International Journal of Engineering Inventions
e-ISSN: 2278-7461, p-ISSN: 2319-6491
Volume 12, Issue 5 [May. 2023] PP: 365-384

A Survey of Some Human-Based Meta-Heuristic Algorithms

M.M.Saber[1], M.A.El Sayed[1, 2], A.N. Meshref [1], M.A.Elsisy[1]
[1] Banha Faculty of Engineering, Banha University, Banha, EGYPT
[2] Faculty of Engineering, BADR University in Cairo BUC, Cairo, Egypt

ABSTRACT: *Human-based algorithms are a type of meta-heuristic algorithm inspired by human behavior, problem-solving strategies, and social interaction. In this paper, human-based meta-heuristic algorithms are presented, as their advantages, limitations, and applications. This paper has an assessment of the rapid evolution of human-based meta-heuristic thoughts, their covering towards a unified tissue, and the richness of possible applications in optimization problems. The paper briefly surveys some different human-based meta-heuristic algorithms aiming to solve optimization problems. Human-based algorithms have at least eight algorithms: Driving Training-Based Optimization (DTBO), Chef-Based Optimization Algorithm (CBOA), Sewing Training-Based Optimization (STBO), Human Behavior-Based Optimization (HBBO), Group Counseling Optimizer (GCO), Seeker Optimization Algorithm (SOA), Tabu Search Algorithms (TSA), Mine Blast Algorithm (MBA).*

Keywords*: Meta-heuristics, Algorithms, Optimization, Human-based algorithms, Mathematical model, Flow chart, Pseudo-Code, Advantages, Limitations, and Applications.*

Date of Submission: 10-05-2023 Date of acceptance: 21-05-2023

---

1. INTRODUCTION

There are four primary categories of meta-heuristic optimization algorithms: Swarm-based, Human-based, Physics-based, and Phylogeny- based. Human-based algorithms are a class of optimization algorithms inspired by human behavior, problem-solving strategies, and social interaction. These algorithms simulate the problem-solving process that humans use to find solutions to complex problems, often relying on the expertise and knowledge of multiple individuals to reach a satisfactory solution.

Human-based meta-heuristic algorithms are needed for several causes:

1. They are an easy tool for the destination of decision-making.
2. Support the optimization problem by creating a structure that the mental process of determining the exact solution cannot be verified.
3. Leaving us to determine which alternatives must be proven, and how the problem's data must be collected) are neglected in the definitions provided with mathematical formulas.
4. Can be applied as a part of the process of determining the exact solution, and for learning purposes.

However, human-based algorithms also face challenges, such as ensuring the quality and consistency of human input and addressing bias and fairness issues. Despite these challenges, human-based algorithms are expected to play an increasingly important role in solving complex problems in various domains, including healthcare, finance, cyber security, and the best in engineering applications by:

1. Accessible to combine with your existing implementation.
2. Not taking gradient information.
3. Can be applied to a wide range of problems including different topics.

The most important algorithms of Human-based algorithms:

1- Driving Training-Based Optimization (DTBO) [1].
2- Chef-Based Optimization Algorithm (CBOA) [2].
3- Sewing Training-Based Optimization (STBO) [3].
4- Human Behavior-Based Optimization (HBBO) [4].
5- Group Counseling Optimizer (GCO) [10].
6- Seeker Optimization Algorithm (SOA) [14].
7- Tabu Search Algorithms (TSA) [21].
8- Mine Blast Algorithm (MBA) [23].

The rest of the paper is organized as follows: In section 1, a brief introduction to some of the famous algorithms of Human-based algorithms. Section 2 presents a new human-based metaheuristic algorithm for solving optimization problems (DTBO) [1]. Section 3, shows a human‐based metaheuristic optimization method based on mimicking cooking training (CBOA) [2]. Section 4, presents the human-inspired met heuristic algorithm for solving optimization problems based on mimicking sewing training (STBO) [3]. Section 5, introduces human behavior-based optimization (HBBO) [4]. Section 6, provides data about the seeker optimization algorithm (SOA) [10]. Section 7, discusses the tabu search algorithm (TSA) [17]. Section 8, Conclusion.

II. DRIVING TRAINING-BASED OPTIMIZATION (DTBO) [1]

"Driving Training ‐Based Optimization" (DTBO) is a relatively new metaheuristic optimization algorithm that takes inspiration from the process of driving training to solve optimization problems.

The algorithm is based on the observation that driving training involves a combination of both exploration and exploitation. During driving training, a driving instructor typically guides the learner driver to explore various routes and scenarios to build up their experience and skills. Once the learner driver has gained some experience, the focus then shifts towards exploiting that experience to hone their driving skills and improve their performance.

In DTBO, this idea is translated into an optimization algorithm by using a combination of random exploration and guided search. The algorithm starts with an initial population of candidate solutions, which are randomly generated. The algorithm then iteratively updates this population by applying a set of driving training-inspired operations, such as "crossing over" or "mutation", to explore new candidate solutions

In addition, the algorithm also incorporates a "driving instructor" in the form of a human expert. The expert provides guidance to the algorithm by evaluating the candidate solutions and providing feedback on their quality. This feedback is then used to further guide the search toward better solutions.

The main advantage of DTBO is its ability to combine both exploration and exploitation effectively, which can lead to faster convergence toward good solutions. However, the use of a human expert also introduces a potential limitation, as it requires the availability and expertise of such an expert.

Overall, DTBO is a promising optimization algorithm that has shown good performance on a range of benchmark problems. However, further research is needed to explore its potential and limitations in different problem domains

Mathematical Model of DTBO

DTBO is a population-based met heuristic with learner drivers and instructors as its members. Members of the DTBO are potential solutions to the given problem, which is modeled by a population matrix in Equation (1). Equation (2) is used to initialize these members' positions at random at the beginning of implementation.

$$X = \begin{bmatrix} X_1 \\ \vdots \\ X_i \\ \vdots \\ X_N \end{bmatrix}_{N\times m} = \begin{bmatrix} x_{11} & \dots & x_{1j} & \dots & x_{1m} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{i1} & \dots & x_{ij} & \dots & x_{im} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{N1} & \dots & x_{Nj} & \dots & x_{Nm} \end{bmatrix}_{N\times m}, \quad (1)$$

$$x_{ij} = lb_j + r\,.\left(ub_j - lb_j\right), \quad i = 1,2,\dots,N, \quad j = 1,2,\dots,m \quad (2)$$

where X is the DTBO population, X_i is the ith candidate solution, x_ij is the value of the jth variable as determined by the ith candidate solution, N is the DTBO population size, m is the number of problem variables, r is a random number from [0, 1], and lb_j and ub_j are the lower and upper bounds of the jth problem variable, respectively.

The problem variables are given values by each potential solution, and these values are then evaluated for the objective function by including them in the objective function. As a result, each potential quick fix's value for the objective function is calculated. The objective function's values are modeled by the vector in equation (3).

$$F = \begin{bmatrix} F_1 \\ \vdots \\ F_i \\ \vdots \\ F_N \end{bmatrix}_{N\times 1} = \begin{bmatrix} F(X_1) \\ \vdots \\ F(X_i) \\ \vdots \\ F(X_N) \end{bmatrix}_{N\times 1} \quad (3)$$

F_i stands for the value of the objective function provided by the i th candidate solution, and F represents the vector of the objective functions.

The primary criterion for judging the quality of potential solutions is the values obtained for the objective function. The population member with the highest value for the objective function is referred to as the best member (X_best) based on a comparison of the values of the objective function. Since candidate solutions are updated and improved with each iteration, the best member must likewise be changed.

The method used to update candidate solutions is the major distinction between met heuristic methods. Candidate solutions in DTBO are updated during the three stages listed below: the student driver is first instructed by a driving teacher, then they model the instructor's techniques, and finally, they practice driving.

Phase 1: Training by the driving instructor (exploration).

The trainee driver chooses their driving teacher during the first step of the DTBO update, and that instructor then trains the novice driver in driving. A small group of the best members of the DTBO is classified as driving instructors, while the remaining members are learner drivers. Members of the population will go to various locations in the search space after selecting the driving teacher and mastering their techniques. As a result, the DTBO will have more exploration capacity while looking everywhere and finding the best location. As a result, this stage of the DTBO update illustrates this algorithm's exploratory capabilities. Based on the comparison of the data from each iteration According to Eq. (4), the N members of the DTBO are chosen as driving instructors based on the objective function.

$$DI = \begin{bmatrix} DI_1 \\ \vdots \\ DI_i \\ \vdots \\ DI_{N_{DI}} \end{bmatrix}_{N_{DI}\times m} = \begin{bmatrix} DI_{11} & \dots & DI_{1j} & \dots & DI_{1m} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ DI_{i1} & \dots & DI_{ij} & \dots & DI_{im} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ DI_{N_{DI}1} & \dots & DI_{N_{DI}j} & \dots & DI_{N_{DI}m} \end{bmatrix}_{N_{DI}\times m}, \quad (4)$$

Where DI is the matrix of driving instructors, DI_i is the ith driving instructor, DI_{ij} is the jth dimension, and $N_{DI} = \lfloor 0.1 \cdot N \cdot (1 - t/T) \rfloor$ is the number of driving instructors, where t is the current iteration and T is the maximum number of iterations.

The new position for each member in this DTBO phase is first determined using Eq. (5), according to the mathematical modeling of this phase. If this new position raises the value of the goal function, it replaces the prior one in accordance with Eq. (6).

$$x_{i,j}^{P_1} = \begin{cases} x_{i,j} + r\,.\,\left(DI_{k_{i,j}} - I\,.\,x_{i,j}\right), & F_{DI_{k_i}} < F_i\,; \\ x_{i,j} + r\,.\,\left(DI_{k_{i,j}} - I\,.\,x_{i,j}\right), & otherwise\,, \end{cases} \quad (5)$$

$$X_i = \begin{cases} X_i^{P_1}, & F_i^{P_1} < F_i\,; \\ X_i, & otherwise\,, \end{cases} \quad (6)$$

Where $X_i^{P_1}$ is the new calculated status for the ith candidate solution based on the first phase of DTBO, $x_{i,j}^{P_1}$ is its jth dimension, $F_i^{P_1}$ is its objective function value, I is a number randomly selected from the set {1, 2} , r is a random number in the interval [0, 1], DI_{k_i} , where k_i is randomly selected from the set {1, 2, . . . , , represents a randomly selected driving instructor to train the ith member, $DI_{k_i,j}$ is its jth dimension, and $F_{DI_{k_i}}$is its objective function value.

Phase 2: Patterning of the instructor skills of the student driver (exploration).

The trainee driver imitates the instructor in the second stage of the DTBO update, trying to mimic all of the instructor's gestures and driving techniques. The members of the DTBO are moved to various locations inside the search space, enhancing the DTBO's exploration power. A new position is created based on the linear combination of each member with the teacher in accordance with Eq. (7) to mathematically mimic this idea. According to Eq. (8), the new position will take the place of the old one if it increases the value of the objective function.

$$x_{i,j}^{P_2} = P\ .\ x_{i,j} + (1 - P\ )\ .\ DI_{k_{i,j}}\ , \tag{7}$$

$$X_i = \begin{cases} X_i^{P_2}\ ,\ F_i^{P_2} < F_i\ ; \\ X_i\ ,\ otherwise\ , \end{cases} \tag{8}$$

Where $X_i^{P_2}$ is the new calculated status for the ith candidate solution based on the second phase of DTBO, $x_{i,j}^{P_2}$ is its jth dimension, is $F_i^{P_2}$ its objective function value, and P is the patterning index given by

$$P = 0.01 + 0.9\left(1 - \frac{t}{T}\right). \tag{9}$$

Phase 3: Personal practice (exploitation).

The third stage of the DTBO update is centered on each learner driver's individual practice to strengthen and improve their driving abilities. Each novice driver aims to get a little nearer to his best abilities in this stage. Each participant is given the opportunity to find a better position during this phase by doing a local search around their current location. The ability of DTBO to leverage local search is demonstrated in this step. This DTBO phase is mathematically modeled so that, in accordance with Eq. (10), a random position is initially created close to each population member. Then, if this position increases the value of the goal function, it replaces the preceding position, using Eq. (11).

$$x_{i,j}^{P_3} = x_{i,j} + (1 - 2r\ ).R\ .\left(1 - \frac{t}{T}\right)\ .\ x_{i,j}\ , \tag{10}$$

$$X_i = \begin{cases} X_i^{P_3}\ ,\ F_i^{P_3} < F_i\ ; \\ X_i\ ,\ otherwise\ , \end{cases} \tag{11}$$

Where $X_i^{P_3}$ is the new calculated status for the ith candidate solution based on the third phase of DTBO, $x_{i,j}^{P_3}$ is its jth dimension, $F_i^{P_3}$ is its objective function value, r is a random real number of the interval [0, 1], R is the constant set to the value 0.05, t is the counter of iterations and T is the maximum number of iterations.

Algorithm 1. Pseudo-Code of DTBO.

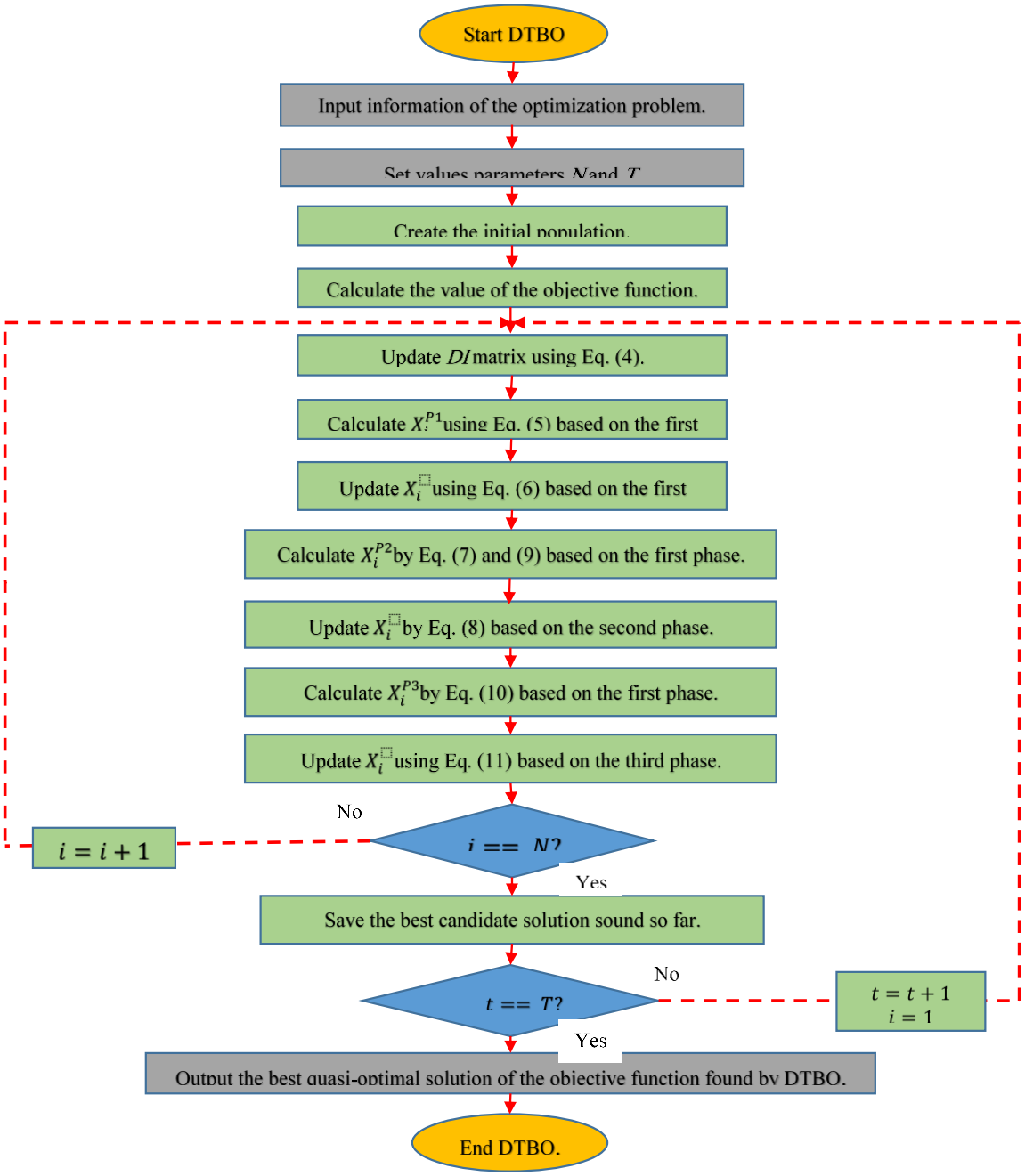
Start DTBO.

1. Input: The optimization problem information.
2. Adjust N and T.
3. Initialize the DTBO population position and evaluate the objective function.
4. For t = 1 to T
5. For i = 1 to N
6. Phase 1: Training by the driving instructor (exploration).
7. Determine the driving instructor matrix based on a comparison of objective function values.
8. Select a driving instructor at random from the matrix DI.
9. Calculate the new position for the ith DTBO member using Equation (5).
10. Update the position of the ith DTBO member using Equation (6).
11. Phase 2: Learner driver patterning from instructor skills (exploration).
12. Calculate the patterning index P using Equation (9).
13. Calculate a new position of the ith DTBO member using Equation (7).
14. Update the position of the ith DTBO member using Equation (8).
15. Phase 3: Personal practice (exploitation).
16. Calculate the new position for the ith DTBO member using Equation (10).
17. Update the position of the ith DTBO member using Equation (11).
18. End.
19. Update the best-found candidate solution.
20. End.

21. Output: The best candidate solution obtained by DTBO.

End DTBO.

The DTBO flow chart.



The advantages of DTBO are:

1. The functions C1 to C30's optimization results demonstrated that DTBO can satisfactorily tackle challenging optimization issues.
2. A comparison of the performance of the proposed DTBO with that of competing algorithms revealed that it is significantly more competitive and more successful at optimizing and reaching optimal solutions than the algorithms evaluated.

3. The application of DTBO to two engineering design problems showed the effectiveness of the suggested strategy in resolving practical applications.

The limitations of DTBO are:

1- DTBO has produced workable results in problem-solving, however, there are some restrictions on this approach in other applications.
2- The authors make no assertions that DTBO is the greatest optimizer for dealing with optimization issues because this claim is categorically and categorically refuted by the NFL theorem. As a result, DTBO might not be successful in tackling some optimization problems.
3- The fundamental drawback of any metaheuristic algorithm, including DTBO, is that new optimization techniques might be discovered to handle optimization applications more effectively in the future.

Applications of DTBO:

1) Pressure vessel design: Pressure vessel design is a real-world optimization theme aimed at minimizing design costs [1].
2) Welded beam design: Welded beam design is an engineering optimization problem aimed at reducing fabrication costs [1].

III. COOKING-BASED OPTIMIZATION ALGORITHM (CBOA) [2]

The Cooking-Based Optimization Algorithm (CBOA) is a new optimization method that draws inspiration from the process of cooking and culinary training. The goal of this approach is to mimic the way that chefs combine ingredients and use different cooking techniques to create a delicious meal and apply it to solving optimization problems.

CBOA is a population-based meta-heuristic optimization algorithm that uses a set of candidate solutions, referred to as "dishes," that are evolved over time through the application of various cooking techniques, such as boiling, frying, or baking. The algorithm is based on the principle that the best solutions are those that combine the right ingredients in the right proportions and apply the right cooking techniques to achieve the desired outcome.

The CBOA algorithm starts with an initial set of dishes, which are generated randomly. Each dish is evaluated according to a fitness function, which measures its quality or effectiveness in solving the optimization problem at hand. Based on this evaluation, the dishes are ranked and selected for further processing.

The CBOA algorithm then applies a series of cooking techniques to the selected dishes, such as mutation, crossover, and selection, to create new dishes that are potentially better than the previous ones. This process is repeated until a satisfactory solution is found or a predetermined stopping criterion is met.

One advantage of CBOA is that it can handle complex optimization problems with a large number of variables and constraints. Another advantage is that it can easily incorporate domain-specific knowledge and constraints into the optimization process.

However, like all optimization algorithms, CBOA also has limitations and challenges. One challenge is the selection of appropriate cooking techniques and their parameters, which can significantly affect the algorithm's performance. Another challenge is the risk of premature convergence or getting stuck in local optima, which can reduce the algorithm's effectiveness.

In summary, the Cooking-Based Optimization Algorithm (CBOA) is a promising new optimization method that uses culinary techniques to solve complex optimization problems. While it has its limitations and challenges, it has the potential to be a valuable tool in many domains, such as engineering, finance, and logistics.

Mathematical modeling of CBOA:-

The group of the first N_C members is chosen as the group of chef instructors and the remaining group of N - N_C members is chosen as the group of cooking students if the rows of the CBOA population matrix are sorted in ascending order according to the value of the objective function (thus, the member in the first row is the best member). The sorted objective function vector and the CBOA sorted population matrix are described in Eqs. (4) and (5), respectively.

$$\text{XS} = \begin{bmatrix} XS_1 \\ \vdots \\ XS_{N_C} \\ XS_{N_C+1} \\ \vdots \\ XS_N \end{bmatrix}_{N\times m} = \begin{bmatrix} x_{1,1} & \dots & x_{1,j} & \dots & x_{1,m} \\ \vdots & \ddots & \vdots & \iddots & \vdots \\ xs_{N_C,1} & & xs_{N_C,j} & & xs_{N_C,m} \\ xs_{N_C+1,1} & \iddots & xs_{N_C+1,j} & \cdots & xs_{N_C+1,m} \\ \vdots & \dots & \vdots & \ddots & \vdots \\ xs_{N,1} & & xs_{N,j} & \dots & xs_{N,m} \end{bmatrix}_{N\times m}, \quad (4)$$

$$\text{FS} = \begin{bmatrix} FS_1 \\ \vdots \\ FS_{N_C} \\ FS_{N_C+1} \\ \vdots \\ F_N \end{bmatrix}_{N\times 1}, \quad (5)$$

Where N_C the number of chef instructors, XS is the sorted population matrix of CBOA, and FS is a vector of ascending objective function values. In the matrix XS, members from XS_1 to XS_{N_C} represent the group of chef instructors, and members from XS_{N_C+1} to XS_N represent the group of cooking students. The vector FS_i includes successively the values of the objective functions corresponding to XS_1 to XS_N.

Phase 1: the updating process for a group of chef instructors (update of XS_1 to XS_{N_C}).

It is considered that numerous chef teachers are in charge of instructing students in cooking techniques at a culinary school. Chef instructors practice two methods to raise their level of culinary proficiency. In the first tactic, they try to imitate the top chef instructor and understand their methods. This tactic exemplifies the capabilities of CBOA exploration and global search.

The benefit of updating the chef instructors according to this technique is that the top chefs (top population members) advance their abilities based on the best chef (best population member) before they begin instructing students. As a result, in CBOA design, there is no direct reliance on upgrading the students' position solely on the basis of the population's best members.

This strategy also avoids the algorithm from being stuck in local optima and improves the accuracy and efficiency with which different regions of the search space are examined. Based on this strategy, a new position for each chef instructor is first calculated for i = 1, 2,..., N_C and j = 1, 2, . . . , m using the following equation

$$xs_{i,j}^{c/s_1} = xs_{i,j} + r\,.\left(BC_j - I.xs_{i,j}\right), \quad (6)$$

Where xs_i^{c/s_1} is the new calculated status for the i th sorted member of CBOA (that is xs_i) based on the first strategy (c/s_1) of updating the chef instructor, $xs_{i,j}^{c/s_1}$ is its j th coordinate, BC is the best chef instructor (denoted as xs_1 in the matrix xs), BC_j is the j th coordinate of the best chef instructor, r is a random number from the interval [0, 1], and I is a number that is selected randomly during execution from the set {1, 2}. This new position is acceptable to the CBOA if it improves the value of the objective function. This condition is modeled using Eq. (7).

$$XS_i = \begin{cases} XS_i^{c/s_1}, & FS_i^{c/s_1} < F_i \ ; \\ XS_i & , \ otherwise \ , \end{cases} \quad (7)$$

Where FS_i^{c/s_1} is the value of the objective function of the member XS_i^{c/s_1}.

In the second approach, each chef instructor works on honing his cooking techniques through personalized activities and exercises. This approach exemplifies the CBOA's capacity for exploitation and local search. If a chef teacher views each problem variable as a cooking talent, they will work to enhance each one in order to raise the objective function value.

The benefit of updating depending on individual activities and workouts is that each member searches for better solutions close to its location, independent of where other population members are located. With simple adjustments to the positioning of population members in the search space, it is possible to find better solutions using local search and exploitation. Using Eqs. (8) to (10), a random position is produced around each culinary instructor in the search area for j = 1, 2...., m. This condition is modeled using Eq. (11) and states that this random position is appropriate for updating if it increases the value of the objective function.

$$lb_j^{local} = \frac{lb_j}{t}, \quad (8)$$

$$ub_j^{local} = \frac{ub_j}{t}, \quad (9)$$

Where lb_j^{local} and ub_j^{local} are the lower and upper local bound of the j th problem variable, respectively, and the variable t represents the iteration counter.

$$xs_{i,j}^{c/s_2} = xs_{i,j} + lb_j^{local} + r\,.\left(ub_j^{local} - lb_j^{local}\right), i = 1,2,\dots,N_C, j = 1,2,\dots,m \quad (10)$$

$$XS_i = \begin{cases} XS_i^{C/S_1}, & FS_i^{C/S_2} < F_i \ ; \\ XS_i & , otherwise \ , \end{cases} \quad (11)$$

Where XS_i^{C/S_2} is the new calculated status for the i th CBOA sorted member (i.e., XS_i) based on the second strategy (C/S2) of chef instructors updating, $xs_{i,j}^{c/s_2}$ is its j th coordinate, and FS_i^{c/s_2} is its value of the objective function.

Phase 2: the updating process for the group of cooking students (update of XS_{N_C+1} to XS_N).

Culinary schools are attended by students who want to learn how to cook and work as chefs. The CBOA's design makes three assumptions about how cooking students would acquire their abilities. In accordance with the first strategy, each culinary student selects at random a class taught by a chef, who then instructs him on cooking techniques. The benefit of updating cooking students using this strategy is that various chef instructors are available to guide them. As a result, cooking students learn various skills (i.e., population members move to other areas of the search space) based on the direction of the selected chef instructor. On the other hand, a successful global search in the space of problem-solving would be impossible if all cookery students only learn from the best chef-instructor (all members of the population gravitated towards the best member).

This strategy is simulated in the CBOA in such a way that first for each cooking student, a new position is calculated based on the training and guidance of the chef instructor, for $i = N_C + 1, N_C + 2, \dots, N, j = 1,2, \dots, m$, using Eq. (12).

$$xs_{i,j}^{S/S_1} = xs_{i,j} + r \cdot \left(CI_{k_{i,j}} - I \cdot xs_{i,j} \right), \quad (12)$$

Where XS_i^{S/S_1} is the new calculated status for the i th sorted member of CBOA (i.e., XS_i) based on the first strategy (S/S1) of the updating of cooking students, $xs_{i,j}^{S/S_1}$ is its j th coordinate and $CI_{k_{i,j}}$ is the selected chef instructor by the i th cooking student, where K_i is randomly selected from the set {1,2, . . . , N_C} (where $CI_{k_{i,j}}$ denotes the value $xs_{k_{i,j}}$).

This new position replaces the previous position for each CBOA member if it improves the value of the objective function. This concept is modeled for $i = N_C + 1, N_C + 2, \dots, N$ by Eq. (13).

$$XS_i = \begin{cases} XS_i^{S/S_1}, & FS_i^{S/S_1} < F_i \ ; \\ XS_i & , otherwise \ , \end{cases} \quad (13)$$

Where FS_i^{S/S_1} is the value of the objective function of XS_i^{S/S_1} .

The CBOA's ability to search and explore the world is improved by this method. The benefit of this approach is that just one variable—one skill, in this case, one recipe—changes, as opposed to all possible solution variables—all cooking student skills—being updated. To arrive at better solutions, it might not be required to update all member position coordinates.

In the design of CBOA, this "skill'' represents a certain component of a vector of cooking skills of a randomly selected chef instructor CI_k (k ∈ {1, 2,. . .,N_C}). Hence, the second strategy is mathematically simulated in such a way that for each cooking student XS_i (members of CBOA with $i = N_C + 1, N_C + 2, \dots, N$), first one chief instructor, which is represented by the vector CI_{k_i} = ($CI_{k_{i,1}}, \dots, CI_{k_{i,m}}$) is randomly selected (a member of CBOA with the index k_i which is randomly selected from the set {1, ..., N_C}), then it is randomly selected his ↕ th coordinate (thus a number ↕ from the set {1, ...m}, which represents a "skill'' of this selected chief instructor) and by this value $CI_{k_{i,↕}}$ we replace the ↕ th coordinate of the vector of I th cooking student XS_i (thus, $xs_{i,↕}$).

According to this concept, a new position is calculated for each CBOA cooking student member using Eq. (14).

$$xs_{i,j}^{S/S_2} = \begin{cases} CI_{k_{i,j}}, & j = l \ ; \\ XS_{ij} \ , & otherwise \ , \end{cases} \quad (14)$$

Where ↕ is a randomly selected number from the set {1, 2, . . . , m}, $i = N_C + 1, N_C + 2, \dots, N, j = 1,2, \dots, m$.

If it increases the target value of the objective function, it is then replaced with the prior location based on Eq. (15).

$$XS_i = \begin{cases} XS_i^{S/S_2}, & FS_i^{S/S_2} < F_i \ ; \\ XS_i & , otherwise \ , \end{cases} \quad (15)$$

Where XS_i^{S/S_2} is the new calculated status for the i th sorted member of CBOA (i.e.,XS_i) based on the second strategy (S/S2) of updating cooking students, $xs_{i,j}^{S/S_2}$ is its j th coordinate, FS_i^{S/S_2} is its objective function value.

The third tactic involves each student working to hone their cooking abilities through activities and exercises. This approach exemplifies the CBOA's capacity for exploitation and local search. A cooking student will want to enhance all of the problem variables if they are thought of as cooking skills in order to improve the objective function value.

This idea states that a random position is generated around each cooking student in the search space by Eqs. (8) And (9), and a new position is derived by Eq. (16).

$$XS_i^{S/S_3} = \begin{cases} xs_{i,j} + lb_j^{local} + r \,.\, \left(ub_j^{local} - lb_j^{local}\right), j = q; \\ xs_{i,j}, \qquad\qquad\qquad\qquad\quad j \neq q \,, \end{cases} \tag{16}$$

where XS_i^{S/S_3} is the new calculated status for the i th sorted member of CBOA (that is xs_i) based on the third strategy (S/S3) of updating cooking students, $xs_{i,j}^{S/S_3}$ is its j th coordinate, and q is randomly selected number from the set {1,2, . . . ,m}, $i = N_C + 1, N_C + 2, \ldots, N, j = 1,2, \ldots, m$. If this new random position improves the value of the objective function, it is acceptable for updating XS_i , which is modeled by Eq. (17).

$$XS_i = \begin{cases} XS_i^{S/S_3}, \; FS_i^{S/S_3} < F_i \; ; \\ XS_i \qquad , \; otherwise \, , \end{cases} \tag{17}$$

Where FS_i^{S/S_3} is the value of the objective function of XS_i^{S/S_3} .

Algorithm 1. Pseudo-Code of CBOA.

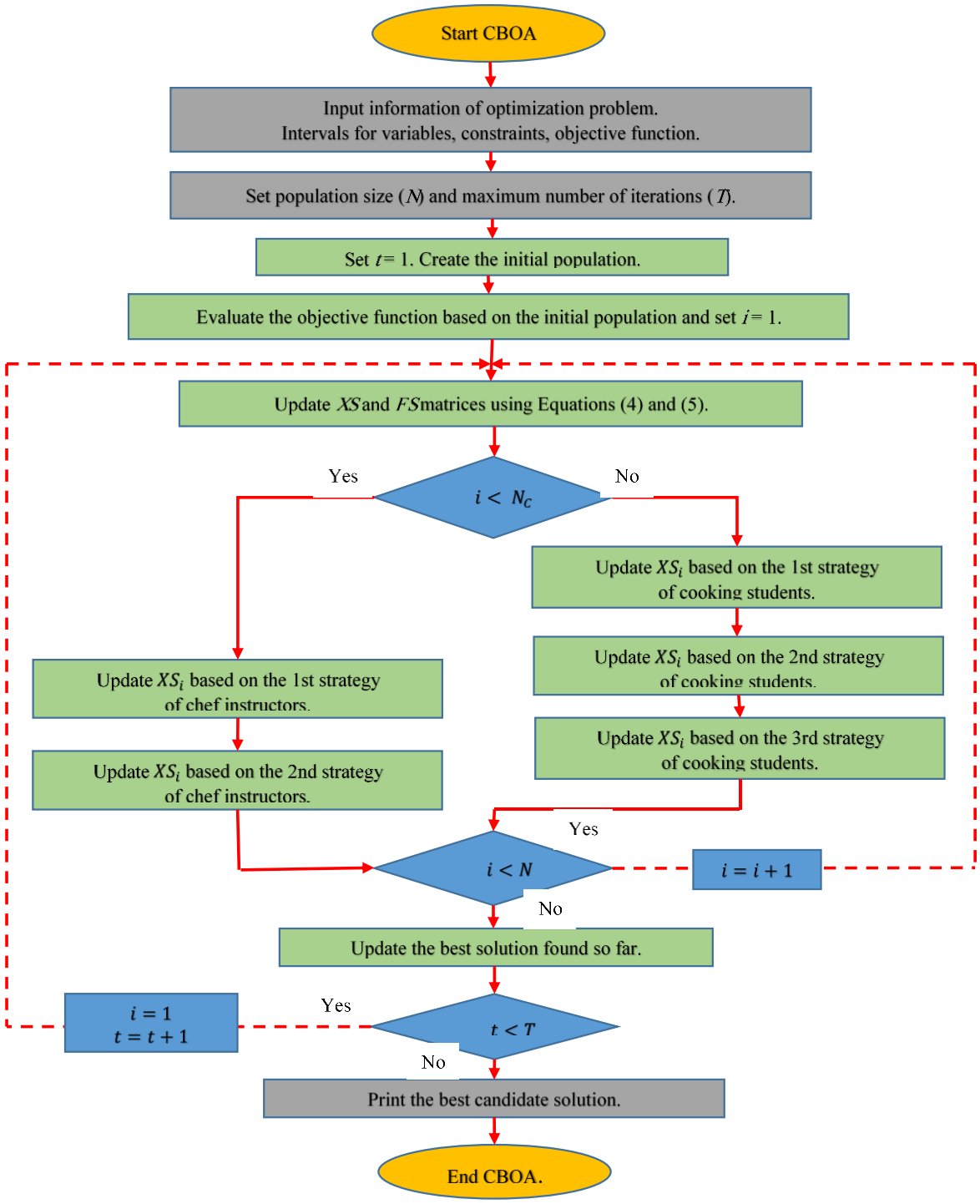
Start CBOA.

1. Input problem information: variables, an objective function, and constraints.
2. Set population size of the CBOA (N) and iterations (T).
3. Randomly generate an initial population matrix X.
4. Evaluate the given objective function to obtain the vector F.
5. For t = 1 to T
6. Sort the matrix X based on the values of the objective function according to Equation (4) and (5).
7. Update the set of chef instructors CI = $\{CI_1, CI_2, \ldots, CI_{N_C}\}$ and the best CBOA member BC (Clearly, we set $= CI_1$)
8. Start Phase 1: Training by the driving instructor (exploration).
9. For i = 1 to N_C
10. Calculate $XS_i^{C/S1}$ using Equation (6) (updating of chef instructors based on the first strategy of Phase 1).
11. Update XS_i using Equation (7).
12. Update the lower and the upper local bound of problem variables using Equation (8) and (9).
13. Calculate $XS_i^{C/S2}$ using Equation (10) (updating of skills of chef instructors based on the second strategy of Phase 1).
14. Update XS_i using Equation (11).
15. End.
16. End Phase 1: The updating process of chef instructors (XS_1 , XS_{N_C})
17. Start Phase 2: The updating process of cocking students (XS_{N_C+1}, XS_N)
18. For i = $N_C + 1$ to N.
19. Choose a chef instructor at random to train the i th cooking student.
20. Calculate $XS_i^{C/S1}$ using Equation (12) (updating of cooking students based on the first strategy of Phase 2).
21. Update XS_i using Equation (13).
22. Calculate $XS_i^{C/S2}$ using Equation (14) (updating of chef instructors based on the second strategy of Phase 2).
23. Update XS_i using Equation (15).
24. Calculate $XS_i^{C/S3}$ using Equation (8), (9), and (16) (updating of chef instructors based on the third strategy of Phase 2).

25. Update XS_i using Equation (17).
26. End.
27. End Phase 2: The updating process of cooking students (XS_{N_C+1}, XS_N).
28. Update the best candidate solution found so far.
29. End.
30. Output: The best quasi-optimal solution obtained by CBOA.

End CBOA.

The CBOA flow chart.



The advantages of CBOA are:

1. The implementation results of CBOA and competing algorithms on functions F8 to F13 demonstrate CBOA's great exploration potential for global search in a variety of problem-solving domains.
2- The outcomes of functions F9 and F11's optimization show this CBOA potential to be exceptionally strong.
3. Its capacity to balance exploitation and exploration allows it to first use global search to identify the primary optimal region without becoming entangled in locally optimal solutions, and then use local search to converge to the global optimum. When it comes to optimizing each objective function, CBOA outperforms some rival algorithms in terms of execution time.
4- The CBOA introduces future research challenges and directions.
5-Other suggestions include using CBOA for optimization applications across several fields and in real-world problems.
6- Twelve well-known meta-heuristic algorithms and fifty-two typical benchmark functions are used to test and compare CBOA's capacity to solve optimization difficulties.

The limitations of CBOA are:

1- Its successful execution in all optimization applications is not conditional on any particular factors. As a result, the suggested CBOA has a drawback and a limitation that could prevent its use in certain optimization issues.
2- It is always feasible that academics will create newer met heuristic algorithms that offer superior answers to actual optimization issues to those provided by existing algorithms.
3- Despite being faster, several rival algorithms failed to reach the required outcomes. Therefore, when optimizing the objective functions, CBOA has a respectable execution time.

Applications of CBOA:

1- Pressure vessel design (PVD) [2].
2- Speed reducer design (SRD) [2].
3- Welded beam design (WBD) [2].
4- Structural tension/ compression springs (TCSD) [2].

IV. SEWING TRAINING-BASED OPTIMIZATION (STBO) [3]

The Sewing Training-based Optimization (STBO) is a relatively new optimization algorithm that mimics the way humans learn sewing. The algorithm was proposed by researchers in 2020, and it has shown promising results in solving various optimization problems.

The STBO algorithm is inspired by the way humans learn sewing. In sewing, a person learns by trial and error, and they gradually improve their stitching skills. Similarly, in the STBO algorithm, a population of solutions is generated, and each solution is improved by learning from the previous ones.

The STBO algorithm has the following steps:

1. Initialization: A population of solutions is randomly generated.
2. Evaluation: Each solution is evaluated based on its fitness function.
3. Selection: The best solutions are selected based on their fitness value.
4. Learning: The selected solutions are used to learn new solutions. This is done by randomly selecting two solutions and merging them to create a new solution. The new solution is then evaluated and added to the population.
5. Termination: The algorithm stops when a termination criterion is met, such as reaching a maximum number of iterations or achieving a satisfactory solution.

The STBO algorithm has shown promising results in solving various optimization problems, including function optimization, feature selection, and data clustering. However, like any other optimization algorithm, its performance depends on the problem being solved and the parameters used.

In conclusion, the Sewing Training-based Optimization (STBO) algorithm is a new optimization algorithm inspired by the way humans learn sewing. It has shown promising results in solving various optimization problems and is worth exploring further.

Mathematical Model of STBO

Mathematically, the STBO population can be represented as a matrix, and each STBO member as an individual vector. In Equation (1), a matrix representation describes the STBO population.

$$X = \begin{bmatrix} X_1 \\ \vdots \\ X_i \\ \vdots \\ X_N \end{bmatrix}_{N\times m} = \begin{bmatrix} x_{1,1} & \dots & x_{1,j} & \dots & x_{1,m} \\ \vdots & \ddots & \vdots & \iddots & \vdots \\ x_{i,1} & \dots & x_{i,j} & \dots & x_{i,m} \\ \vdots & \iddots & \vdots & \ddots & \vdots \\ x_{N,1} & \dots & x_{N,j} & \dots & x_{N,m} \end{bmatrix}_{N\times m}, \quad (1)$$

Where X is the STBO population matrix, X_i is the i th STBO's member, N is the number of STBO population members, and m is the number of problem variables. At the beginning of the STBO implementation, all population members are randomly initialized using Equation (2).

$$x_{ij} = lb_j + r\,.\,(ub_j - lb_j), \quad i = 1,2,\dots,N,\ j = 1,2,\dots,m, \quad (2)$$

Where x_{ij} is the value of the j th variable determined by the i th STBO's member X_i, r is a random number in the interval [0, 1], lb_j and ub_j are the lower and upper bound of the j th problem variable, respectively.

The values produced for the goal function can be modeled using a vector by Equation (3) based on where the candidate solutions are placed in the problem variables.

$$F = \begin{bmatrix} F_1 \\ \vdots \\ F_i \\ \vdots \\ F_N \end{bmatrix}_{N\times 1} = \begin{bmatrix} F(X_1) \\ \vdots \\ F(X_i) \\ \vdots \\ F(X_N) \end{bmatrix}_{N\times 1}, \quad (3)$$

Where F is the objective function vector and F_i is the objective function value for the i th candidate solution. The best candidate solution or the best person in the population X_(best) is determined to be the solution with the best value for the objective function.

Three stages are involved in updating candidate solutions in STBO: teaching, imitation of the instructor's skills, and practice.

Phase 1: Training (exploration)

The set of all candidate members as the group of possible training instructors for each STBO member X_k, $i = 1,2,\dots,N$, is defined using the following identity

$$CSI_i = \{X_k | F_k < F_i,\ k \in \{1,2,\dots,N\}\} \cup \{X_{best}\}, \quad (4)$$

Where CSI_i is the set of all possible candidate training instructors for the i th STBO member. In this case $X_i = X_{best}$ the only possible candidate training instructor is X_{best} itself, i. e., $CSI_i = X_{best}$. Then, for each $i \in \{1,2,\dots,N\}$, a member from the set CSI_i is randomly selected as the training instructor of the i th member of STBO, and it is denoted as SI_i . This selected instructor SI_i teaches the i th STBO member to sewing skills.

To update population members based on this phase of the STBO, a new location is first created for each population member using Equation (5).

$$x_{i,j}^{P_1} = x_{i,j} + r_{i,j}\,.\,(SI_{i,j} - I_{i,j}\,.\,x_{i,j}), \quad (5)$$

Where $x_{i,j}^{P_1}$ is its d th dimension, $F_i^{P_1}$ is its objective function value, $I_{i,j}$ are numbers that are selected randomly from the set {1, 2}, and $r_{i,j}$ are random numbers from the interval [0, 1].

This update condition is modeled using Equation (6).

$$X_i = \begin{cases} X_i^{P_1}, & F_i^{P_1} < F_i; \\ X_i, & otherwise, \end{cases} \quad (6)$$

Where $X_i^{P_1}$ is the new position of the i th STBO member based on the first phase of STBO.

Phase 2: Imitation of the instructor's skills (exploration)

In this phase of STBO, it is assumed that each decision variable represents a sewing skill. Each STBO member imitates ms skills of the chosen instructor, $1 \le ms \le m$.

The algorithm's population is moved across the search space during this procedure, demonstrating the STBO's capacity for exploration. Equation (7) specifies the set of variables that each STBO member imitates (i.e., the set of training instructor's skills).

$$SD_i = \{d_1, d_2, \dots, d_{m_s}\}, \quad (7)$$

Where SDi is an ms – a combination of the set {1, 2, …, m}, which represents the set of the indexes of decision variables (i.e., skills) identified to imitate by the i th member from the instructor and $m_s = \left[1 + \frac{t}{2T}m\right]$ is the number of skills selected to mimic, t is the iteration counter, and T is the total number of iterations.

Using the following identification, the new position for each STBO member is determined based on the simulation of copying these instructor skills.

$$x_{i,j}^{P_2} = \begin{cases} SI_{i,j} & , \ j \in SD_i \ ; \\ x_{i,j} & , \ otherwise \ , \end{cases} \quad (8)$$

Where $X_i^{P_2}$ is the newly generated position for the i th STBO member based on the second phase of STBO, $x_{i,j}^{P_2}$ is the d th dimension of $X_i^{P_2}$. This new position replaces the previous position of the corresponding member if it improves the value of the objective function

$$X_i = \begin{cases} X_i^{P_2} & , \ F_i^{P_2} < F_i \ ; \\ X_i & , \ otherwise \ , \end{cases} \quad (9)$$

Where $F_i^{P_2}$ is the objective function value of $X_i^{P_2}$.

Phase 3: Practice (exploitation)

The ability of the suggested algorithm to be used in local search is represented by this STBO phase. Equation (10 is used to create a new location around each STBO member before modeling this STBO phase mathematically (with an adjustment to keep all newly computed population members in the specified search area).

$$x_{i,j}^{P_3} = \begin{cases} lb_j \ , \ x_{i,j}^* < lb_j \ ; \\ x_{i,j}^* , \ x_{i,j}^* \in [lb_j \ , ub_j \ ] \ ; \\ ub_j \ , \ x_{i,j}^* > ub_j \ , \end{cases} \quad (10)$$

Where $x_{i,j}^* = x_{i,j} + (lb_j + r_{i,j} \ . (ub_j - lb_j)/t$ and $r_{i,j}$ is a random number from the interval [0, 1]. Then, if the value of the objective function improves, it replaces the previous position of the STBO member according to Equation (11).

$$X_i = \begin{cases} X_i^{P_3} & , \ F_i^{P_3} < F_i \ ; \\ X_i & , \ otherwise \ , \end{cases} \quad (11)$$

Where $X_i^{P_3}$ is the newly generated position for the i th STBO member based on the second phase of STBO, $X_{i,j}^{P_3}$ is its d th dimension, and $F_i^{P_3}$ is its objective function value.

Repetition Process and Pseudo-Code of STBO

The update procedure is then continued until the algorithm's final iteration, which is based on Equations (4) to (11). The best candidate solution noted throughout the algorithm iteration is presented as the solution when the STBO has been fully applied to the given problem. Finally, Algorithm 1 presents the pseudo-code for the STBO implementation steps.

Algorithm 1. Pseudo-Code of STBO.

Start STBO.

1. Input the optimization problem information: variables, an objective function, and constraints.
2. Set population size of the STBO (N) and iterations (T).
3. Initialize the STBO population by (2) and create vector F of the values of the objective function by (3).
4. For t = 1 to T
5. For i = 1 to N
6. Phase 1: Training (exploration).
7. Determine the set of candidate training instructors for the i th member by (4). $CSI_i \leftarrow \{X_K | F_K < F_i, k \in \{1,2, \dots, N\}\} \cup \{X_{best}\}$.
8. Choose the training SI_i from CSI_i to teach sewing the i th STBO member.
9. Calculate the new position for the i th STBO member using (5). $x_{i,j}^{P1} \leftarrow x_{i,j} + r_{i,j} \ . (SI_{i,j} - I_{i,j} . x_{i,j})$
10. Update the position of the i th STBO member using (6). $X_i \leftarrow \begin{cases} X_i^{P1} , & F_i^{P1} < F_i \\ X_i \ , & else \end{cases}$
11. Phase 2: Imitation of the instructor skills (exploration)
12. Calculate SD_iusing Equation (7).
13. Calculate the new position of the i th STBO member using Equation (8). $x_{i,j}^{P2} \leftarrow \begin{cases} SI_{i,j} \ , & j \in SD; \\ x_{i,j} \ , & else. \end{cases}$
14. Update the position of the i th STBO member using (9). $X_i \leftarrow \begin{cases} X_i^{P2} , & F_i^{P2} < F_i \ ; \\ X_i \ , & else \end{cases}$
15. Phase 3: Practice (exploitation)

16. Calculate the new position for the i th STBO member using (10). $X_{i,j}^{P3} \leftarrow x_{i,j} + \frac{lb_j + r_{i,j}(ub_j - lb_j)}{t}$.
17. Update the position of the i th STBO member using (11). $X_i \leftarrow \begin{cases} X_i^{P3}, & F_i^{P3} < F_i; \\ X_i, & else \end{cases}$
18. End.
19. Update the best candidate solution.
20. End
21. Output: The best quasi-optimal solution obtained by CBOA.

End STBO

The advantages of STBO are:

1. The proposed STBO algorithm's suitability for applications in optimization and the presentation of solutions is assessed.
2 Mean, standard deviation (std), best, worst, median, and rank are the six statistical indicators used to report the results of the execution of metaheuristic algorithms. The performance of optimization algorithms in each of the objective functions is evaluated using the mean of rank as a ranking criterion.

3- The STBO's introduction initiates a number of research tasks for the next investigations.
One of the most focused STBO research suggestions is the creation of binary and multimodal versions.
5. Other ideas for additional research include using STBO in numerous scientific and practical optimization applications.

Applications of STBO:

1- Pressure vessel design (PVD) [3].
2- Speed reducer design (SRD) [3].
3- Welded beam design (WBD) [3].
4- Tension/compression spring design (TCSD) [3].

5. HUMAN BEHAVIOR-BASED OPTIMIZATION (HBBO) [4]

Human behavior-based optimization (HBBO) is a process of optimizing systems, processes, and technologies based on human behavior and preferences. The aim of HBBO is to make systems more user-friendly, intuitive, and efficient by taking into account how people interact with them.
HBBO is based on the principle that people tend to prefer certain behaviors, such as simplicity, efficiency, and ease of use, and will tend to repeat these behaviors when given the opportunity. By understanding these behaviors, HBBO seeks to create systems that are optimized for these preferences, making them easier and more enjoyable for people to use.
HBBO can be applied in a variety of contexts, from website design to product development to organizational processes. By using data analytics and user research, designers and developers can identify patterns in user behavior and preferences and use this information to improve the overall user experience.
Overall, HBBO is an important approach for creating systems that are both effective and enjoyable to use, and it can lead to increased user satisfaction, loyalty, and engagement.
This algorithm consists of the five steps as follows:

Step 1: Initialization
Step 2: Education
Step 3: Consultation
Step 4: Field-changing probability
Step 5: Finalization

2.1 Initialization

In this step, the initial people are created, assessed, and somewhat dispersed among the fields. An individual is defined in an optimization problem with $x_{N_{var}}$ variables as follows:

Individual = $\left[x_1, x_2, \ldots, x_{N_{var}}\right]$ (1)

The algorithm creates N_{pop} initial populations and distributes them at random among N_{field} initial fields. These people make up society. The initial number of people in each field is as follows:

$$N.Ind_i = round\left\{\frac{N_{pop}}{N_{field}}\right\} \quad (2)$$

Where $N.Ind_i$ is the number of initial individuals in the i-th field. After generating the initial individuals, their function values will be calculated. The function value for an individual is defined as follows:

Function value = $f\left(x_1, x_2, \ldots, x_{N_{var}}\right)$ (3)

2.2 Education

In an N-dimensional optimization problem by using the definition of spherical coordinate system for N-dimensional Euclidean space [21], the algorithm will find a random radial coordinate (r) between $r_{min} = k_1 d$ and $r_{min} = k_1 d$, where d is the Euclidian distance between the origin and individual, and k_i, as an algorithm parameter, is the weighting factor. In addition, the algorithm will find $N-1$ random angular coordinates $(\theta_1, \theta_2, \ldots., \theta_{N-1})$ where θ_{N-1} will be found between 0 and 2π radians and the other angles will be selected between 0 and π radians.

2.3 Consultation

In this scenario, the new group of variables will take the place of the individual variables. However, nothing will change if the new set of variables does not have a better function value. The following method is used to determine how many random variables will be altered:

$$N_c = round\{\sigma \times N_{var}\} \quad (4)$$

The number of random variables N_C that may be altered during the consultation process is determined by the consultation factor, which acts as an algorithmic parameter and is denoted by the symbol.

2.4 Field changing probability

This approach sorts each field based on its expert individual function value, as shown below:

$$\text{Sort fields} = [field_1, field_2, \ldots., field_n] \quad (5)$$

Whereas $field_1$ and $field_2$'s expert individuals, respectively, have the worst and best function values among the others. Following that, it is possible to calculate the changing probability for each field as follows:

$$P_i = \frac{O_i}{N_{field}+1} \quad (6)$$

Where P_i and O_i are the field-changing probability and the sort order for the i-th field, respectively. After that, by generating a random number between 0 and 1, the following expression is checked, and if the expression is satisfied, the field changing for one of the individuals in this field occurs:

$$\text{if rand} \leq P_i \rightarrow \text{field changing occurs} \quad (7)$$

In the field-changing procedure, a selection probability for each person will be defined in accordance with the function value as follows:

$$P.S_j = \left| \frac{f(Individual_j)}{\sum_{k=1}^{N_{ind}} f(Individual_k)} \right| \quad (8)$$

Where $P.S_j$ the selection probability is for the j-th individual and N_{ind} is the number of individuals in the selected field.

2.5 Finalization

Individuals' positions alter as a result of dialogue and education processes. The function values of the people will therefore be calculated in this phase, and if one of the stopping requirements is satisfied, the algorithm will be ended; otherwise, it will proceed to step 2. The following are the stopping criteria:

(A) The maximum number of iterations has been reached.

(b) There have been as many function evaluations as possible.

(c) Function tolerance is reached when the average relative change in the objective function value across stall iterations is smaller.

The advantages of HBBO are

1- This technique demonstrates that there are no imaginative resource restrictions for optimization, and that, with careful consideration, two seemingly unrelated scientific fields can merge to produce amazing outcomes.

2- Based on the results of the experiments, it can be concluded that HBBO performs better than other optimization algorithms in terms of algorithm dependability, result correctness, and convergence speed.

3- The outcomes demonstrate that HBBO is the fastest and least CPU-intensive optimization procedure.

4- HBBO is simple to use and effective at resolving a wide range of challenging real-world optimization issues.

Applications of HBBO:

1) FoSIL at CheckThat! 2022: Using Human Behaviour-Based Optimization for Text Classification [5].
2) Solving the Manufacturing Cell Design Problem Using Human Behavior-Based Algorithm Supported by Autonomous Search [6].
3) HBBO-based Intelligent Setting and Coordination of Directional Overcurrent Relays Considering Different Characteristics [7].
4) Human Behavior Based Optimization Supported With Self-Organizing Maps for Solving the S-Box Design Problem [8].
5) Reactive and Active Power Losses Minimization using Human Behavior Based Optimization [9].

6. SEEKER OPTIMIZATION ALGORITHM (SOA) [10]

The Seeker Optimization Algorithm (SOA) is a metaheuristic optimization algorithm inspired by the social foraging behavior of certain animal species such as bees, ants, and birds. The algorithm was proposed by K.S. Lee and colleagues in 2015.

In the SOA, a population of "seekers" (i.e., potential solutions) move in a search space according to a set of rules that mimic the behavior of foraging animals. The algorithm is designed to balance the exploration of the search space with the exploitation of promising regions and to adapt to changes in the environment (i.e., the fitness landscape) during the search process.

The basic steps of the SOA are as follows:

1. Initialization: Generate an initial population of seekers randomly in the search space.
2. Foraging: Each seeker explores the search space by moving randomly with a certain probability. At the same time, each seeker also attracts other seekers to its location based on its fitness value (i.e., how good the solution is). This mimics the social attraction behavior of foraging animals.
3. Updating: After a certain number of iterations, the position of each seeker is updated based on its own movement and the attraction of other seekers. The fitness of each seeker is also evaluated and updated.
4. Termination: The search process continues until a stopping criterion is met (e.g., a maximum number of iterations is reached, or a satisfactory solution is found).

The SOA has been applied to various optimization problems, including function optimization, feature selection, and clustering. It has shown promising results compared to other metaheuristic algorithms such as particle swarm optimization and genetic algorithms, especially in high-dimensional and multimodal optimization problems.

Cloud Theory

The Ex is the location at U that corresponds to the cloud's center of gravity. En is a measurement of how widely a notion is used in discourse. He measures the dispersion of the cloud droplets and is the entropy of the .entropy En

The following procedure, known as the basic normal cloud generator, produces the cloud with n cloud droplets given the three parameters (Ex, En, and He) of a normal cloud model [10].

Algorithm 1. Basic normal cloud generator

Input: $Ex,\ En,\ He$, n

Output: $\{(x_1\ ,\ \mu_1), \ldots\ldots, (x_n\ ,\ \mu_n)\}$

for i =1 to n

$En' = RANDN\ (En\ , He)$

$x_i = RANDN\ (Ex\ , En)$

$\mu_i\ = e^{\frac{-(x_i - Ex)^2}{2(En\prime)^2}}$

$cloud\ (x_i\ , \mu_i)$

end.

Here, the function RANDN (a, b) produces a normally distributed random number with mean a and standard deviation b. the $cloud\ (x_i\ , \mu_i)$ is the i th cloud drop in the universe.

In the SOA, every seeker has a start position vector $\vec{c}$, which may be viewed as the expected value $Exthe$ of the cloud model, as the start location to find the next solution. Moreover, each seeker holds a search radius $\vec{r}$ which is equivalent to the En' of the cloud model, a trust degree $\vec{\mu}$ described by the membership degree of the cloud model, and a search direction $\overrightarrow{d}$ showing him where to go.

At each time step t, the search decision-making is conducted to choose the four parameters and the seeker moves to a new position $\vec{x}(t+1)$. The update of the position from the start position is a process of uncertainty reasoning, and is determined by a like Y-conditional cloud generator [10] as follows:

$$x_{ij}(t+1) = C_{ij} + d_{ij} r_{ij}\ (-\ln(\mu_{ij}))^{0.5} \qquad (1)$$

Where “i” is the index of seekers, and “j” is the index of variable dimensions.

The pseudocode of the main algorithm is presented as follows.

```
begin
    t•0;
    generating S positions randomly and uniformly;
    repeat
    evaluating each seeker;
    giving search parameters: start position, search
      direction, search radius, and trust degree;
    updating positions using (1);
    t•t+1;
```

until $t = T_{max}$
end.

Algorithm Parameters

1- Start Point Vector

Intuitively, the start position vector $\vec{c}$ is set to the current position $\vec{x}(t)$. Inspired by PSO, Every seeker contains a memory storing its own best position so far $\vec{p}$ and a global best position $\vec{g}$ obtained through communication with its fellow neighbor seekers.

$$\vec{c} = \vec{x}(t) + \emptyset_1(\vec{p}(t) - \vec{x}(t)) + \emptyset_2(\vec{g}(t) - \vec{x}(t)). \quad (2)$$

Where $\emptyset_1$ and $\emptyset_2$ are real numbers chosen uniformly and randomly in the interval [0, 1].

2- Search Direction

Each seeker has four significative directions: local temporal direction dlt, local special direction dls, global temporal direction dgt, and global special direction $\vec{d}_{gs}$, respectively.

$$\vec{d}_{lt} = \begin{cases} \text{sign}(\vec{x}(t) - \vec{x}(t-1)) & if\ fit(\vec{x}(t)) \geq fit(\vec{x}(t-1)) \\ \text{sign}(\vec{x}(t-1) - \vec{x}(t)) & if\ fit(\vec{x}(t)) < fit(\vec{x}(t-1)) \end{cases} \quad (3)$$

$$\vec{d}_{ls} = \text{sign}(\vec{x}'(t) - \vec{x}(t)) \quad (4)$$

$$\vec{d}_{gt} = \text{sign}(\vec{p}(t) - \vec{x}(t)) \quad (5)$$

$$\vec{d}_{gs} = \text{sign}(\vec{g}(t) - \vec{x}(t)) \quad (6)$$

Where sign (·) is a signum function, $\vec{x}'(t)$ is the position of the seeker with the largest fitness in a given neighborhood region, and fit ($\vec{x}(t)$) is the fitness function of $\vec{x}(t)$.

Then, the search direction is assigned depending on the four directions, we can give the search direction as follows.

$$\vec{d} = sign\left(\omega\left(sign\left(fit(\vec{x}(t)) - fit(\vec{x}(t-1))\right)\right)(\vec{x}(t) - \vec{x}(t-1)) + \emptyset_1(\vec{p}(t) - \vec{x}(t)) + \emptyset_2(\vec{g}(t) - \vec{x}(t))\right) \quad (7)$$

Where ω is the inertia weight and $\omega = (T_{max} - t)/T_{max}$. $\emptyset_1$ And $\emptyset_2$ are real numbers chosen uniformly and at random in a given interval [0, 1].

3- Search Radius

It is essential but challenging to provide a search radius in a logical manner.

Algorithm 2. The cloud-based method of the search radius

$$En_r = \vec{x}_{max} - \vec{x}_{min}$$
$$He_r = \frac{En_r}{10};$$
$$\overrightarrow{r'} = RANDN(En_r, He_r);$$
$$\vec{r} = RANDN\left(0, \overrightarrow{r'}\right);$$

Where $\vec{x}_{max}\ and\ \vec{x}_{min}$ are the positions with the maximum fitness and the minimum fitness within its fellow neighbor, respectively. Such as, the En may be viewed as the "known" region of the problem domain. The function $RANDN\left(0, \overrightarrow{r'}\right)$ is given as real numbers chosen uniformly and randomly in a given interval $\left(0, \overrightarrow{r'}\right)$.

To decrease computing time, the simple method of search radius was expressed as $\vec{r} = RANDN(0, En_r)$ where En_r is presented as ALGORITHM 2. That is to say, fuzzy logic was used to deal with uncertainty reasoning.

4- Trust Degree

The parameter μ is, in fact, the grade of membership from the cloud model and fuzzy set theory. According to the discussion in section 1, the uncertainty rule of intelligent search is described as "If {fitness is large}, Then {search radius is small}".linear membership function was used for "large" of "fitness". Namely, it is directly proportional to the fitness $\vec{x}(t)$ or the index of the ascensive sort order of the fitness of $\vec{x}(t)$ (we applied the latter in our experiments). That is, the best position so far has the maximum μ_{max} , while the other position has a μ<1.0, and the worst position so far has the minimum μ_{min} . The expression is presented as (8) and (9).

$$\mu_i = \mu_{max} - \frac{S - I_i}{S-1}(\mu_{max} - \mu_{min}). \quad (8)$$

$$\mu_{i,j} = RAND(\mu_i, 1) \quad (9)$$

Where S is the neighbor search group size, and I_i is the index (sequence number) of $\vec{x}_i$ after sorting the finesses of neighbor seekers in ascending order.

The advantages of SOA are:
1- Cloud theory effectively satisfies the requirements of real-life scenarios because the uncertainty in transition is preserved.
2- Has already been applied successfully in data mining [10] and intelligent control [9].
3- Significantly different from the search methods now in use.
4- Found the global optimum more quickly, robustly, and effectively than GA and PSO.
5- When addressing many classes of problems with various degrees of complexity, performed really well, convergent to nearly global optimal solutions.

The limitations of SOA are:
1- The cloud theory [8] is taken from fuzzy logic theory and advanced from it, but it addresses the shortcomings of strict specification and excessive certainty.
2- Appearing in widely used transition models, it conflicts with the human recognition process.

Applications of SOA:
1- Seeker Optimization Algorithm for Optimal Reactive Power Dispatch [11].
2- Seeker Optimization Algorithm for Digital IIR Filter Design [12].
3- Reactive power dispatch considering voltage stability with seeker optimization algorithm [13].
4- Seeker optimization algorithm for tuning the structure and parameters of neural networks [14].
5- Solution of economic dispatch problems by seeker optimization algorithm [15].
6- Seeker optimization algorithm: A novel stochastic search algorithm for global numerical optimization [16].

7. TABU SEARCH ALGORITHM (TSA) [17]

Tabu search algorithm (TSA) is a metaheuristic algorithm that is used for solving combinatorial optimization problems. TSA is based on the concept of maintaining a list of recently visited solutions, known as the "tabu list", to avoid revisiting them in the search process.
The basic idea of the TSA is to start with an initial solution and then iteratively improve it by exploring neighboring solutions. In each iteration, the algorithm evaluates a set of candidate solutions, which are generated by applying certain moves or transformations to the current solution.
However, the search process in TSA is guided by a set of rules that control the exploration of the solution space. These rules are defined by a set of tabu conditions that restrict the moves that can be made in the search. The tabu conditions are designed to prevent the algorithm from getting trapped in local optima, by avoiding moves that lead to previously visited solutions or violate certain constraints.
The key components of TSA are the objective function, which defines the optimization problem, the neighborhood structure, which defines the set of moves that can be made to generate new solutions, and the tabu list, which stores the recently visited solutions.
The TSA algorithm is iterative and terminates when a stopping criterion is met, such as a maximum number of iterations or a satisfactory solution is found. At the end of each iteration, the tabu list is updated with the current solution and the algorithm moves to the next iteration.
TSA has been successfully applied to a wide range of optimization problems, including scheduling, routing, packing, and design problems. However, like other metaheuristic algorithms, its performance depends on the choice of parameters and the problem instance. Therefore, proper tuning of the algorithm parameters is essential for achieving good results.

The advantages of TSA are:
1. It has been utilized to solve challenging issues, particularly those involving combinatorial optimization.
2- Depending on the specific problem type and the type of solutions (within the set of good solutions) sought, different approaches for diversification and intensification are used.
3- The optimization method may be tuned via aspiration criteria.
4- How the tabu search method performs on combinatorial optimization issues like the scheduling of jobs in a shop and the traveling salesman problem.

The limitations of TSA are:
1- Local Optima: Tabu search may get stuck in local optima, especially in complex and multimodal optimization problems.
2- Parameter Tuning: Tabu search requires the specification of various parameters, such as the tabu list size, aspiration criteria, and diversification strategies.

3- Computational Complexity: The computational complexity of Tabu search can be high, especially for large-scale optimization problems. As the problem size increases, the number of possible solutions and the search space grow exponentially, making it difficult to find a good solution within a reasonable amount of time.
4-Lack of Theoretical Guarantees: Tabu search is a heuristic algorithm, meaning it does not provide theoretical guarantees on finding the global optimum.

Applications of TSA:
1- A TABU SEARCH ALGORITHM TO SOLVE A COURSE TIMETABLING PROBLEM [18].
2- A Tabu search heuristic for smoke term curation in safety defect discovery [19].
3- A tabu search algorithm for scheduling pharmaceutical packaging operations [20].
4- Tabu search algorithms for water network optimization [21].
5- Designing a Multiobjective Human Resource Scheduling Model Using the Tabu Search Algorithm [22].

8. CONCLUSION

Human-based meta-heuristic algorithms, advantages, limitations, and applications are presented. The paper briefly surveys some different human-based meta-heuristic algorithms aiming to solve optimization problems. Some Human-based algorithms are discussed such as Driving Training-Based Optimization (DTBO), Chef-Based Optimization Algorithm (CBOA), Sewing Training-Based Optimization (STBO), Human Behavior-Based Optimization (HBBO), Group Counseling Optimizer (GCO), Seeker Optimization Algorithm (SOA) and Tabu Search Algorithms (TSA)

REFRENCES

[1]. Dehghani, Mohammad, Eva Trojovská, and Pavel Trojovský. "A new human-based metaheuristic algorithm for solving optimization problems on the basis of simulation of driving training process," Scientific Reports 12, no. 1 (2022): 9924
[2]. Trojovská, Eva, and Mohammad Dehghani. "A new human-based metaheuristic optimization method based on mimicking cooking training," Scientific Reports 12, no. 1 (2022): 14861
[3]. Dehghani, Mohammad, Eva Trojovská, and Tomáš Zuščák. "A new human-inspired metaheuristic algorithm for solving optimization problems based on mimicking sewing training," Scientific Reports 12, no. 1 (2022): 17387.
[4]. Ahmadi, Seyed-Alireza. "Human behavior-based optimization: a novel metaheuristic approach to solve complex optimization problems." Neural Computing and Applications 28, no. Suppl 1 (2017): 233-244.
[5]. Ludwig, Andy, Jenny Felser, Jian Xi, Dirk Labudde, and Michael Spranger. "FoSIL at CheckThat! 2022: using human behaviour-based optimization for text classification." Working Notes of CLEF (2022).
[6]. Soto, Ricardo, Broderick Crawford, Francisco González, Emanuel Vega, Carlos Castro, and Fernando Paredes. "Solving the manufacturing cell design problem using human behavior-based algorithm supported by autonomous search." IEEE Access 7 (2019): 132228-132239.
[7]. Behkam, Reza, Behrooz Vahidi, Mahdi Zolfaghari, Mahdi Salay Naderi, and G. B. Gharehpetian. "HBBO-based intelligent setting and coordination of directional overcurrent relays considering different characteristics." In 2020 28th Iranian Conference on Electrical Engineering (ICEE), pp. 1-4. IEEE, 2020.
[8]. Soto, Ricardo, Broderick Crawford, Francisco González Molina, and Rodrigo Olivares. "Human behaviour based optimization supported with self-organizing maps for solving the S-box design Problem." IEEE Access 9 (2021): 84605-84618.
[9]. Parmar, Siddharth, Indrajit Trivedi, R. H. Bhesdadiya, and Pradeep Jangir. "Reactive and Active Power Losses Minimization using Human Behavior Based Optimization." (2016).
[10]. Dai, Chaohua, Yunfang Zhu, and Weirong Chen. "Seeker optimization algorithm." In Computational Intelligence and Security: International Conference, CIS 2006. Guangzhou, China, November 3-6, 2006. Revised Selected Papers, pp. 167-176. Springer Berlin Heidelberg, 2007.
[11]. Dai, Chaohua, Weirong Chen, Yunfang Zhu, and Xuexia Zhang. "Seeker optimization algorithm for optimal reactive power dispatch." IEEE Transactions on power systems 24, no. 3 (2009): 1218-1231.
[12]. Dai, Chaohua, Weirong Chen, and Yunfang Zhu. "Seeker optimization algorithm for digital IIR filter design." IEEE transactions on industrial electronics 57, no. 5 (2009): 1710-1718.
[13]. Dai, Chaohua, Weirong Chen, Yunfang Zhu, and Xuexia Zhang. "Reactive power dispatch considering voltage stability with seeker optimization algorithm." Electric Power Systems Research 79, no. 10 (2009): 1462-1471.
[14]. Dai, Chaohua, Weirong Chen, Yunfang Zhu, Zhiling Jiang, and Zhiyu You. "Seeker optimization algorithm for tuning the structure and parameters of neural networks." Neurocomputing 74, no. 6 (2011): 876-883.
[15]. Shaw, Binod, V. Mukherjee, and S. P. Ghoshal. "Solution of economic dispatch problems by seeker optimization algorithm." Expert Systems with Applications 39, no. 1 (2012): 508-519.
[16]. Dai, Chaohua, Weirong Chen, Yonghua Song, and Yunfang Zhu. "Seeker optimization algorithm: a novel stochastic search algorithm for global numerical optimization." Journal of Systems Engineering and Electronics 21, no. 2 (2010): 300-311.
[17]. Piniganti, Lemasri. "A survey of tabu search in combinatorial optimization." (2014).
[18]. Aladağ, Çağdaş Hakan, and G. Hocaoğlu. "A tabu search algorithm to solve a course timetabling problem." Hacettepe journal of mathematics and statistics 36, no. 1 (2007): 53-64.
[19]. Goldberg, David M., and Alan S. Abrahams. "A Tabu search heuristic for smoke term curation in safety defect discovery." Decision Support Systems 105 (2018): 52-65.
[20]. Venditti, Luca, Dario Pacciarelli, and Carlo Meloni. "A tabu search algorithm for scheduling pharmaceutical packaging operations." European Journal of Operational Research 202, no. 2 (2010): 538-546.
[21]. da Conceicao Cunha, Maria, and Luisa Ribeiro. "Tabu search algorithms for water network optimization." European Journal of Operational Research 157, no. 3 (2004): 746-758.

[22]. Shayannia, Seyed Ahmad. "Designing a Multiobjective Human Resource Scheduling Model Using the Tabu Search Algorithm." Discrete Dynamics in Nature and Society 2022 (2022).
[23]. Singh, Gurasis, and Sahil Sharma. "Mine blast algorithm." Think India Journal 22, no. 16 (2019): 2656-2665.