number is less than PT, the transfer will not take place. Note that in the first iterations, the probability of transmission is higher and gradually decreases. The two principles of exploration and exploitation have used in many different ways in TSR, which has made it very powerful in solving hard problems. If the transfer takes place, strong trees have replaced weak ones, so all sub-jungles still have an equal number of trees. After the stop condition set, the best answer in the forest send to the output. In general, the stop condition considers a certain number of repetitions of new population production. The flowchart of the proposed method is shown in Fig. 7.

3.2.1. Algorithm

The pseudo-code of the TSR metaheuristic algorithm, which is inspired by the social life process of forest trees, is shown in Fig. 8.

As mentioned in the TSR algorithm, each answer is considered a tree. So, the algorithm is looking for the best tree. First, the algorithm parameters receive as input. These parameters are the number of the initial population, the number of subsets, the parameters related to the algorithm operators, and the number of iterations of the algorithm. The input parameters of the TSR algorithm are described in Table 5, along with the description.

The parameter n determines the number of initial populations in the algorithm. The number of sub-jungles denotes by k. The p_s parameter specifies what percentage of sub-jungle trees are considered seedlings. The percentage of

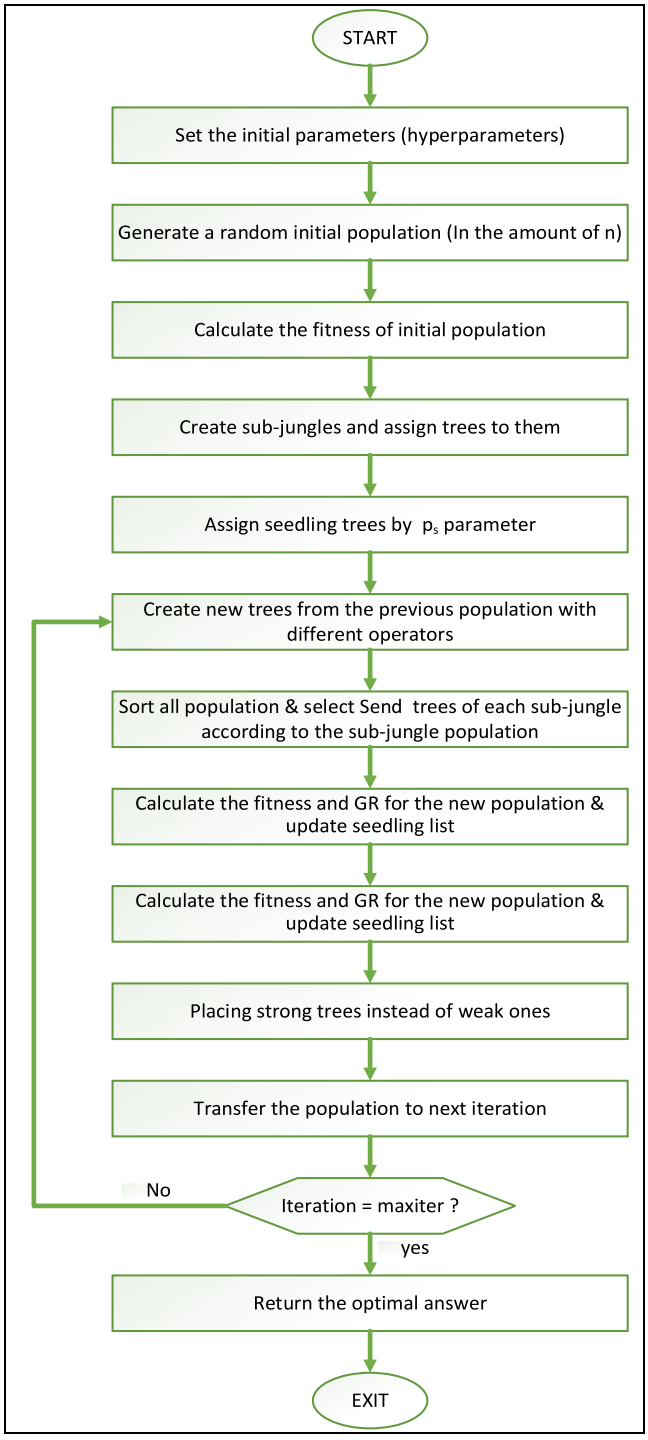

Fig. 7. Flowchart of the proposed TSR.

the population provided by the Proliferation, Seedlings-Proliferation, and Layering operators determines by the parameters p_p, p_{sp}, and p_l, respectively. The maxiter parameter also specifies the number of iterations of the algorithm. After determining the values of the parameters, n trees produce as the initial population. The fitness of each of the initial trees is then calculated. At first, the GR of the trees is set to 0. Because trees have an initial

Algorithm Trees social Relations Optimization Algorithm (TSR)

Input: *n, k, p_s, p_p, p_{sp}, p_l, p_e, maxiter*
Output: *Best Solution*

Procedure TSR
Generate initial trees (jungle) *P* with *n* trees
Compute *fitness* for each tree $\in$ p
Set the *GR* of all trees in p_i to 0
Generate Z one set: $\begin{cases} Z_j(j = 1.2. \dots k). |Z_j| = \frac{n}{k}\ .\ \ n \le k \le 1 \\ z_m \cap z_n = \emptyset \text{ for } \forall\ (m.n) \in j\ .\ m \ne n \end{cases}$
For each Z_j in Zone set **do**
  SD_j = select $(\frac{n}{k}) \times p_s$ trees in *Zj* randomly as seedlings
For i =1 to *maxiter* do
  Sub-jungle Process:
    For each Z_j in Zone set **do**
      Proliferation:
        For h=1 to ((n/k) $\times P_p$) **do**
          TA1 = Select a random tree from *Zj* based on the fitness proportionate
          TA2 = Select a random tree from *Zj* based on the fitness proportionate
          Do *Proliferation* function with *TA1* & *TA2* and generate *NA1* & *NA2*;
          Compute *fitness* of *NA1* & *NA2* and Add *them* to *Zj*;
          $GR_{NA1} = \frac{fitness_{NA1}}{(fitness_{TA1}+fitness_{TA2})/2}$, $GR_{NA2} = \frac{fitness_{NA2}}{(fitness_{TA1}+fitness_{TA2})/2}$
      SeedlingsProliferation:
        For h=1 to (n/k × P_{sp}) **do**
          TG1 = Select a random tree from *SDj*
          TG2 = Select a random tree from *Zj - SDj*
          Do *SeedlingsProliferation* function with *TG1* & *TG2* and generate *NG1* & *NG2*;
          Compute *fitness* of *NG1* & *NG2* and Add them to *Zj*;
          $GR_{NA1} = \frac{fitness_{NA1}}{(fitness_{TA1}+fitness_{TA2})/2}$, $GR_{NA2} = \frac{fitness_{NA2}}{(fitness_{TA1}+fitness_{TA2})/2}$
      Layering:
        For h=1 to (n/k × P_l) **do**
          TL = Select a random tree from *Zj*
          Do *Layering* function with *TL* and generate *NL*;
          Compute *fitness* of *NL* and Add *NL* to *Zj*;
          $GR_{NL} = \frac{fitness_{NL}}{fitness_{TL}}$
      Eliminate Pest Trees:
        Sort all trees in *Zj* by *fitness*
        Eliminate the last [(n/k) × (P_p+ P_{sp} +P_l)] trees from *Zj*
        Reuse some eliminate trees
      Seedlings update:
        Sort all trees in *Zj* by *GR*
        Select the first ((n/k) × p_s) trees as SD_{zj}
  Global King Process:
    For each zone Z_j in *P* **do**
      $zf_j = \frac{k \times \sum_{u=1}^{k} fitness_u}{n}$
    Sort all Zone by *Zf*
    For j=1 to *k/2* **do**
      BT = Select the best ($(\frac{n}{k}) \times P_e$) trees form Z_j
      $PT = 1 - \frac{zf_j}{\sum_{w=1}^{all\ zone} zf_w}$
      If (rand (0,1) > PT) **Then**
        BT = Best trees
      Add BT to Z_{n-j+1}
      Eliminate the worst ($(\frac{n}{k}) \times P_e$) trees from Z_{k-j+1}
Best Solution = the tree with the best *fitness* in *P*

Fig. 8. Semi code of the TSR Algorithm.

amount of fitness, new responses are needed to calculate the GR. In the next step, k sub-jungles are produced. The number ($Z_j = \frac{n}{k}$) of initial trees is given to the sub-jungle as a subset. According to the p_s parameter, seedling trees are identified for each sub-jungle. In the next iterations, the trees select as seedlings based on GR. The number of iterations determined by the (maxiter) parameter performs on each sub-forest of offspring production and processing operations. First, the sub-jungle process completes. The Proliferation operator receives the corresponding trees according to the p_p parameter and produces new answers by combining a pair of parents, then calculates the fitness of new answers. Finally, the amount of GR is determined. The GR value determines how well new trees have grown relative to their parents on a fitness scale. A value bigger than 1 means more growth. Note that in this operator, parents select with a roulette wheel. In the Seedling-Proliferation operator, one parent randomly selects

from seedling trees and the other parent from non-seedling trees. The number of trees sent to this operator determines according to the p_{sp} parameter. In this operator like Proliferation, after creating new trees, their fitness and GR are calculated. In Layering, a new answer generates from a single mother. The mother randomly selects in this operator. The number of trees sent to this operator is determined based on the p_l parameter. After having a new answer, as in the previous two cases, its fitness and GR calculate. In each sub-jungle, weak trees remove. For this purpose, the trees of each sub-jungle arrange according to the values of fitness. Then the number of Z_j trees remain, and the rest removed. Then the trees of per sub-jungles are sorted according to GR values, and (p_s) percent of them determine as seedlings. The Global Process then apply to trees throughout the jungle. First, the average fitness or zf_j for each sub-jungle calculated. Next, the sub-jungles are sorted according to these values. Then, based on the p_e parameter, at the same percentage, the best answers transfer from the selected strong sub-jungle to the weak sub-jungle. Appropriate answers also transfer from the second strong sub-jungle to the second weak sub-jungle. In the same way, the transition from strong to weak sub-jungles takes place. Of course, note that the transfer is not always greedy. For this purpose, PT is calculated. Then a random number between 0 and 1 is generated. If the random number is bigger than PT, then the transfer takes place. Note that in the initial iterations, the chance of transmission is more, and in the final iterations, this chance decreases. After completing the number of iterations of the algorithm, the best answer sends as output.

According to pseudocode, TSR includes its own innovations, including a special structure, dividing data into a zone called sub-jungle to be processed in parallel. In addition to giving poor answers a chance, specimens that are likely to get better answers in later generations are identified with GR to be used in future generations. Samples are converted to new samples at each stage by three different manufacturers: Proliferation, Seedling-Proliferation and Layering. Also, in some iteration randomly, weak sub-jungle answers are replaced with stronger responses. All of these factors work together to find the best possible solution for exploration and exploitation.

3.2.2. Computational complexity

In this section, we examine the complexity of the TSR algorithm. In the TSR algorithm, we have the initial population of n, which divides into k sub-jungles. In each sub-jungle, we have different operators, each of which works on a percentage of the initial population depending on the defined parameter: proliferation, seedling-proliferation, and layering apply to the trees of the sub-jungles with percentages of p_p, p_{sp}, and p_l, respectively. There are also two sorts in the algorithm for removing weak trees and updating seedlings. In the end, there is a transfer of $(Z_j/2)$ from strong sub-jungles to weak sub-jungles. This algorithm repeats by default in the number of maxiter. Therefore, the time complexity of the TSR algorithm can be expressed as Eq. (7).

$$O\left(maxiter*\left(\frac{n}{k}*p_p+\frac{n}{k}*p_{sp}+\frac{n}{k}*p_l+nlogn+nlogn+\frac{k}{2}\right)\right) = O(maxiter*(\frac{n}{k}*(p_p+p_{sp}+p_l)+2nlogn+k/2)) \tag{7}$$

In terms of memory usage complexity, it can say that the algorithm needs as much space as the initial population and new responses each iteration. Therefore, the memory consumption complexity of the TSR algorithm can be expressed as Eq. (8).

$$O\left(n+n/k\left(p_p+p_{sp}+p_l\right)\right) \tag{8}$$

3.2.3. Parallelism in TSR

One of the essential features of Trees social Relations Optimization Algorithm is the possibility of parallelization in it. TSR response space can process in two ways, single-core, and multi-core. As mentioned earlier, we consider the initial population to be n. If we consider the whole algorithm as a vast jungle, it can be divided into several sub-jungles, each of which has a leader called the mother, which oversees the performance of that sub-jungle. The whole jungle is under the supervision of a tree called the king. If we consider the number of subsets of the algorithm as (k) and the number of initial answers as n, then we will have a population of ($Z_j = n/k$) in each subcategory. Under the supervision of the mother of each group, the operation of making new answers perform. During parallel implementation, each sub-jungle is delivered to a kernel to perform its related processes. So Z_j can be the number of populations assigned to each kernel in a computer system. This process can also define continuous problems as the scope of operation of each core. That is, each kernel examines a specific domain of the problem, and the

king tree determines the best answer to the whole problem as the output. In parallel execution of this algorithm, the accuracy of the answers increases due to the construction of different answers, and the algorithm converges faster. In continuous problems, each core implements the two principles of exploration and exploitation in smaller intervals, and finally, the best answer selects among the various answers. As mentioned earlier, for all the algorithms studied in this paper to consider on equal terms, the TSR is not run in parallel.

4. Experimental results and discussion

In this section, experiments related to the performance of the proposed algorithm in both areas of continuous and discrete problems present. To test the TSR algorithm in the field of discrete problems, several classical problems in this field introduce, and the efficiency of the proposed method examines. Then, for continuous problems, some standard functions are introduced, and the TSR test results on these functions examine. Several experiments have performed to prove the performance of the TSR algorithm compared to other algorithms. Section 4.1 first introduces several classical discrete problems and then lists the parameters of the discrete algorithms used in the tests in a table. Section 4.2 introduces the continuous algorithms, along with the parameters used. Also, the standard functions with the domain and their minimum values given in separate tables. In Section 4.3, first, the results of different tests of discrete problems on TSR with different parametric values are given. The comparisons between the TSR algorithm and other algorithms are given graphically and, in the table, numerically for better comparison. These comparisons are first on classic problems and then on an issue related to the power required by servers in different geographical locations. In Section 4.4, first, the TSR algorithm is tested independently with different parametric values, then the continuous and TSR algorithms are compared, and graphical and numerical results given. First, comparisons done on standard functions. These comparisons then perform on three practical problems. These issues are robot routing, feature selection, and image color clustering, respectively.

4.1. Benchmark test for discrete problems and algorithms used for comparisons

Several tests and comparisons have performed to test the performance of TSR in discrete problems. First, the TSR test uses to examine the effect of changes in the initial population and the number of variables on discrete problems. These experiments performed on the knapsack (KN) problem [29] and the traveling salesman [102] problem (TSP). Then, the response of TSR algorithm is compared with GA [61], GWO [88], ICA [49], SA [35,42], TABU [67,68] and ACO [57] algorithm in backpack problem. Then, the results of comparisons of the proposed method with GA, GWO, ICA, SA, and TABU algorithms in the TSP problem are examined in two cases of 1000 cities and 10,000. After this part, an essential and practical problem related to servers distributed in various locations is solved by TSR and compared with other discrete algorithms. In this case, several servers located in various locations are supposed to provide services to the surrounding antennas. Each antenna must connect to the nearest server. Of course, all servers must use optimally, and everyone's power must use in a balanced way. The parametric values of discrete algorithms are described in Table 6.

4.2. Benchmark functions for continuous problems and algorithms used for comparisons

First, experiments are performed on several standard functions to evaluate the performance of the algorithm in continuous problems. These experiments perform on unimodal [8] and multimodal [80] functions as well as a series of fixed-dimension multimodal functions [8,80]. GA [72], HS [1], ICA [49], GWO [88], MFO [21], MVO [23], PSO [77], SCA [86] and WOA [87] algorithms were uses for comparison. Unimodal functions listed in Table 7, multimodal functions in Table 8, and Table 9 also include fixed-multimodal dimension functions. These tables list the optimal values as well as the range of functions. The diagrams of the functions of Tables 7–9 are shown in Figs. 9–11, respectively. We will first see the effect of changes in the initial population on the TSR through a graph. In the following, we will apply the TSR algorithm and other mentioned algorithms to the above function, and we will see the results on the graph. We also implement three application problems and apply algorithms to them. These include finding the shortest path in an environment with obstacles to guide the robot, the cost of image color clustering, and feature selection. The parametric values of the continuous algorithms describe in Table 10.

Table 6
Parameter values of discrete competitor algorithms.

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

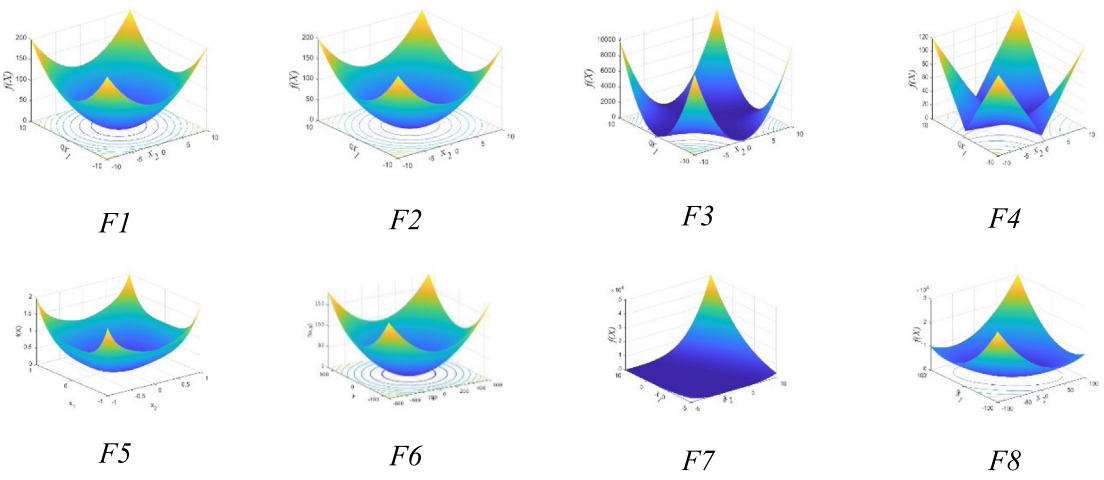

Fig. 9. 3-D plotting of unimodal benchmark functions.

4.3. TSR in discrete problems

To experiment with the performance of the TSR algorithm in solving discrete problems, we have performed several tests below. Initially, the traveling salesman problem (TSP) performed with the different initial populations and the number of various repetitions. Also, with different initial populations and the number of different variables, the knapsack (KN) problem is presented. Then discrete algorithms were implemented to solve the backpack and TSP problems, and their results show for comparison on the diagram. For this evaluation of the proposed method

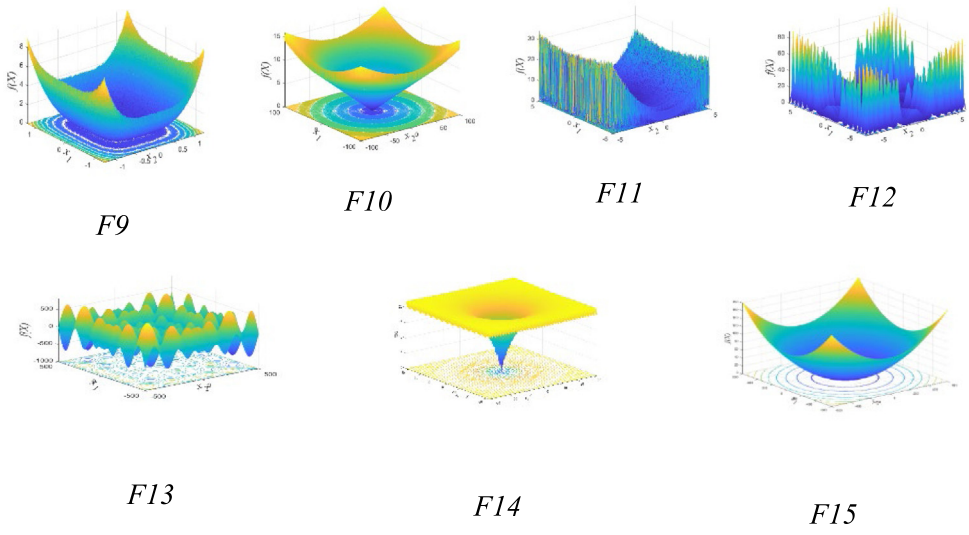

Fig. 10. 3-D plotting of multimodal benchmark functions.

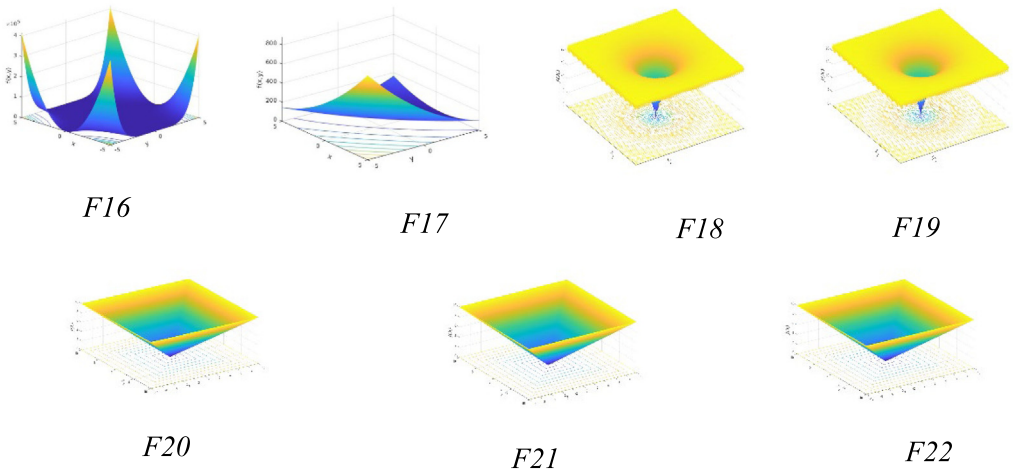

Fig. 11. 3-D plotting of fixed-dimension multimodal benchmark functions.

to have a practical aspect, an applied problem solves at the end of this section with algorithms, and its comparison gives in the relevant section.

4.3.1. TSR in discrete problem solving

First, we evaluate the proposed algorithm separately. In this way, the algorithm first uses to solve discrete problems in the two classic problems of TSP and KN problem to determine what effect the number of the initial population and the number of repetitions have on solving these problems. Then, the TSR response compared with the answers of other algorithms are plotted on a graph. Fig. 12 shows the effect of the number of iterations, and Fig. 13 shows the effect of the initial population on the TSP. As shown in Fig. 12, the higher the iteration, the better the algorithm. Fig. 13 shows that the larger the initial population, the sooner it converges. Of course, even in smaller populations, the TSR algorithm guarantees the optimal answer. Fig. 14 shows the effect of the number of variables in KN problem. The greater the number of initial variables, the more complicated the problem. Fig. 15 shows the effect of the initial population on KN problem. Like TSP, more early populations accelerate convergence.

Table 7
Unimodal benchmark functions.

Function	*Dim*	*Range*	*Fmin*
$f1(x) = \sum_{i=1}^{n} x_i^2 + random[0;1]$	30	[−5.12,5.12]	0
$f2(x) = \sum_{i=1}^{n} x_i^2$	30	[−100,100]	0
$f3(x) = \sum_{i=1}^{n} x_i^2 + \prod_{i=1}^{n} x_i^2$	30	[−100,100]	0
$f4(x) = \sum_{i=1}^{n} i x_i^2$	30	[−10,10]	0
$f5(x) = \sum_{i=1}^{n-1} \left(x_i^2\right)^{x_{i+1}^2+1} + \left(x_{i+1}^2\right)^{x_i^2+1}$	30	[−1,4]	0
$f6(x) = 1 + \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} cos(\frac{x_i}{\sqrt{i}})$	30	[−600,600]	0
$f7(x) = \sum_{i=1}^{n} x_i^2 + (\sum_{i=1}^{n} 0.5 i x_i)^2 + (\sum_{i=1}^{n} 0.5 i x_i)^4$	30	[−5,10]	0
$f8(x) = \sum_{i=1}^{n} (\sum_{j-1}^{i} x_j)^2$	30	[−100,100]	0

Table 8
Multimodal benchmark functions.

Function	*Dim*	*Range*	*Fmin*
$f9(x) = \sum_{i=1}^{n} i x_i^4 + random\ [0;1]$	30	[−1.28,1.28]	0
$f10(x) = 1 - cos(2\pi \sqrt{\sum_{i=1}^{D} x_i^2}) + 0.1(\sqrt{\sum_{i=1}^{D} x_i^2})$	30	[−100,100]	0
$f11(x) = \sum_{i=1}^{n} \varepsilon_i \lvert x \rvert_i^i$	30	[−5,5]	0
$f12(x) = (\sum_{i=1}^{n} \lvert x_i \rvert) exp(-\sum_{i=1}^{n} sin(x_i^2))$	30	[−2π,2π]	0
$f13(x) = \sum_{i=1}^{n} -x_i sin(\sqrt{\lvert x_i \rvert})$	30	[−500,500]	−418.98
$f14(x) = -20exp\ (-0.2 \sqrt{\frac{1}{n} \sum_{i=1}^{n} x_i^2}) - exp\ (\frac{1}{n} \sum_{i=1}^{n} cos(2\pi x_i))$	30	[−32,32]	0
$f15(x) = \frac{1}{4000} \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} cos(\frac{zi}{\sqrt{i}}) + 1$	30	[−600,600]	0

4.3.2. TSR and other discrete algorithms comparison

In this section, TSR compares with other discrete algorithms. Fig. 16 shows a comparison chart of the TSR algorithm with other algorithms in the KN problem. In this case, we are looking to fill a bag with a capacity of 625 per unit weight to achieve the maximum benefit. In this case, the initial population is 50, and the number of iterations is 100. Also, the maximum number allowed for each object is 5. With all these initial assumptions, Fig. 16 shows a comparative diagram of this problem. As shown in the figure, the TSR algorithm retains the best response from the first iteration compared to other algorithms. The best answer obtains in the 75th repetition for TSR. Also, the answer received by the TSR algorithm in the 8th iteration is better than the best answer of the GWO algorithm,

Table 9
Fixed dimension multimodal benchmark functions.

Function	*Dim*	*Range*	*Fmin*
$f16(x) = \sum_{i=1}^{11}[a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4}]^2$	*4*	*[−5, 5]*	*0.00030*
$f17(x) = -(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6)^2 + 10\left(1 - \frac{1}{8\pi}\right)cosx_1 + 10$	*2*	*[−5,5]*	*−398*
$f18(x) = -\sum_{i=1}^{4} c_i exp(-\sum_{j=1}^{3} a_{ij}(x_j - p_{ij})^2)$	*3*	*[0,1]*	*−3.86*
$f19(x) = -\sum_{i=1}^{4} c_i exp(-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2)$	*6*	*[0,1]*	*−3.32*
$f20(x) = -\sum_{i=1}^{5}[(X - a_i)(X - a_i)^r + c_i]^{-1}$	*4*	*[0,10]*	*−10.1532*
$f21(x) = -\sum_{i=1}^{7}[(X - a_i)(X - a_i)^r + c_i]^{-1}$	*4*	*[0,10]*	*−10.4028*
$f22(x) = -\sum_{i=1}^{10}[(X - a_i)(X - a_i)^r + c_i]^{-1}$	*4*	*[0,10]*	*−10.5363*

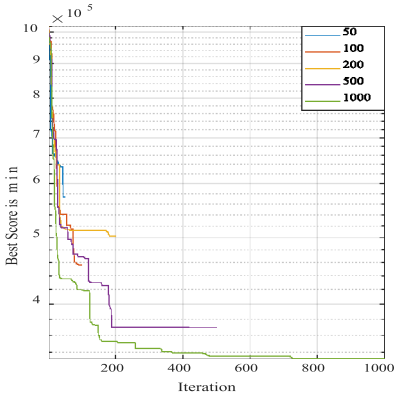

Fig. 12. Influence of the number of iterations on the TSP.

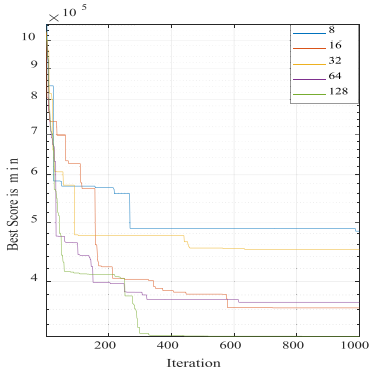

Fig. 13. Influence of the number of populations on the TSP.

which is in the second place. Two datasets have used to compare the TSR algorithm with the discrete algorithms in TSP. In a case where the comparison diagram describes in Fig. 17, the database consists of 1000 cities. To solve this problem, the initial population is 50, and the number of iterations is 100. The TSR algorithm gives the best

Table 10
Parameter values of continuous competitor algorithms.

Algorithms	Parameters	Values
Trees social Relations Optimization Algorithm (TSR)	Population size	50
	Number of generations	1000
	Number of sub-jungles	1
Genetic Algorithm (GA)	Population size	50
	Number of generations	1000
	Gamma	0.4
Grey Wolf Optimizer (GWO)	Control Parameter a	[2, 0]
	Number of generations	1000
	Search Agent	50
Imperialist Competitive Algorithm (ICA)	Number of particles	50
	Number of countries	50
	Number of generations	1000
Harmony Search (HS)	Number of nimp	10
	Neighboring value rate	0.3
	Discrete set and fret width	17700, 1
	Number of generations	1000
Moth–Flame Optimization	Convergence constant	[−1, −2]
	Logarithmic spiral	0.75
	Search agents	100
	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
Sine Cosine Algorithm (SCA)	Number of elites	2
	Number of generations	1000
	Search agent	100
Whale Optimization Algorithm (WOA)	Number of generations	1000
	Parameter b	1
	Initial population	100
Multi-Verse Optimization (MVO)	Wormhole Existence Prob.	[0.2,1]
	Traveling Distance Rate	[0.6,1]
	Number of generations	1000
	Search agents	100

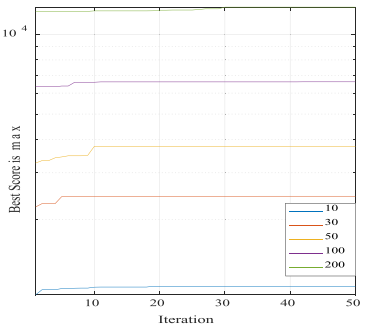

Fig. 14. Influence of the number of variables on the KNAPSACK.

answer in comparison or other algorithms from the fifth iteration onwards. The slope of the TSR diagram is steeper than other algorithms. For this reason, with increasing repetition, better results are obtained. In the second case, the number of cities is 10,000, and a standard and real database are used [46]. To solve this problem, the initial

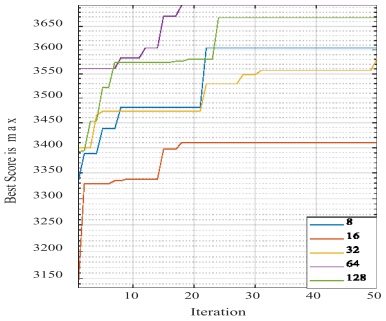

Fig. 15. Influence of the number of populations on the KNAPSACK.

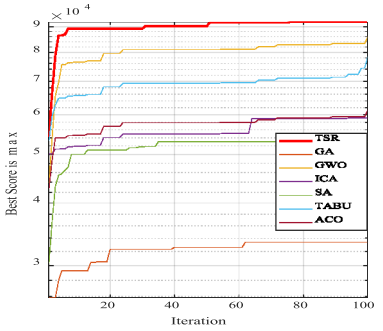

Fig. 16. Comparison of TSR with other discrete algorithms on Knapsack problem.

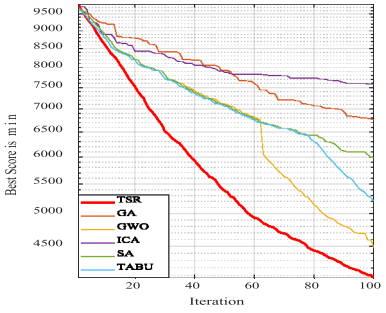

Fig. 17. Comparison of TSR with other discrete algorithms on TSP with 1000 cities.

population is 100, and the number of repetitions is 1000. As shown in Fig. 18, the TSR algorithm has the best response in all iterations compared to other algorithms. The GWO and TABU algorithms, which are in the second and third positions, are very far from the TSR algorithm. It should also note that the TSR algorithm competes with other algorithms in terms of time and is a fast algorithm. In TSP, for example, with a small number of cities, the ant colony algorithm responds better than TSR. But, when the number of cities reaches tens of thousands, the ant colony algorithm becomes very slow, and the TSR algorithm quickly achieves the optimal or near-optimal response.

4.3.3. Server placement in mobile edge computing

In this section, we compare the proposed algorithm with other discrete algorithms using a real problem. In this case, 300 antennas with the required service values, which include different values, are located in a location marked with x, *y* coordinates. The antenna must connect to the nearest server that can service. Distribution of work across

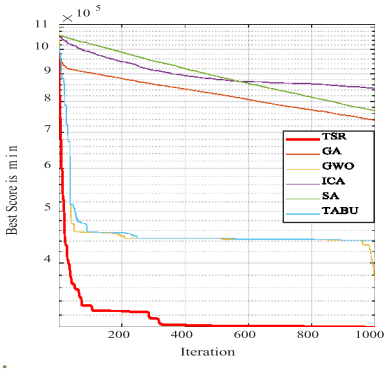

Fig. 18. Comparison of TSR with Other Discrete Algorithms on TSP with 10,000 cities.

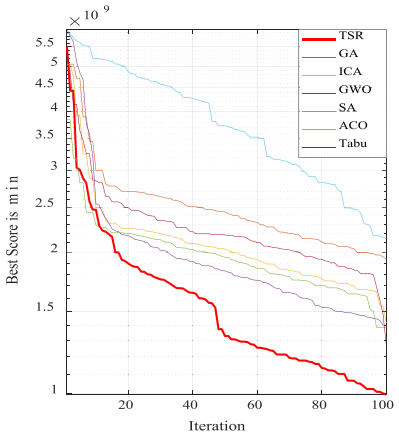

Fig. 19. Comparison of TSR with other discrete algorithms on server placement.

Table 11
Antenna and server implementations results.

Algorithm	Worst	Best	Median	Std.Dev.	Mean
TSR	5.50E+09	**9.95E+08**	**1.32E+09**	**6.32E+08**	**1.62E+09**
GA	5.05E+09	1.93E+09	2.43E+09	6.59E+08	2.55E+09
ICA	5.04E+09	1.42E+09	2.02E+09	6.41E+08	2.14E+09
GWO	5.99E+09	1.40E+09	1.82E+09	9.30E+08	2.07E+09
SA	3.84E+09	1.38E+09	1.95E+09	1.18E+09	1.99E+09
ACO	5.88E+09	2.14E+09	3.72E+09	1.04E+09	3.84E+09
Taboo	5.49E+09	1.27E+09	2.16E+09	6.42E+08	2.30E+09

servers should be fair. The total power of the servers uses on average. Algorithms must pursue two goals. A critical issue in the mobile networks to provide quality Internet is the proper distribution of load across servers. Of course, the number of antennas is more than the number of servers, and several antennas connect to one server at the same time. The idea for this take from the article Edge server placement in mobile edge computing [114]. Fig. 19 shows a comparison diagram of the proposed algorithm with other algorithms in this problem. Table 11 also shows the numerical values of these experiments. This problem implements with ACO, GA, GWO, ICA, SA, TABU, and TSR algorithms. In the early iterations, the competition between the algorithms is close except for the ant colony. The TSR algorithm has the best answer among the algorithms since the 12th iteration. According to the diagram, from the 50th iteration onwards, the TSR has distanced itself from other algorithms.

Table 12
Results of unimodal benchmark functions.

	TSR		GA		GWO		HS		ICA	
F	Ave	Std	Ave	Std	Ave	Std	Ave	Std	Ave	Std
f_1	0.27E+003	1.46E−003	0.01E+003	0.34E+003	0.16E+003	3.67E+000	0.25E+003	3.25E+003	0.085E+000	0.90E+000
f_2	148.09E+003	2.94E+003	162.54E+003	1.52E+003	619.96E+003	4.22E+003	122.43E+003	96.65E−000	4.9E0E+000	47.56E+000
f_3	154.03E+003	2.33E+003	1.39E+003	27.00E+003	433.44E+003	7.50E+000	133.49E−003	1.53E+000	42.93E+000	1.10E+003
f_4	24.01E−003	2.75E+000	43.91E−003	854.5E−003	839.60E−003	9.08E+000	290.77E−0003	3.79E+000	99.42E−003	1.52E+000
f_5	188.85E−003	356.35E−003	1.714E−003	31.154E−003	126.95E−003	1.96E+000	628.96E−003	9.30E+003	8.329E−003	93.43E−003
f_6	30.92E−003	580.54E−003	46.33E−003	248.01E−003	92.16E−003	947.47E003	20.72E−003	35.35E−003	106.63E−003	870.43E−003
f_7	855.17E−003	2.93E+000	25.31E−003	224.72E−003	302.12E−003	4.76E+000	916.89E−003	7.52E+000	347.37E−003	1.74E+000
f_8	750.51E+000	7.42E+003	1.450E+003	2.69E+003	3.63E+003	28.22E+003	680.79E−003	2.98E+003	26.91E+000	207.22E+000
	MVO		PSO		SCA		WOA		MFO	
F	Ave	Std	Ave	Std	Ave	Std	Ave	Std	Ave	Ave
f_1	0.06E+003	0.94E+003	0.01E+000	0.04E+000	0.06E+000	0.81E+000	0.10E+000	1.561E+002	0.18E+003	1.59E+003
f_2	225.52E+003	988.45E+003	14.22E+000	227.72E+000	664.50E+000	1.64E+003	350.56E+000	3.96E+003	206.58E+003	1.24E+003
f_3	15.00E+000	154.14E+000	66.06E+000	1.56E+003	3.11E+000	33.61E+000	28.02E+000	802E+000	313.09E+000	5.88E+003
f_4	99.40E−003	816.40E−003	94.86E−003	586.63E−003	794.09E−003	7.53E+000	140.10E−003	2.03E+000	198.76E−003	1.719E+000
f_5	17.78E−003	121.49E−003	8.78E+003	63.74E−003	6.14E−003	47.45E−003	9.064E−003	133.11E+003	21.13E−003	317.13E+003
f_6	290.32E−003	911.73E−003	638.28E003	3.05E+000	215.38E+003	1.08E+000	49.59E−003	805.30E−003	259.22E−003	645.80E−003
f_7	126.03E−003	591.73E−003	93.17E−003	815.79E−003	335.38E+003	2.88E+000	168.29E−003	1.03E+000	172.11E−003	1.45E+003
f_8	259.45E+000	983.97E+000	10.19E+000	149.74E+000	1.92E+003	3.58E+003	57.08E+003	40.14E+003	480.31E+000	1.87E+003

4.4. TSR in continuous problems

In this section, the performance of the proposed algorithm in solving the continuous problems test. For this purpose, the TSR algorithm is first measured alone and with different initial populations. The TSR algorithm then compares with several continuous algorithms using 22 functions, and the results present in the form of graphs and tables. Then the proposed method is compared with continuous algorithms using three practical problems.

4.4.1. Standard function by TSR

In Section 4.2, 22 functions are introduced, including eight unimodal functions, seven multimodal functions, and seven fixed-dimension multimodal functions. Table 7 details the unimodal functions, and Fig. 9 illustrates their 2D figures. Table 8 shows the multimodal functions in detail, and their figures are shown in Fig. 10. Table 9 also details the fixed-dimension multimodal functions, and their image can be seen in Fig. 11. Fig. 20 shows the effect of the initial population number on TSR responses for 22 functions. In these graphs, the effect of the initial population value on the answers can be seen. The initial population in this experiment is 10, 30, 50, 80, and 100, respectively, and the algorithm terminates during 1000 iterations, and the results plotted. Indeed, increasing the initial population leads to better responses and population diversity. Of course, due to the inherent strength of the TSR algorithm, which it has inherited from the vastness and power of the jungle, this algorithm also works very powerfully with a small population, which is quite evident in the diagrams.

4.4.2. TSR and other continuous algorithms comparison

In this section, using the functions listed in Tables 7–9, the TSR algorithm is compared with several continuous metaheuristic algorithms. GA, HS, ICA, GWO, MFO, MVO, PSO, SCA, WOA algorithms have used for these comparisons. Table 10 lists the parameter values of the algorithms. In these experiments, for algorithms, the initial population is 100, and the repetition is 1000. Fig. 21 shows the results of these comparisons. The red color chart, which is bolder than the rest of the lines, is for the TSR. In most cases, the TSR reaches the minimum value much faster than other algorithms. Tables 12–14 present the values of these comparisons in numerical detail. In Table 15, the algorithms rank according to experiments. Each algorithm receives a score according to its rank. When the results of algorithms are equal, the score is divided between them, finally, these scores added together, and the best score is the lowest number. As can be seen from the graphs, the TSR algorithm moves with a steeper slope than other algorithms towards minimization. By increasing the repetition of TSR answers, it moves with a steeper slope than other methods and achieves a better answer. The TSR algorithm uses GR to find areas where there is high hope for improvement. In the TSR algorithm, while the hope for local responses is high, exploration continues elsewhere. This algorithm does not wait for local responses to weaken. This balance in exploration and exploitation makes the TSR algorithm closer to the optimal solution faster than other algorithms.

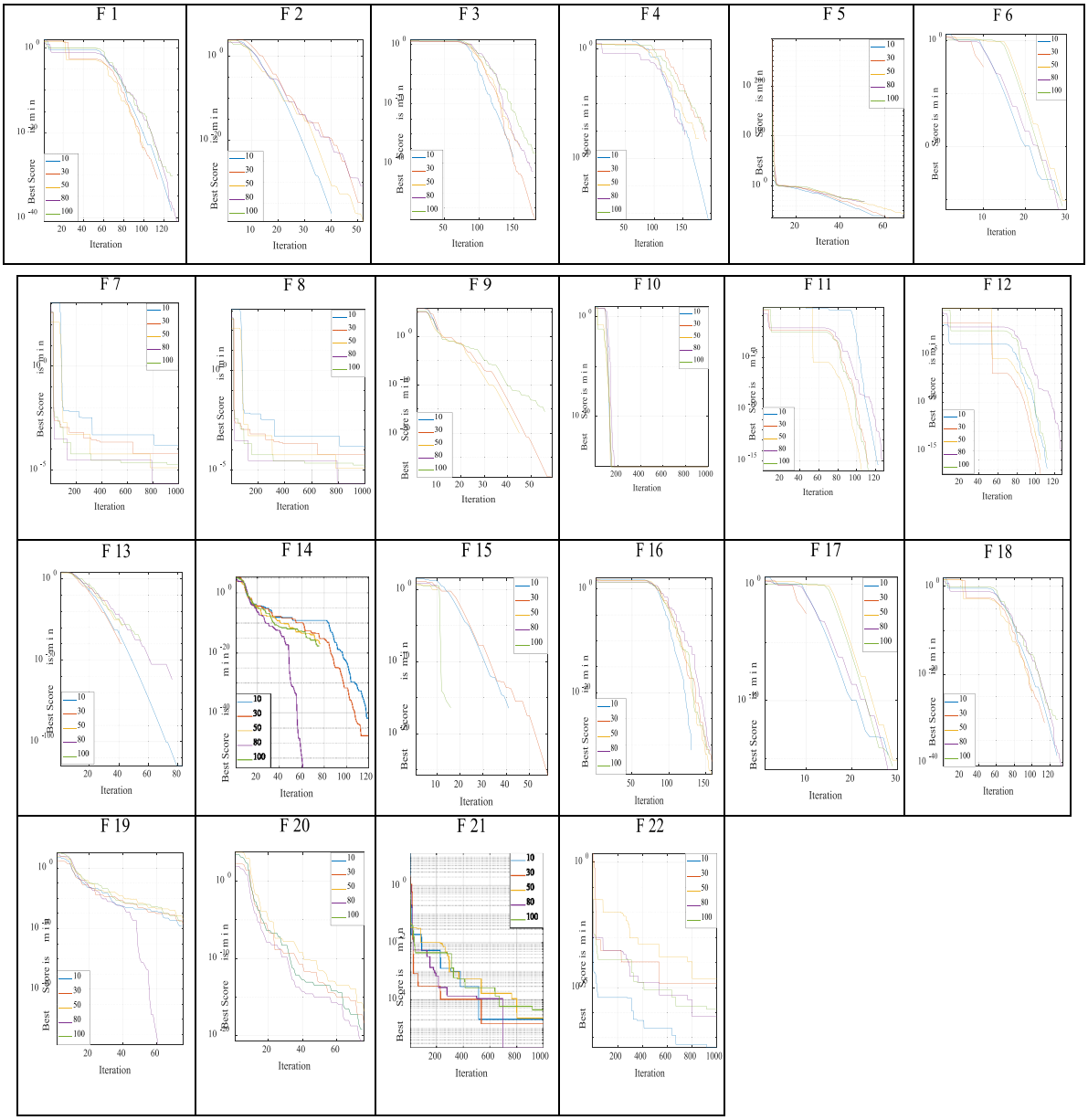

Fig. 20. Influence of initial population on algorithm responses in continuous functions.

4.4.3. Robot path planning problem

In this problem, the robot must move from the origin to the target, which is full of obstacles. The goal is to find the shortest path between the two points of source and destination [22]. In this case, the diameter of these obstacles is random. For this experiment, 40 obstacles with random coordinates create. This experiment implements with an initial population of 50 and a replication rate of 100. Table 16 shows the numerical values obtained from this experiment. The values in the best column indicate that the TSR algorithm has achieved the shortest path compared to other algorithms. The mean values of the table represent the logical process of the algorithm. Fig. 24 shows the problem-solving process by algorithms graphically. As can be seen, before the tenth iteration, the TSR algorithm achieved the best answer, which had a better performance in this regard compared to other algorithms.

Table 13
Results of multimodal benchmark functions.

	TSR		GA		GWO		HS		ICA	
F	Ave	Std	Ave	Std	Ave	Std	Ave	Std	Ave	Std
f_9	318.09E_003	4.15E+000	61.98E−003	995.77E+003	1.02E+000	10.73E+000	1.51E+000	29.54E+000	12.94E−003	13.99E−003
f_{10}	38.06E−003	395.49E−003	116.28E−003	183.07E+003	173.145E03	593.33E−003	215.09E003	49.367E−003	283.81E−003	524.35E−003
f_{11}	9.31E−003	121.53E−003	1.14E−003	35.14E−003	16.29E−003	190.14E−003	128.76E−003	1.94E+000	12.69E−003	68.83E−003
f_{12}	3.42E−003	17.72E−003	42.16E−003	4.82E−003	100.21E−003	56.79E−003	43.64E−003	22.67E−003	42.22E−003	5.511E−003
f_{13}	−28.69E+003	546.88E+000	−7.38E+003	719.78E+000	−3.46E+003	692.33E+000	−19.72E+000	297.35E−003	−2.84E+003	439.68E+003
f_{14}	485.56E−003	2.99E+000	1.284E+000	2.14E+000	777.40E−003	3.165E+000	110.99E+003	630.17E−003	212.76E−003	1.54E+000
f_{15}	454.74E−003	12.19E+000	1.94E+000	15.25E+000	10.50E+000	61.08E+000	18.52E−003	27.03E−003	56.12E−003	226.05E−003
	MVO		PSO		SCA		WOA		MFO	
F	Ave	Std	Ave	Std	Ave	Std	Ave	Std	Ave	Ave
f_9	37.18E−003	250.90E−003	5.86E−003	24.68E−003	460.80E+003	1.16E+003	447.59E−003	4.08E+003	100.55E−003	484.72E−000
f_{10}	254.94E+003	351.30E−003	122.59E003	176.36E−003	221.68E−003	402.61E+003	150.65E−003	247.36E−003	272.37E−003	453.75E−003
f_{11}	6.28E−003	84.85E−003	7.66E−003	54.83E−003	7.65E−003	140.77E+003	4.05E−003	94.15E−003	9.19E−003	69.405E−003
f_{12}	93.61E−003	13.22E−003	42.96E−003	5.17E−003	117.88E−003	33.84E+003	11.26E−003	38.37E−003	94.45E−003	36.80E−003
f_{13}	−2.9E+003	375.19E+000	−1.61E+003	56.89E+000	−2.04E+003	91.63E+000	−11.96E+003	654.38E+000	−2.68E+003	176.96E+000
f_{14}	5.02E+000	2.83E+000	338.35E−003	1.45E+000	3.23E+000	6.23E+000	254.29E−003	1.625E+000	1.27E+000	3.81E+000
f_{15}	2.86E+000	7.35E+000	97.61E−003	310.44E−003	7.28E+000	21.33E+003	5.04E+000	47.76E+000	4.36E+000	19.81E+000

Table 14
Results of fixed dimension multimodal benchmark functions.

	TSR		GA		GWO		HS		ICA	
F	Ave	Std	Ave	Std	Ave	Std	Ave	Std	Ave	Std
f_{16}	8.02E−003	2.93E−003	5.15E−003	3.45E−003	3.66E−003	7.60E−003	5.16E−003	3.48E−003	5.32E−003	2.43E−003
f_{17}	−1.86E+000	2.65E+003	−1.08E+000	3.22E+003	−1.06E+000	7.03E−000	−1.02E−000	3.65E−000	−1.81E−000	3.24E+000
f_{18}	−3.67E+003	4.89E+000	−3.30E+000	4.37E−001	−3.84E+000	1.52E−003	−3.21E+000	9.69E−000	−3.02E+001	1.39E+003
f_{19}	1.09E−003	5.11E+000	7.36E+003	2.31E−003	−3.27E+000	7.21E−000	−2.11E+000	1.61E−000	−2.12E−000	1.21E+000
f_{20}	−1.08E−000	5.26E+000	−1.00E+000	4.11E−003	−9.61E+000	1.51E+000	−7.31E+000	1.21E+000	−5.02E−001	2 .14E−000
f_{21}	−3.89E+000	1.86E−000	−3.30E+000	4.37E−000	−1.04E+000	2.71E−000	−1.0E+000	2.81E−000	−2.655E−000	2.11E−000
f_{22}	−9.91E−000	2.36E−000	−2.39E+000	4.71E−000	−1.51E+000	1.81E−000	−2.61E+000	1.91E+000	−2.42E−000	1.65E−000
	MVO		PSO		SCA		WOA		MFO	
F	Ave	Std	Ave	Std	Ave	Std	Ave	Std	Ave	Ave
f_{16}	7.12E−003	1.28E−003	9.94E−003	2.64E−003	1.01E−003	3.78E+003	2.15E−003	1.49E−003	1.59E−003	3.52E−003
f_{17}	−1.03E+000	4.74E+003	−1.03E+000	0.12E+000	−1.03E+000	3.22E−000	−1.84E−000	2.87E−000	−1.03E+000	0.02E+000
f_{18}	−3.75E+000	3.52E−002	3.91E+000	3.308E+000	−3.86E+000	4.37E+000	−3.77E−000	3.11E−000	−3.86E+000	3.16E−003
f_{19}	−3.31E+000	3.31E−000	−3.31E+000	2.61E−000	−2.81E+000	3.71E+000	−3.31E+000	7.21E−000	−3.23E+000	6.65E−002
f_{20}	−7.31E+000	2.10E+000	−7.51E+000	2.71E+000	−2.21E+000	1.80E+000	−9.61E+000	1.51E+000	−6.20E+000	3.52E+000
f_{21}	−8.51E+000	3.02E+000	−8.51E+000	3.08E+000	−3.91E+000	3.02E+000	−1.04E+000	2.71E−000	−7.59E+000	3.20E+000
f_{22}	−8.42E+000	3.11E+000	−9.91E+000	2.51E+000	−4.91E+000	1.91E+000	−7.11E+000	1.81E−000	−7.51E+000	3.86E+000

Table 15
Ranking Algorithms based on least good.

Function	Rank order	TSR	GA	GWO	HS	ICA	MFO	MVO	PSO	SCA	WOA
$f1$	TSR<PSO<WOA<MFO<SCA<GWO<ICA <GA<HS<MVO	**1**	8	6	9	7	4	10	2	5	3
$f2$	TSR< PSO<WOA<ICA<MVO<MFO<GWO<HS<SCA<GA	**1**	10	7	8	4	6	5	2	9	3
$f3$	TSR<PSO<WOA< ICA<MVO<MFO<GWO<GA<SCA<HS	**1**	8	7	10	4	6	5	2	9	3
$f4$	TSR<PSO<WOA< MVO<MFO<GWO<ICA<HS <GA<SCA	**1**	9	6	8	7	5	4	2	10	3
$f5$	TSR<PSO<WOA< MFO<MVO<GWO<SCA<GA<HS<ICA	**1**	8	6	9	10	4	5	2	7	3
$f6$	TSR<WOA<SCA<GWO<ICA<HS<PSO<GA <PSO <MFO	**1**	8	4	6	5	10	9	7	3	2
$f7$	TSR<SCA<PSO<MFO<GWO<WOA<ICA<MVO<GA<HS	**1**	9	5	10	7	4	8	2	3	6
$f8$	TSR<PSO<ICA<SCA<MFO<HS<GWO<MVO<WOA<GA	**1**	10	7	6	3	5	8	2	4	9
$f9$	TSR<SCA<MVO<PSO<GWO<GA<MFO<MFP<ICA<HS	**1**	7	5	10	9	8	3	4	2	6
$f10$	TSR<GWO<PSO<WOA<ICA<HS≈MVO≈SCA≈MFO≈GA	**1**	7.5	2	7.5	5	7.5	7.5	3	7.5	4
$f11$	TSR<MFO<PSO<SCA<WOA<GWO<GA<HS<MVO<ICA	**1**	7	6	8	10	2	9	3	4	5
$f12$	TSR<WOA<GA≈HS≈PSO≈ICA<GWO≈MFO≈MVO<SCA	**1**	4.5	8	4.5	4.5	8	8	4.5	10	2
$f13$	TSR<WOA<MVO<GWO<GA<ICA<MFO<SCA<PSO<HS	**1**	5	4	10	6	7	3	9	8	2
$f14$	TSR≈WOA≈PSO< MFO<ICA<SCA<GWO<HS<GA<MVO	**1.5**	9	7	8	5	4	10	**1.5**	6	**1.5**
$f15$	TSR<WOA<GWO<SCA<PSO<HS<ICA<GA<MFO<MVO	**1**	8	3	6	7	9	10	5	4	2
$f16$	TSR<PSO<WOA<MFO<SCA<GWO<ICA<HS<GA<MVO	**1**	9	6	8	7	4	10	2	5	3
$f17$	TSR<PSO<WOA<MFO<GA<GWO<ICA<HS≈SCA<MVO	**1**	5	6	8.5	7	4	10	2	8.5	3
$f18$	TSR<PSO<WOA<MFO<SCA<GWO<ICA<HS<GA <MVO	**1**	9	6	8	7	4	10	2	5	3
$f19$	TSR<PSO<WOA<MFO<SCA<GWO<ICA<HS<GA<MVO	**1**	9	6	8	7	4	10	2	5	3
$f20$	TSR<PSO<ICA<WOA<MVO<SCA<HS<GWO<MFO<GA	**1**	10	8	7	3	9	5	2	6	4
$f21$	TSR<PSO<ICA<WOA<MVO<SCA<HS<GWO<MFO<GA	**1**	10	8	7	3	9	5	2	6	4
$f22$	TSR<PSO<ICA<WOA<MVO<SCA<HS<GWO<MFO<GA	**1**	10	8	7	3	9	5	2	6	4
	TOTAL	**22.5**	183	131	173.5	130.5	132.5	159.5	65	133	78.5
	AVERAGE	**≈1**	8.31	5.95	7.88	5.93	6.01	7.25	2.95	6.04	3.56
	RANK	**1**	10	5	9	4	6	8	2	7	3

4.4.4. Feature selection problem

Feature selection is one of the most critical issues in machine learning. In this case, an attempt made to find potential features for classification. These features improve classification accuracy. Finding these features is an NP-hard problem. The Ionosphere database has used for this experiment [44]. For implementations, the initial population is ten, and the number of iteration is 100. In this experiment, the goal is to reach the minimum. Table 17 shows

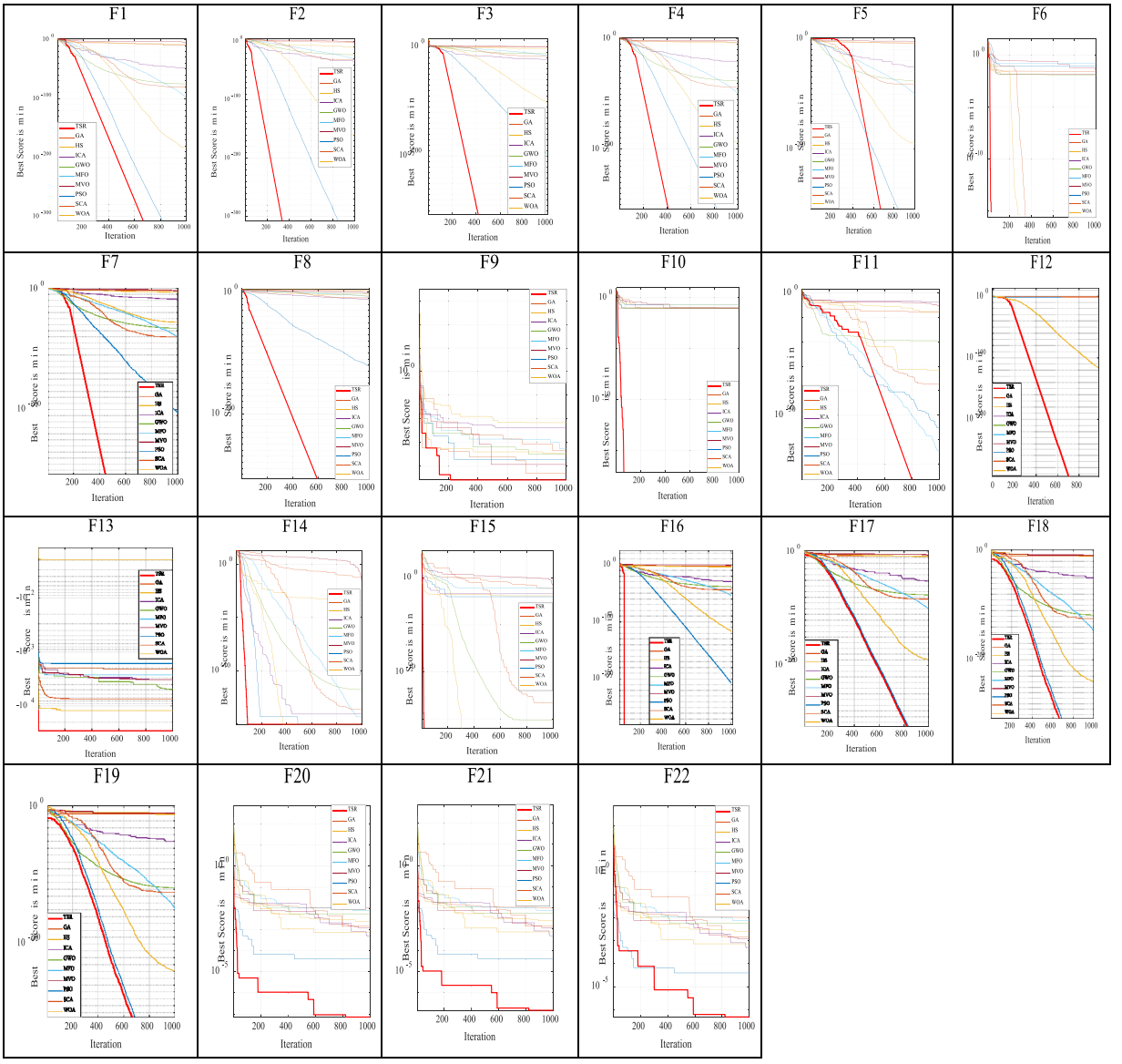

Fig. 21. Comparison of the TSR with other continuous algorithms in solving standard functions.

the numerical values of the results for a better comparison. The results of Table 17 show that the TSR algorithm has achieved the best answer compared to other algorithms. Fig. 25 shows the performance of the algorithms in the form of a graph. As can be seen from the 30th iteration onwards, the TSR algorithm retains the best answer until the last iteration.

4.4.5. Image's color clustering cost problem

In this case, the color clustering of an image is supposed to happen at the lowest cost. This case is one of the most NP-hard problems in image processing. An image receives as input, and color clustering should do at the lowest cost. The cost of clustering obtains according to Eq. (9). The initial population size is 30, and the number of replicates is 100. Table 18 shows the numerical values of this experiment. According to the values of the best column, the TSR algorithm has the best result compared to other algorithms, and the lowest Mean value belongs to the TSR algorithm. The TSR algorithm, according to the diagram in Fig. 26, retains the best answer until the

Table 16
Robot path planning implementations results.

Algorithm	Worst	Best	Median	Std.Dev.	Mean
TSR	163.83	**20.29**	**20.74**	19.16	**25.04**
GA	104.69	37.87	88.57	19.48	76.83
HS	134.68	34.80	64.58	**14.90**	60.90
ICA	137.83	29.23	31.06	19.20	37.37
GWO	154.67	34.65	36.21	19.09	42.15
MFO	160.39	29.48	31.19	20.58	37.56
MVO	144.65	26.59	28.13	18.56	33.87
PSO	133.83	25.74	25.92	15.67	30.09
SCA	164.40	28.29	30.07	21.40	36.69
WOA	149.45	25.72	27.33	19.46	33.35

Table 17
Feature selection implementations results.

Algorithm	Worst	Best	Median	Std.Dev.	Mean
TSR	0.16	**0.08**	**0.09**	0.01	**0.10**
GA	0.14	0.09	0.10	0.01	**0.10**
HS	0.18	0.10	0.11	0.01	0.11
ICA	0.18	0.10	0.11	0.01	0.12
GWO	0.20	0.10	0.11	0.01	0.12
MFO	0.16	0.13	0.13	**0.00**	0.14
MVO	0.15	0.10	0.10	**0.00**	0.11
PSO	0.14	0.12	0.12	**0.00**	0.12
SCA	0.19	0.09	0.10	0.02	0.12
WOA	0.22	0.12	0.13	0.02	0.14

Table 18
Image's color clustering implementations results.

Algorithm	Worst	Best	Median	Std.Dev.	Mean
TSR	7.32E+05	**4.80E+05**	**4.97E+05**	**0.32E+05**	**5.12E+05**
GA	8.48E+05	5.78E+05	6.80E+05	0.61 E+05	6.52E+05
HS	8.40E+05	5.85E+05	5.85E+05	1.02E+05	6.47E+05
ICA	7.81E+05	5.13E+05	5.14E+05	0.72E+05	5.58E+05
GWO	8.58E+05	5.34E+05	5.46E+05	0.85E+05	5.98E+05
MFO	8.87E+05	6.02E+05	6.02E+05	0.77E+05	6.49E+05
MVO	7.85E+05	5.68E+05	5.68E+05	0.46E+05	5.94E+05
PSO	7.28E+05	5.13E+05	5.13E+05	0.34E+05	5.22E+05
SCA	8.33E+05	5.31E+05	5.46E+05	1.18E+05	5.85E+05
WOA	8.97E+05	6.22E+05	6.22E+05	0.62E+05	6.59E+05

Table 19
Welded beam design implementations results.

Algorithm	Worst	Best	Median	Std.Dev.	Mean
TSR	16.23	**1.67**	**2.15**	**0.72**	**2.15**
GA	18.11	1.94	1.98	0.98	2.05
HS	15.42	1.77	1.87	0.63	1.88
ICA	10.63	1.91	1.75	0.46	1.95
GWO	9.16	1.97	1.98	0.72	2.09
MFO	18.02	3.93	2.05	0.98	2.05
MVO	21.34	2.44	2.55	0.87	2.60
PSO	19.33	1.94	2.14	0.91	2.16
SCA	3.10	2.44	2.39	0.02	2.45
WOA	10.74	2.44	2.48	0.69	2.53

Table 20
Speed reducer implementations results.

Algorithm	Worst	Best	Median	Std.Dev.	Mean
TSR	2.78e+03	**2.04e+03**	**3.40e+03**	**1.21e+03**	**3.51e+03**
GA	2.87e+04	3.07e+03	3.34e+03	1.55e+03	3.24e+03
HS	2.63e+04	2.53e+03	3.19e+03	1.02e+03	3.21e+03
ICA	1.88e+04	3.11e+03	3.11e+03	798.51	3.33e+03
GWO	1.56e+04	3.33e+03	3.12e+03	1.24e+03	3.58e+03
MFO	7.07e+04	6.01e+03	6.41e+03	3.36e+03	6.49e+03
MVO	3.62e+04	3.62e+03	3.36e+03	1.48e+03	3.43e+03
PSO	3.09e+04	3.05e+03	3.17e+03	1.45e+03	3.45e+03
SCA	5.19e+03	3.39e+03	4.13e+03	46.15	4.18e+03
WOA	1.74e+04	3.37e+03	4.09e+03	1.19e+03	4.06e+03

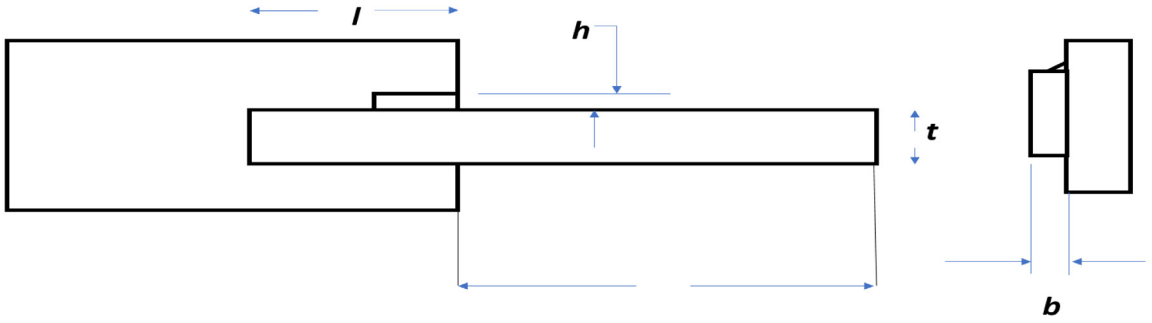

Fig. 22. Schematic view of welded beam design problem.

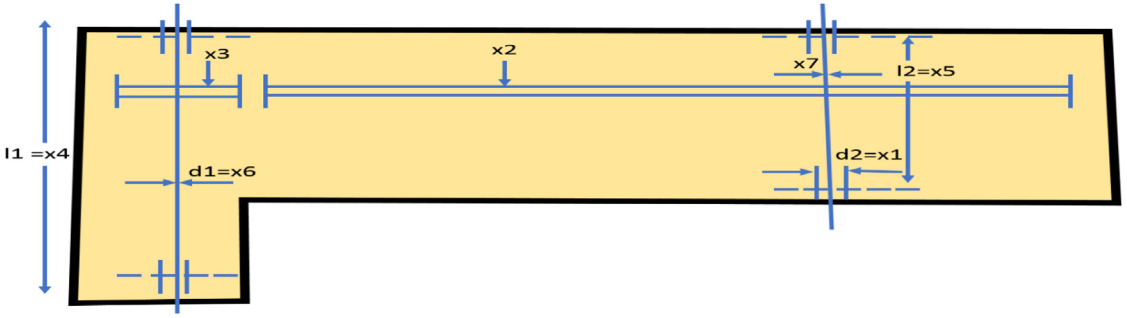

Fig. 23. Schematic view of speed reducer design problem.

end after 30 repetitions.

$$\sum min(dim.\frac{x}{dim}) \tag{9}$$

4.4.6. Welded beam design problem

In this case, the objective is to minimize the fabrication cost of the welded beam shown in Figure 22 [54]. Fig. 22 is inspired by similar articles [45]. The constraints are as follows:

- Shear stress (τ).
- Bending stress in the beam (σ).
- End deflection of the beam (δ).
- Buckling load on the bar (p_c).
- Side constraints.

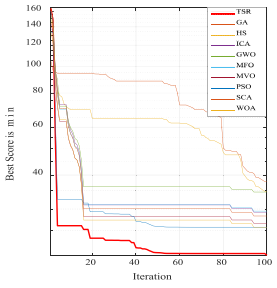

Fig. 24. Comparison of the TSR with other continuous algorithms in Solving Robot Path Planning.

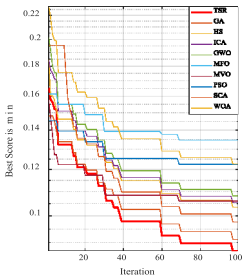

Fig. 25. Comparison of the TSR with other continuous algorithms in Solving Feature selection problem.

As shown in Fig. 22 a rigid member is welded onto a beam and a load is applied to the end of the member. The total cost of making is equal to the effort costs plus the cost of the weld and beam material. So, the aim of this problem is to minimize the fabrication cost of a welded beam. There is four decision variable such as the thickness of weld (h), length of the attached part of the bar (l), the height of the bar (t), and thickness of the bar (b) as shown in Fig. 22. Also, the constraints of this problem are based on shear stress (τ), bending stress in the beam (σ), bucking load on the mathematical formulation is as follows:

Consider

$$
\begin{aligned}
\text{Consider} \quad & \vec{x} = x_1 x_2 x_3 x_4 = [h\, l\, t\, b] \\
\text{Minimize} \quad & f(\vec{x}) = 1.10471 {x_1}^2 x_2 + 0.04811 x_3 x_4 (14.0 + x_2), \\
\text{Subject to} \quad & g_1(\vec{x}) = \tau(\vec{x}) - \tau_{max} \leq 0, \\
& g_2(\vec{x}) = \sigma(\vec{x}) - \sigma_{max} \leq 0, \\
& g_3(\vec{x}) = \delta(\vec{x}) - \delta_{max} \leq 0, \\
& g_4(\vec{x}) = x_1 - x_4 \leq 0, \\
& g_5(\vec{x}) = p - p_c(\vec{x}) \leq 0, \\
& g_6(\vec{x}) = 0.125 - x_1 \leq 0, \\
& g_7(\vec{x}) = 1.10471 {x_1}^2 + 0.04811 x_3 x_4 (14.0 + x_2) - 5.0 \leq 0 \\
\text{Variable range} \quad & 0.1 \leq x_1 \leq 2, \\
& 0.1 \leq x_2 \leq 10, \\
& 0.1 \leq x_3 \leq 10, \\
& 0.1 \leq x_4 \leq 2
\end{aligned}
$$

Table 19 presents the results and shows that TSR finds the minimum optimal cost. The results indicate that TSR converged to the best optimal solution. According to the numerical results of Table 19 and Fig. 27, the TSR algorithm has obtained the best optimal answer compared to other algorithms. According to the diagram, from the 700th iteration onwards, the best result belongs to the proposed algorithm.

4.4.7. Speed reducer design problem

Addressing the issue of speed reducer is a grave challenge. Because it has seven variables, it is not straightforward to design. In this problem, which can be seen in Fig. 23, we are supposed to reach a general minimum by minimizing the weights in the speed reducer, which teaches in the field of bending stress of the gear, stresses in the shafts, surface stress, transverse deflections of the shafts are limited [55]. The variables of this problem are defined as follows:

Minimize

$$\begin{aligned}
\text{Minimize}\quad & f(\vec{x}) = 0.7854x_1{x_2}^2\left(3.3333{x_3}^2 + 14.933x_3 - 43.0934\right) - 1.508x_1\left({x_6}^2 + {x_7}^2\right) \\
& + 7.4777\left({x_6}^3 + {x_7}^3\right) + 0.7854\left(x_4{x_6}^2 + x_5{x_7}^2\right) \\
& g_1(\vec{x}) = \frac{27}{x_1{x_2}^2x_3} - 1 \le 0, \\
& g_2(\vec{x}) = \frac{397.5}{x_1{x_2}^2{x_3}^2} - 1 \le 0, \\
& g_3(\vec{x}) = \frac{1.93{x_4}^3}{x_2{x_6}^4x_3} - 1 \le 0, \\
& g_4(\vec{x}) = \frac{1.93{x_5}^3}{x_2{x_7}^4x_3} - 1 \le 0, \\
& g_5(\vec{x}) = \frac{\left[\left(754\left(\frac{x_4}{x_2x_3}\right)\right)^2 + 16.9 * 10^6\right]^{\frac{1}{2}}}{110{x_6}^3} - 1 \le 0, \\
& g_6(\vec{x}) \frac{\left[\left(754\left(\frac{x_5}{x_2x_3}\right)\right)^2 + 157.5 * 10^6\right]^{\frac{1}{2}}}{85{x_7}^3} - 1 \le 0, \\
& g_7(\vec{x}) = \frac{x_2x_3}{40} - 1 \le 0, \\
& g_8(\vec{x}) = \frac{5x_2}{x_1} - 1 \le 0, \\
& g_9(\vec{x}) = \frac{x_1}{12x_2} - 1 \le 0, \\
& g_{10}(\vec{x}) = \frac{1.5x_6 + 1.9}{x_4} - 1 \le 0, \\
& g_{11}(\vec{x}) = \frac{1.1x_7 + 1.9}{x_5} - 1 \le 0, \\
\text{Where}\quad & 2.6 \le x_1 \le 3.6, 0.7 \le x_2 \le 0.8, 17 \le x_3 \le 28, 7.3 \le x_4 \le 8.3, 7.3 \le x_5 \\
& \le 8.3, 2.9 \le x_6 \le 3.9, 5.0 \le x_7 \le 5.5
\end{aligned}$$

The constraints are as follows:

- Bending stress of the gear teeth.
- Surface stress.
- Transverse deflections of the shafts.
- Stresses in the shafts.

As shown in Fig. 23, we have several variables that include face width (b), the module of teeth (m), number of teeth in the pinion (z), length of the first shaft between bearings ($l1$), length of the second shaft between bearings ($l2$), the diameter of first shaft ($d1$), and the diameter of second shafts ($d2$). Table 20 shows the numerical values obtained from this implementation to compare the methods. Numerical values and the diagram in Fig. 28 show the TSR method's superiority compared to other methods. According to Fig. 28, the TSR algorithm obtains the best optimal solution for this problem during the course of iterations. According to the diagram, the proposed algorithm achieved the best result near the 800th iteration and maintained this superiority until the end.

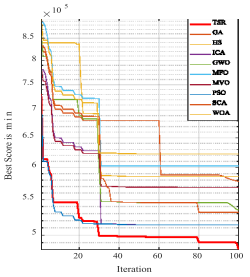

Fig. 26. Comparison of the TSR with other continuous algorithms in Solving image's color clustering problem.. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

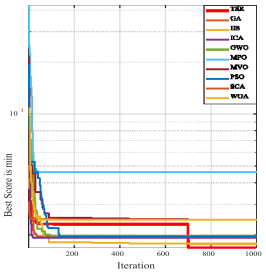

Fig. 27. Comparison of the TSR with other continuous algorithms in welded beam design problem.

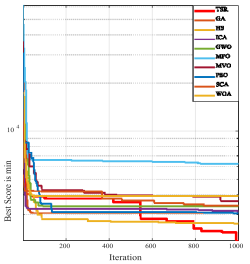

Fig. 28. Comparison of the TSR with other continuous algorithms in speed reduce design problem.

5. Conclusion

Metaheuristic algorithms use to solve problems that do not have a specific solution in an acceptable time. These issues include a large group of scientific issues, which are called NP-hard. The use of metaheuristics in solving NP-hard problems results in acceptable relative responses in a short time. These algorithms belong to the category of optimization algorithms. In other words, in these algorithms, the priority is to achieve the right answer in a short time instead of the best answer in a very long time. A variety of methods of metaheuristic algorithms have introduced, none of which can be superior to the other methods for solving all problems. For this reason, efforts to design metaheuristic methods continue to cover a broader range of issues. In this paper, a new optimization algorithm called Trees social Relationship Optimization (TSR) algorithm presents. The social life of trees is dynamic and based on the hierarchical relationships of trees. The social life of trees includes a collection of trees that work under the supervision of the mother tree. All trees compete and cooperate in a large community. The social life of trees can analyze in two ways. The first is a large community that is supposed to solve a common problem. Second, social groups that are going to compete for a common food source. In this competition and cooperation, the goal is to preserve this community and the survival of the tree community. Following this cooperation and competition, the TSR algorithm consists of two processes: local (sub-jungle) and global. The TSR algorithm practically compares with several discrete and continuous metaheuristic algorithms in solving various problems. The results of practical evaluations showed that the TSR algorithm is capable of solving real and classical NP-hard

problems and achieves optimal solutions. It also performs better in solving many problems and achieves better answers than other metaheuristic algorithms. Studies have also shown that the TSR algorithm is suitable for solving problems with huge data sizes. The proposed algorithm can also implement in parallel. Proper speed and high accuracy of the TSR algorithm make it a good option for optimizing various problems. Therefore, it can say that the proposed algorithm is proper as a suitable optimization algorithm for solving discrete and continuous problems.

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.

References

[1] A new heuristic optimization algorithm: Harmony search -?zong woo geem, joong hoon kim, g.V. Loganathan, 2001, https://journals.sagepub.com/doi/abs/10.1177/003754970107600201 (accessed Aug. 09, 2021).
[2] Algorithm of marriage in honey bees optimization based on the wolf pack search |, IEEE Conf. Publ. IEEE Xplore (2021) https://ieeexplore.ieee.org/abstract/document/4438476/ (accessed Aug. 09, 2021).
[3] Artificial bee colony (ABC) optimization algorithm for solving constrained optimization problems | SpringerLink, 2021, https://link.springer.com/chapter/10.1007/978-3-540-72950-1_77 (accessed Aug. 09, 2021).
[4] Artificial immune systems: A new computational intelligence approach - Leandro Nunes de Castro, Leandro Nunes Castro, Jonathan Timmis - Google Books, 2021, https://books.google.com/books?hl=en&lr=&id=aMFP7p8DtaQC&oi=fnd&pg=PA1&dq=%5B43%5D%09Castro,Leandro+Nunes,Leandro+Nunes+De+Castro,and+Jonathan+Timmis.+Artificial+immune+systems:+a+new+computational+intelligence+approach.+Springer+Science+%26+Business+Media,2002.&ots=zLeqYA3YhQ&sig=sh1mCRu2OzuqAuf1PuHny9aPkjk#v=onepage&q=%5B43%5D%09Castro%2C%20Leandro%20Nunes%2C%20Leandro%20Nunes%20De%20Castro%2C%20and%20Jonathan%20Timmis.%20Artificial%20immune%20systems%3A%20a%20new%20computational%20intelligence%20approach.%20Springer%20Science%20%26%20Business%20Media%2C%202002.&f=false (accessed Aug. 09, 2021).
[5] Biogeography-based optimization |, IEEE J. Mag. IEEE Xplore (2021) https://ieeexplore.ieee.org/abstract/document/4475427 (accessed Aug. 09, 2021).
[6] CiNii articles - modern heuristics techniques for combinatorial problems, 2021, https://ci.nii.ac.jp/naid/10010555508/ (accessed Aug. 09, 2021).
[7] Evolution strategies – A comprehensive introduction | SpringerLink, 2021, https://link.springer.com/article/10.1023/A:1015059928466 (accessed Aug. 09, 2021).
[8] An experimental study of benchmarking functions for genetic algorithms, Int. J. Comput. Math. 79 (4) (2021) https://www.tandfonline.com/doi/abs/10.1080/0020716021093 9?journalCode=gcom20 (accessed Aug. 09, 2021).
[9] Firefly algorithm, stochastic test functions and design optimisation, Int. J. Bio-Inspired Comput. 2 (2) (2021) https://www.inderscienceonline.com/doi/abs/10.1504/IJBIC.2010.032124 (accessed Aug. 09, 2021).
[10] Firefly-inspired stochastic resonance for spectrum sensing in CR-based IoT communications | SpringerLink, 2021, https://link.springer.com/article/10.1007/s00521-019-04584-0 (accessed Aug. 09, 2021).
[11] Fireworks algorithm for optimization | SpringerLink, 2021, https://link.springer.com/chapter/10.1007/978-3-642-13495-1_44 (accessed Aug. 09, 2021).
[12] Genetic algorithms on JSTOR, 2021, https://www.jstor.org/stable/24939139 (accessed Aug. 09, 2021).
[13] GPU parallelization strategies for metaheuristics: a survey, Int. J. Parallel Emergent Distrib. Syst. 34 (5) (2021) https://www.tandfonline.com/doi/abs/10.1080/17445760.2018.1428969 (accessed Aug. 09, 2021).
[14] Group search optimizer: An optimization algorithm inspired by animal searching behavior |, IEEE J. Mag. IEEE Xplore (2021) https://ieeexplore.ieee.org/abstract/document/5196714 (accessed Aug. 09, 2021).
[15] GSA: A gravitational search algorithm - ScienceDirect, 2021, https://www.sciencedirect.com/science/article/abs/pii/S0020025509001200 (accessed Aug. 09, 2021).
[16] Lexicographic optimization-based clustering search metaheuristic for the multiobjective flexible job shop scheduling problem - Bissoli - 2021 - international transactions in operational research - Wiley Online Library, 2021, https://onlinelibrary.wiley.com/doi/full/10.1111/itor.12745?casa_token=RToRJeTPIEsAAAAA%3AgO0-Kn7fu3AbfIzBuMP9uQU5mXqNDRsjzXFC8UNgAKp3h2MuKvwshUhYuA-m25QmwTw4x_oVmjM (accessed Aug. 09, 2021).
[17] MBO: marriage in honey bees optimization-A haplometrosis polygynous swarming approach |, IEEE Conf. Publ. IEEE Xplore (2021) https://ieeexplore.ieee.org/abstract/document/934391 (accessed Aug. 09, 2021).
[18] Metaheuristics: A bibliography | SpringerLink, 2021, https://link.springer.com/article/10.1007/BF02125421 (accessed Aug. 09, 2021).
[19] Metaheuristics in combinatorial optimization: Overview and conceptual comparison, ACM Comput. Surv. 35 (3) (2021) https://dl.acm.org/doi/abs/10.1145/937503.937505 (accessed Aug. 09, 2021).
[20] Monkey search: A novel metaheuristic search for global optimization, AIP Conf. Proc. 953 (1) (2021) https://aip.scitation.org/doi/abs/10.1063/1.2817338 (accessed Aug. 09, 2021).
[21] Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm - ScienceDirect, 2021, https://www.sciencedirect.com/science/article/abs/pii/S0950705115002580 (accessed Aug. 09, 2021).

[22] A multi-objective PSO-based algorithm for robot path planning |, IEEE Conf. Publ. IEEE Xplore (2021) https://ieeexplore.ieee.org/abstract/document/5472755 (accessed Aug. 09, 2021).
[23] Multi-verse optimizer: a nature-inspired algorithm for global optimization | SpringerLink, 2021, https://link.springer.com/article/10.1007%2Fs00521-015-1870-7 (accessed Aug. 09, 2021).
[24] Multilevel image thresholding using entropy of histogram and recently developed population-based metaheuristic algorithms | SpringerLink, 2021, https://link.springer.com/article/10.1007/s12065-017-0152-y (accessed Aug. 09, 2021).
[25] Mycorrhizal networks: a review of their extent, function, and importance, 2021, https://cdnsciencepub.com/doi/abs/10.1139/b04-116 (accessed Aug. 09, 2021).
[26] A new biologically inspired optimization algorithm | IEEE conference publication | IEEE xplore, 2021, https://ieeexplore.ieee.org/abstract/document/5429852 (accessed Aug. 09, 2021).
[27] A new heuristic optimization algorithm for modeling of proton exchange membrane fuel cell: bird mating optimizer - Askarzadeh - 2013, International Journal of Energy Research - Wiley Online Library. (2021) https://onlinelibrary.wiley.com/doi/full/10.1002/er.2915?casa_token=-gUlMJHn7BMAAAAA%3Az_iLJT9xZN8kvYcYACSVST166RIjRWy8-_jsRA5o5zaUNQNakpbyiHIxE9kqLlazm5qtzO_0OzQ (accessed Aug. 09, 2021).
[28] A new metaheuristic bat-inspired algorithm | SpringerLink, 2021, https://link.springer.com/chapter/10.1007/978-3-642-12538-6_6 (accessed Aug. 09, 2021).
[29] A new version of ant system for subset problems |, IEEE Conf. Publ. IEEE Xplore (2021) https://ieeexplore.ieee.org/abstract/document/782655 (accessed Aug. 09, 2021).
[30] A novel global convergence algorithm: Bee collecting pollen algorithm | SpringerLink, 2021, https://link.springer.com/chapter/10.1007/978-3-540-85984-0_62 (accessed Aug. 09, 2021).
[31] A novel heuristic optimization method: charged system search | SpringerLink, 2021, https://link.springer.com/article/10.1007/s00707-009-0270-4 (accessed Aug. 09, 2021).
[32] Novel multiobjective TLBO algorithms for the feature subset selection problem - ScienceDirect, 2021, https://www.sciencedirect.com/science/article/abs/pii/S0925231218304442 (accessed Aug. 09, 2021).
[33] A novel path planning algorithm based on plant growth mechanism | SpringerLink, 2021, https://link.springer.com/article/10.1007%2Fs00500-016-2045-x (accessed Aug. 09, 2021).
[34] Optimal inspection planning for onshore pipelines subject to external corrosion - ScienceDirect, 2021, https://www.sciencedirect.com/science/article/abs/pii/S0951832013001063 (accessed Aug. 09, 2021).
[35] Optimization by simulated annealing | Science, 2021, https://science.sciencemag.org/content/220/4598/671.abstract (accessed Aug. 09, 2021).
[36] Practical genetic algorithms - randy l. Haupt, sue ellen haupt - google books, 2021, https://books.google.com/books?hl=en&lr=&id=k0jFfsmbtZIC&oi=fnd&pg=PR7&dq=Haupt,+Randy+L.,+and+Sue+Ellen+Haupt.+%22Practical+genetic+algorithms.%22+(2004).+&ots=PS2UNNQCs7&sig=8HmgvHp3Qzs1VAI7rJaVi1PEPHM#v=onepage&q=Haupt%2C%20Randy%20L.%2C%20and%20Sue%20Ellen%20Haupt.%20%22Practical%20genetic%20algorithms.%22%20(2004).&f=false (accessed Aug. 09, 2021).
[37] Principal components analysis by the galaxy-based search algorithm: A novel metaheuristic for continuous optimisation, Int. J. Comput. Sci. Eng. 6 (1–2) (2021) https://www.inderscienceonline.com/doi/abs/10.1504/IJCSE.2011.041221 (accessed Aug. 09, 2021).
[38] Seeding the initial population with feasible solutions in metaheuristic optimization of steel trusses, Eng. Optim. 50 (1) (2021) https://www.tandfonline.com/doi/abs/10.1080/0305215X.2017.1284833 (accessed Aug. 09, 2021).
[39] Seeker optimization algorithm for optimal reactive power dispatch |, 2021, https://ieeexplore.ieee.org/abstract/document/4912428 (accessed Aug. 09, 2021),
[40] A study on metaheuristics approaches for gene selection in microarray data: algorithms, applications and open challenges | SpringerLink, 2021, https://link.springer.com/article/10.1007/s12065-019-00306-6 (accessed Aug. 09, 2021).
[41] A survey on handling computationally expensive multiobjective optimization problems with evolutionary algorithms | SpringerLink, 2021, https://link.springer.com/article/10.1007/s00500-017-2965-0 (accessed Aug. 09, 2021).
[42] Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm | SpringerLink, 2021, https://link.springer.com/article/10.1007/BF00940812 (accessed Aug. 09, 2021).
[43] Thinking capability of saplings growing up algorithm | SpringerLink, 2021, https://link.springer.com/chapter/10.1007/11875581_47 (accessed Aug. 09, 2021).
[44] UCI machine learning repository: Ionosphere data set, 2021, https://archive.ics.uci.edu/ml/datasets/Ionosphere (accessed Aug. 09, 2021).
[45] Useful infeasible solutions in engineering optimization with evolutionary algorithms | SpringerLink, 2021, https://link.springer.com/chapter/10.1007/11579427_66 (accessed Aug. 09, 2021).
[46] World traveling salesman problem, 2021, http://www.math.uwaterloo.ca/tsp/world/index.html (accessed Aug. 09, 2021).
[47] B. Alatas, ACROA: Artificial chemical reaction optimization algorithm for global optimization, Expert Syst. Appl. 38 (10) (2011) 13170–13180, http://dx.doi.org/10.1016/j.eswa.2011.04.126.
[48] A. Askarzadeh, A novel metaheuristic method for solving constrained engineering optimization problems: Crow search algorithm, Comput. Struct. 169 (2016) 1–12, http://dx.doi.org/10.1016/j.compstruc.2016.03.001.
[49] E. Atashpaz-Gargari, C. Lucas, Imperialist competitive algorithm: An algorithm for optimization inspired by imperialistic competition, in: 2007 IEEE Congress on Evolutionary Computation, 2007, pp. 4661–4667, http://dx.doi.org/10.1109/CEC.2007.4425083.
[50] M. Birattari, L. Paquete, T. Stützle, K. Varrentrapp, Classification of metaheuristics and design of experiments for the analysis of components, Tek. Rap. AIDA (2001) 01–05.
[51] C. Blum, J. Puchinger, G.R. Raidl, A. Roli, Hybrid metaheuristics in combinatorial optimization: A survey, Appl. Soft Comput. 11 (6) (2011) 4135–4151, http://dx.doi.org/10.1016/j.asoc.2011.02.032.

[52] I. Boussaïd, J. Lepagnot, P. Siarry, A survey on optimization metaheuristics, Inform. Sci. 237 (2013) 82–117, http://dx.doi.org/10.1016/j.ins.2013.02.041.
[53] M.-Y. Cheng, D. Prayogo, Symbiotic organisms search: A new metaheuristic optimization algorithm, Comput. Struct. 139 (2014) 98–112, http://dx.doi.org/10.1016/j.compstruc.2014.03.007.
[54] C.A. Coello Coello, Use of a self-adaptive penalty approach for engineering optimization problems, Comput. Ind. 41 (2) (2000) 113–127, http://dx.doi.org/10.1016/S0166-3615(99)00046-9.
[55] G. Dhiman, V. Kumar, Spotted hyena optimizer: A novel bio-inspired based metaheuristic technique for engineering applications, Adv. Eng. Softw. 114 (2017) 48–70, http://dx.doi.org/10.1016/j.advengsoft.2017.05.014.
[56] T. Dokeroglu, E. Sevinc, T. Kucukyilmaz, A. Cosar, A survey on new generation metaheuristic algorithms, Comput. Ind. Eng. 137 (2019) 106040, http://dx.doi.org/10.1016/j.cie.2019.106040.
[57] M. Dorigo, M. Birattari, T. Stutzle, Ant colony optimization, IEEE Comput. Intell. Mag. 1 (4) (2006) 28–39, http://dx.doi.org/10.1109/MCI.2006.329691.
[58] H. Du, X. Wu, J. Zhuang, Small-world optimization algorithm for function optimization, in: Advances in Natural Computation, Berlin, Heidelberg, 2006, pp. 264–273, http://dx.doi.org/10.1007/11881223_33.
[59] O.K. Erol, I. Eksin, A new optimization method: Big Bang–Big crunch, Adv. Eng. Softw. 37 (2) (2006) 106–111, http://dx.doi.org/10.1016/j.advengsoft.2005.04.005.
[60] S. Farzi, Efficient job scheduling in grid computing with modified artificial fish swarm algorithm, Int. J. Comput. Theory Eng. 1 (1) (2009) 13.
[61] A.M. Fathollahi-Fard, M. Hajiaghaei-Keshteli, R. Tavakkoli-Moghaddam, The social engineering optimizer (SEO), Eng. Appl. Artif. Intell. 72 (2018) 267–293, http://dx.doi.org/10.1016/j.engappai.2018.04.009.
[62] V. Filipović, A. Kartelj, D. Matić, An electromagnetism metaheuristic for solving the maximum betweenness problem, Appl. Soft Comput. 13 (2) (2013) 1303–1313, http://dx.doi.org/10.1016/j.asoc.2012.10.015.
[63] R.A. Formato, Central force optimization, Prog. Electromagn. Res. 77 (1) (2007) 425–491.
[64] A.H. Gandomi, Interior search algorithm (ISA): A novel approach for global optimization, ISA Trans. 53 (4) (2014) 1168–1183, http://dx.doi.org/10.1016/j.isatra.2014.03.018.
[65] A.H. Gandomi, A.H. Alavi, Krill herd: A new bio-inspired optimization algorithm, Commun. Nonlinear Sci. Numer. Simul. 17 (12) (2012) 4831–4845, http://dx.doi.org/10.1016/j.cnsns.2012.05.010.
[66] N. Ghorbani, E. Babaei, Exchange market algorithm, Appl. Soft Comput. 19 (2014) 177–187, http://dx.doi.org/10.1016/j.asoc.2014.02.006.
[67] F. Glover, Tabu search—Part I, ORSA J. Comput. 1 (3) (1989) 190–206, http://dx.doi.org/10.1287/ijoc.1.3.190.
[68] F. Glover, Tabu search—Part II, ORSA J. Comput. 2 (1) (1990) 4–32, http://dx.doi.org/10.1287/ijoc.2.1.4.
[69] W.J.S. Gomes, A.T. Beck, R.H. Lopez, L.F.F. Miguel, A probabilistic metric for comparing metaheuristic optimization algorithms, Struct. Saf. 70 (2018) 59–70, http://dx.doi.org/10.1016/j.strusafe.2017.10.006.
[70] A. Hatamlou, Black hole: A new heuristic optimization approach for data clustering, Inform. Sci. 222 (2013) 175–184, http://dx.doi.org/10.1016/j.ins.2012.08.023.
[71] M. Heil, R. Karban, Explaining evolution of plant communication by airborne signals, Trends Ecol. Evol. 25 (3) (2010) 137–144, http://dx.doi.org/10.1016/j.tree.2009.09.010.
[72] J.H. Holland, Genetic algorithms, Sci. Am. 267 (1) (1992) 66–73.
[73] A. Husseinzadeh Kashan, An efficient algorithm for constrained global optimization and application to mechanical engineering design: League championship algorithm (LCA), Comput.-Aided Des. 43 (12) (2011) 1769–1792, http://dx.doi.org/10.1016/j.cad.2011.07.003.
[74] A. Kaveh, N. Farhoudi, A new optimization method: Dolphin echolocation, Adv. Eng. Softw. 59 (2013) 53–70, http://dx.doi.org/10.1016/j.advengsoft.2013.03.004.
[75] A. Kaveh, M. Khayatazad, A new meta-heuristic method: Ray optimization, Comput. Struct. 112–113 (2012) 283–294, http://dx.doi.org/10.1016/j.compstruc.2012.09.003.
[76] A. Kaveh, V.R. Mahdavi, Colliding bodies optimization: A novel meta-heuristic method, Comput. Struct. 139 (2014) 18–27, http://dx.doi.org/10.1016/j.compstruc.2014.04.005.
[77] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of ICNN'95 - International Conference on Neural Networks, 1995, pp. 1942–1948, http://dx.doi.org/10.1109/ICNN.1995.488968, 4, 4.
[78] J.R. Koza, J.R. Koza, Genetic Programming: On the Programming of Computers by Means of Natural Selection, MIT Press, 1992.
[79] Y. Labbi, D.B. Attous, H.A. Gabbar, B. Mahdad, A. Zidan, A new rooted tree optimization algorithm for economic dispatch with valve-point effect, Int. J. Electr. Power Energy Syst. 79 (2016) 298–311, http://dx.doi.org/10.1016/j.ijepes.2016.01.028.
[80] X. Li, A. Engelbrecht, M.G. Epitropakis, Special session and competition on niching methods for multimodal function optimization, Benchmark Funct. CEC'RMIT Univ. Evol. Comput. Mach. Learn. Group Aust. Tech. Rep. (2013).
[81] M.A. Lopes Silva, S.R. de Souza, M.J. Freitas Souza, M.F. de França Filho, Hybrid metaheuristics and multi-agent systems for solving optimization problems: A review of frameworks and a comparative analysis, Appl. Soft Comput. 71 (2018) 433–459, http://dx.doi.org/10.1016/j.asoc.2018.06.050.
[82] S. Mahdavi, M.E. Shiri, S. Rahnamayan, Metaheuristics in large-scale global continues optimization: A survey, Inform. Sci. 295 (2015) 407–428, http://dx.doi.org/10.1016/j.ins.2014.10.042.
[83] A.R. Mehrabian, C. Lucas, A novel numerical optimization algorithm inspired from weed colonization, Ecol. Inform. 1 (4) (2006) 355–366, http://dx.doi.org/10.1016/j.ecoinf.2006.07.003.
[84] F. Merrikh-Bayat, The runner-root algorithm: A metaheuristic for solving unimodal and multimodal optimization problems inspired by runners and roots of plants in nature, Appl. Soft Comput. 33 (2015) 292–303, http://dx.doi.org/10.1016/j.asoc.2015.04.048.

[85] Z. Michalewicz, Genetic Algorithms Data Structures=Evolution Programs, Springer Science & Business Media, 2013.
[86] S. Mirjalili, SCA: A Sine Cosine algorithm for solving optimization problems, Knowl.-Based Syst. 96 (2016) 120–133, http://dx.doi.org/10.1016/j.knosys.2015.12.022.
[87] S. Mirjalili, A. Lewis, The whale optimization algorithm, Adv. Eng. Softw. 95 (2016) 51–67, http://dx.doi.org/10.1016/j.advengsoft.2016.01.008.
[88] S. Mirjalili, S.M. Mirjalili, A. Lewis, Grey wolf optimizer, Adv. Eng. Softw. 69 (2014) 46–61, http://dx.doi.org/10.1016/j.advengsoft.2013.12.007.
[89] F.F. Moghaddam, R.F. Moghaddam, M. Cheriet, Curved space optimization: A random search based on general relativity theory, 2012, arXiv12082214 Cs, Aug. 2012, Accessed: Aug. 09, 2021. [Online]. Available: http://arxiv.org/abs/1208.2214.
[90] N. Moosavian, B. Kasaee Roodsari, Soccer league competition algorithm: A novel meta-heuristic algorithm for optimal design of water distribution networks, Swarm Evol. Comput. 17 (2014) 14–24, http://dx.doi.org/10.1016/j.swevo.2014.02.002.
[91] H. Murase, Finite element inverse analysis using a photosynthetic algorithm, Comput. Electron. Agric. 29 (1) (2000) 115–123, http://dx.doi.org/10.1016/S0168-1699(00)00139-3.
[92] F. Neumann, C. Witt, Combinatorial optimization and computational complexity, in: F. Neumann, C. Witt (Eds.), Bioinspired Computation in Combinatorial Optimization: Algorithms and their Computational Complexity, Springer, Berlin, Heidelberg, 2010, pp. 9–19, http://dx.doi.org/10.1007/978-3-642-16544-3_2.
[93] R. Oftadeh, M.J. Mahjoob, M. Shariatpanahi, A novel meta-heuristic optimization algorithm inspired by group hunting of animals: Hunting search, Comput. Math. Appl. 60 (7) (2010) 2087–2098, http://dx.doi.org/10.1016/j.camwa.2010.07.049.
[94] W.-T. Pan, A new fruit fly optimization algorithm: Taking the financial distress model as an example, Knowl.-Based Syst. 26 (2012) 69–74, http://dx.doi.org/10.1016/j.knosys.2011.07.001.
[95] R. Pellerin, N. Perrier, F. Berthaut, A survey of hybrid metaheuristics for the resource-constrained project scheduling problem, European J. Oper. Res. 280 (2) (2020) 395–416, http://dx.doi.org/10.1016/j.ejor.2019.01.063.
[96] W. Peres, I.C. Silva Júnior, J.A. Passos Filho, Gradient based hybrid metaheuristics for robust tuning of power system stabilizers, Int. J. Electr. Power Energy Syst. 95 (2018) 47–72, http://dx.doi.org/10.1016/j.ijepes.2017.08.014.
[97] R. Pilla, A.T. Azar, T.S. Gorripotu, Impact of flexible AC transmission system devices on automatic generation control with a metaheuristic based fuzzy PID controller, Energies 12 (21) (2019) http://dx.doi.org/10.3390/en12214193, Art. (21).
[98] P.C. Pinto, T.A. Runkler, J.M.C. Sousa, Wasp swarm algorithm for dynamic MAX-sat problems, in: Adaptive and Natural Computing Algorithms, Berlin, Heidelberg, 2007, pp. 350–357, http://dx.doi.org/10.1007/978-3-540-71618-1_39.
[99] S.K. Prabhakar, H. Rajaguru, S.-W. Lee, Metaheuristic-based dimensionality reduction and classification analysis of PPG signals for interpreting cardiovascular disease, IEEE Access 7 (2019) 165181–165206, http://dx.doi.org/10.1109/ACCESS.2019.2950220.
[100] M.H. Qais, H.M. Hasanien, S. Alghuwainem, Whale optimization algorithm-based sugeno fuzzy logic controller for fault ride-through improvement of grid-connected variable speed wind generators, Eng. Appl. Artif. Intell. 87 (2020) 103328, http://dx.doi.org/10.1016/j.engappai.2019.103328.
[101] F. Ramezani, S. Lotfi, Social-based algorithm (SBA), Appl. Soft Comput. 13 (5) (2013) 2837–2856, http://dx.doi.org/10.1016/j.asoc.2012.05.018.
[102] G. Reinelt, The Traveling Salesman: Computational Solutions for TSP Applications, Springer, 2003.
[103] M. Roper, E. Dressaire, Fungal biology: Bidirectional communication across fungal networks, Curr. Biol. 29 (4) (2019) R130–R132, http://dx.doi.org/10.1016/j.cub.2019.01.011.
[104] M. Roth, Termite: A swarm intelligent routing algorithm for mobile wireless ad-hoc networks, 2005, Accessed: Aug. 09, 2021. [Online]. Available: https://ecommons.cornell.edu/handle/1813/240.
[105] A. Sadollah, A. Bahreininejad, H. Eskandar, M. Hamdi, Mine blast algorithm: A new population based algorithm for solving constrained engineering optimization problems, Appl. Soft Comput. 13 (5) (2013) 2592–2612, http://dx.doi.org/10.1016/j.asoc.2012.11.026.
[106] M.-A. Selosse, F. Richard, X. He, S.W. Simard, Mycorrhizal networks: des liaisons dangereuses? Trends Ecol. Evol. 21 (11) (2006) 621–628, http://dx.doi.org/10.1016/j.tree.2006.07.003.
[107] Y. Shiqin, J. Jianjun, Y. Guangxing, A dolphin partner optimization, in: 2009 WRI Global Congress on Intelligent Systems, 2009, pp. 124–128, http://dx.doi.org/10.1109/GCIS.2009.464, 1.
[108] S.W. Simard, The foundational role of mycorrhizal networks in self-organization of interior douglas-fir forests, For. Ecol. Manag. 258 (2009) S95–S107, http://dx.doi.org/10.1016/j.foreco.2009.05.001.
[109] S.W. Simard, Mycorrhizal networks facilitate tree communication, learning, and memory, in: F. Baluska, M. Gagliano, G. Witzany (Eds.), Memory and Learning in Plants, Springer International Publishing, Cham, 2018, pp. 191–213, http://dx.doi.org/10.1007/978-3-319-75596-0_10.
[110] S.W. Simard, K.J. Beiler, M.A. Bingham, J.R. Deslippe, L.J. Philip, F.P. Teste, Mycorrhizal networks: Mechanisms, ecology and modelling, Fungal Biol. Rev. 26 (1) (2012) 39–60, http://dx.doi.org/10.1016/j.fbr.2012.01.001.
[111] S.W. Simard, T. Blenner-Hassett, I.R. Cameron, Pre-commercial thinning effects on growth, yield and mortality in even-aged paper birch stands in british columbia, For. Ecol. Manag. 190 (2) (2004) 163–178, http://dx.doi.org/10.1016/j.foreco.2003.09.010.
[112] S.W. Simard, D.L. Sachs, A. Vyse, L.L. Blevins, Paper birch competitive effects vary with conifer tree species and stand age in interior british columbia forests: implications for reforestation policy and practice, For. Ecol. Manag. 198 (1) (2004) 55–74, http://dx.doi.org/10.1016/j.foreco.2004.03.036.
[113] E.-G. Talbi, Metaheuristics: From Design to Implementation, John Wiley & Sons, 2009.
[114] S. Wang, Y. Zhao, J. Xu, J. Yuan, C.-H. Hsu, Edge server placement in mobile edge computing, J. Parallel Distrib. Comput. 127 (2019) 160–168, http://dx.doi.org/10.1016/j.jpdc.2018.06.008.
[115] B. Webster, P.J. Bernhard, A local search optimization algorithm based on natural principles of gravitation, 2003, Accessed: Aug. 09, 2021. [Online]. Available: https://repository.lib.fit.edu/handle/11141/117.

[116] D.H. Wolpert, W.G. Macready, No free lunch theorems for optimization, IEEE Trans. Evol. Comput. 1 (1) (1997) 67–82, http://dx.doi.org/10.1109/4235.585893.
[117] D.H. Wolpert, W.G. Macready, No free lunch theorems for optimization, IEEE Trans. Evol. Comput. 1 (1) (1997) 67–82, http://dx.doi.org/10.1109/4235.585893.
[118] G. Wu, R. Mallipeddi, P.N. Suganthan, Ensemble strategies for population-based optimization algorithms – A survey, Swarm Evol. Comput. 44 (2019) 695–711, http://dx.doi.org/10.1016/j.swevo.2018.08.015.
[119] X.-S. Yang, Engineering Optimization: An Introduction with Metaheuristic Applications, John Wiley & Sons, 2010.
[120] X.-S. Yang, S. Deb, Cuckoo search via Lévy flights, in: 2009 World Congress on Nature Biologically Inspired Computing (NaBIC), 2009, pp. 210–214, http://dx.doi.org/10.1109/NABIC.2009.5393690.
[121] J. Zhang, M. Xiao, L. Gao, Q. Pan, Queuing search algorithm: A novel metaheuristic algorithm for solving engineering optimization problems, Appl. Math. Model. 63 (2018) 464–490, http://dx.doi.org/10.1016/j.apm.2018.06.036.
[122] Y. Zhou, Y. Zhou, Q. Luo, M. Abdel-Basset, A simplex method-based social spider optimization algorithm for clustering analysis, Eng. Appl. Artif. Intell. 64 (2017) 67–82, http://dx.doi.org/10.1016/j.engappai.2017.06.004.