FACTA UNIVERSITATIS
Series: **Mechanical Engineering** Vol. 21, Nº 3, Special Issue, 2023, pp. 529 - 552
https://doi.org/10.22190/FUME230602024M

Original scientific paper

A NOVEL DISCRETE RAT SWARM OPTIMIZATION ALGORITHM FOR THE QUADRATIC ASSIGNMENT PROBLEM

Toufik Mzili[1], Ilyass Mzili[2], Mohammed Essaid Riffi[1], Dragan Pamucar[3,4], Vladimir Simic[5], Mohamed Kurdi[6]

[1]Department of Computer Science, Faculty of Science, Chouaib Doukkali University, EI Jadida, Morocco
[2]Department of Management, Faculty of Economics and Management, Hassan First University, Settat, Morocco
[3]Department of Operations Research and Statistics, Faculty of Organizational Sciences, University of Belgrade, Belgrade, Serbia
[4]College of Engineering, Yuan Ze University, Taiwan
[5]Faculty of Transport and Traffic Engineering, University of Belgrade, Belgrade, Serbia
[6]Faculty of Informatics Engineering, Idlib University, Idlib, Syria

Abstract. *The quadratic assignment problem (QAP) is an NP-hard problem with a wide range of applications in many real-world applications. This study introduces a discrete rat swarm optimizer (DRSO)algorithm for the first time as a solution to the QAP and demonstrates its effectiveness in terms of solution quality and computational efficiency. To address the combinatorial nature of the QAP, a mapping strategy is introduced to convert real values into discrete values, and mathematical operators are redefined to make then suitable for combinatorial problems. Additionally, a solution quality improvement strategy based on local search heuristics such as 2-opt and 3-opt is proposed. Simulations with test instances from the QAPLIB test library validate the effectiveness of the DRSO algorithm, and statistical analysis using the Wilcoxon parametric test confirms its performance. Comparative analysis with other algorithms demonstrates the superior performance of DRSO in terms of solution quality, convergence speed, and deviation from the best-known values, making it a promising approach for solving the QAP*.

Key words: *Discrete rat swarm optimizer, Quadratic assignment problem, Combinatorial optimization, Swarm intelligence*

Received: June 02, 2023 / Accepted August 10, 2023
Corresponding author: Toufik Mzili
Department of Computer Science, Faculty of Science, Chouaib Doukkali University, Avenue Jabran Khalil Jabran, B.P 299-24000, El Jadida,Morocco
E-mail: mzili.t@ucd.ac.ma

1. INTRODUCTION

The quadratic assignment problem (QAP) is a typical combinatorial optimization problem and also an NP-hard problem. Since 1957, when Koopmans and Beckmann [1] first presented the quadratic assignment problem as a combinatorial optimization problem, it has attracted much attention and research from researchers in mathematics, computer science, and many other applications. On the one hand, quadratic assignment problems are widely used in practice, and many real-world problems can be formalized as quadratic assignment problems, such as integrated circuit wiring [2-3], factory location layout [4], typewriter keyboard design, task scheduling [5-6], etc. On the other hand, some classical NP-hard combinatorial optimization problems, such as the traveling salesman problem, the triangulation problem, and the Max Clique problem, can also be transformed into quadratic assignment problems [7-11].

Therefore, it is of great theoretical and practical importance to find an efficient algorithm to solve the quadratic assignment problem. Traditional methods for solving quadratic assignment problems can be divided into two categories: exact algorithms and approximate algorithms.

Exact algorithms are able to find the global optimal solution, but the time required increases sharply with the size of the problem and is not suitable for practical applications.

Approximate algorithms trade accuracy for time, seeking to find a feasible solution as close as possible to the optimal solution in a reasonable amount of computation time.

Heuristics, as typical approximation algorithms, suffer from poor adaptability and as soon as the problem configuration changes, the original method is no longer superior.

The potential of the problem is such that the model has to be redesigned. Moreover, for large-scale complex problems, traditional methods can lead to a "combinatorial explosion", as the problem size increases, and the temporal and spatial complexity of the computation grows exponentially.

Therefore, it is still difficult to design efficient algorithms to solve quadratic assignment problems.

In recent years, with the rapid development of computer science, the evolution of artificial intelligence technologies, and the simulation techniques of nature and predators, a number of new methods have emerged to solve combinatorial optimization problems using the modeling of predator's behaviors in nature, such as hunting, attacking, quarreling, and foraging, as well as their prey, thus providing a new way of thinking to solve quadratic assignment problems.

We distinguish several heuristics and metaheuristics inspired by nature and predator behavior [12-16], physics [17-21], humans [22-23], and evolutionary [24-28]. Moreover, these nature-inspired heuristics and metaheuristics have shown promising results in tackling complex real-world problems, spanning various domains such as logistics, transportation, network design, and scheduling, by mimicking the efficient and adaptive strategies observed in nature and predator-prey interactions, these methods offer innovative approaches to address combinatorial optimization problems with improved efficiency and effectiveness. The fusion of computer science and natural processes opens up exciting possibilities for the advancement of optimization techniques, paving the way for more sophisticated and intelligent problem-solving paradigms in the future. As researchers continue to explore and refine these nature-inspired approaches, we can anticipate significant advancements in solving quadratic assignment problems and other optimization challenges.

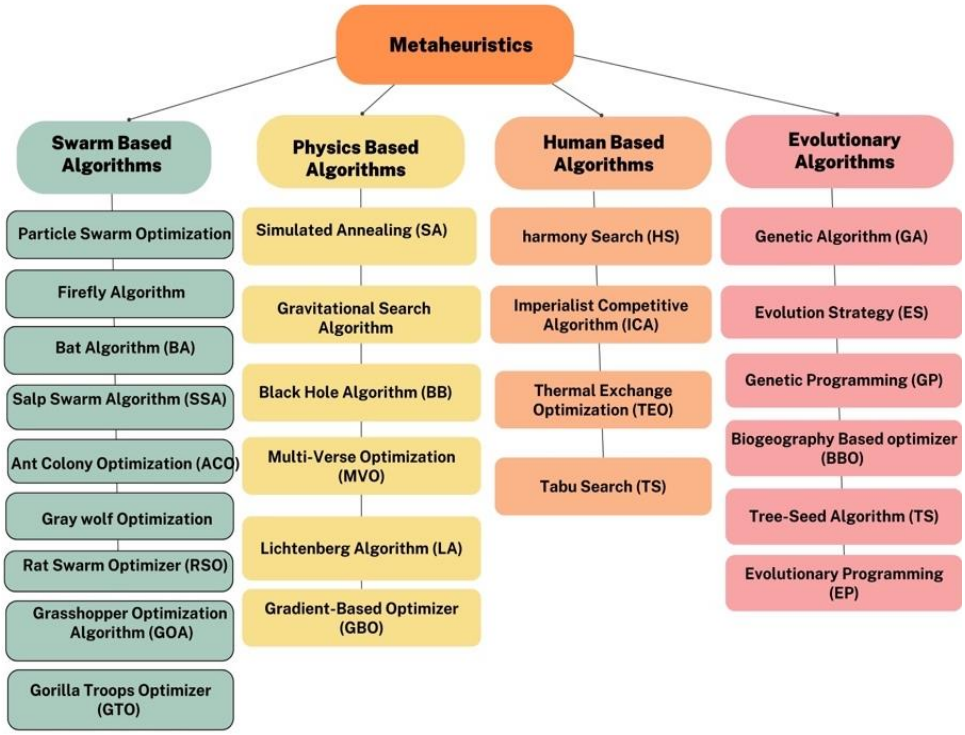

Fig. 1 The best-known metaheuristics

In this paper, we propose a new optimizer based on the hunting and attacking behavior of rats in the wild.

Rats are a species of predator known for their intelligence, they participate in group activities such as hunting and attacking prey. These behaviors will be defined mathematically to create an intelligent and robust optimizer capable of solving more than 38 continuous and nonlinear [16] optimization problems.

This optimizer could give excellent results in solving continuous optimization problems, which are better than most of the known metaheuristics in this context and also in solving the famous discrete traveling salesman problem [29].

In this paper, we will introduce another version to solve the discrete combinatorial optimization problem by redefining the mathematical operators of this optimizer with discrete Maurice Clerc [30] operators, and we will add other strategies to improve the solutions and exploit and explore the discrete search space of the quadratic assignment problem to minimize the total cost of the assignment.

The motivation and benefits of choosing this optimization are:

- The algorithm has a small number of operators compared to other artificial intelligence-based algorithms.
- This AI-based algorithm can access information from the entire search space

and is simple to implement.

- The algorithm has fewer parameters, which reduces its storage requirements and complexity.
- The algorithm maintains a good balance between exploration and exploitation during the search process.
- The algorithm has been shown to be effective at solving 38 continuous and linear problems.
- The algorithm was able to solve the well-known discrete traveling salesperson problem and performed well.

The main contributions of this work are presented as follows:

- This study introduces the DRSO algorithm for the first time as a solution to the QAP.
- The study proposes a mapping strategy to convert real values into discrete values to address the combinatorial nature of the QAP.
- The study redefines mathematical operators to solve combinatorial and discrete optimization problems, specifically the QAP.
- The study proposes a solution quality improvement strategy based on local search heuristics such as 2-opt and 3-opt.
- The effectiveness of the proposed algorithm is demonstrated through simulations and comparisons of test instances from the QAPLIB test library.
- The study proposes a statistical analysis using the Wilcoxon parametric test to validate the performance of the proposed algorithm.

The organization of the remaining sections of this paper is as follows: Section 2 presents related work. Section 3 presents the Quadratic Assignment Problem (QAP). Section 4 presents the presentation of the rat swarm optimizer algorithm and mathematical behavior modeling. Section 5 presents the proposed discrete RSO algorithm and its modification for solving the QAP. Section 6 presents the computer results and analysis, including a Wilcoxon validation test. Finally, in the final section, the concluding remarks and suggestions for future work are presented.

2. RELATED WORK

In recent years, the solution of combinatorial optimization problems by metaheuristics has undergone a great evolution. Metaheuristics have undergone a great evolution, which can be summarized by the speed of development of metaheuristic algorithms thanks to the evolution of the capacities of computing machines. This evolution has allowed, on the one hand, to measure the impact of the search for methods on the problems and, on the other hand, it has had a vision of the result of the methods in a reduced time, especially for the future where the time to find a solution has become more and more requested.

However, the development of metaheuristics can be seen in the appearance of recent methods from different sources of inspiration, such as the algorithm of the hunting mechanism of owls [31] and that of the hunting of anteaters [32], the algorithm of symbiotic interaction strategies adopted by organisms to survive and propagate in the ecosystem [33], the algorithm of searching for squirrels [34], and the phenomenon of food storage in crows

[35]. The appearance of heuristics does not indicate the absence or inefficiency of previous methods, but their interest in appearing is mainly based on a logic of intensification and diversification different from the other methods.

Each algorithm has its own search characteristics that are different from the others, this produces a large variety of metaheuristics.

Metaheuristics are usually presented in the continuous or discrete case, and the majority are presented in the continuous case to solve benchmarks of a continuous function form. However, in our research case, which is interested in solving combinatorial problems to measure the quality of methods in the face of real problems, it is obvious to migrate to the aspect of finding a continuous algorithm in a combinatorial computational environment that does not admit real type values. This migration of an algorithm is known by the adaptation of the algorithm for the combinatorial case, which leads to indicating a logical formula to convey the search strategy of intensification and diversification in the combinatorial environment.

Each new adaptation is applied to the richness of the combinatorial optimization problems in order to be compared to existing heuristic methods.

For example, in the case of solving the quadratic assignment problem, we have seen a great evolution of metaheuristics to solve it, for example PeSOA [36], which is based on a population of penguins, and with random probability, or DCSO [37], discrete cat swarm optimization, which is a metaheuristic method based on the natural behavior of cats. The SSO swallow swarm optimization algorithm [38] is a bio-inspired algorithm based on the behavior of swarms of swallows, The harmony search algorithm [39] inspired by the analogy to music.

In this study, we will present the RSO algorithm as a swarm intelligence algorithm that bases its behavior on attacking and arguing when searching for prey. This approach was originally created to improve continuous functions. This algorithm has been compared to seven continuous algorithms. In fact, the algorithm shows good efficiency when applied to solve various continuous optimization problems.

Implementing and improving the original algorithm to address various challenges. In order to efficiently optimize the traveling salesman problem, Mzili, and Riffi added additional heuristics of mechanisms to this method, to improve the local search capability to ensure efficient exploitation of the search space and escape local minima.

In what follows, we will redefine this optimizer to solve the quadratic assignment problem by taking up these mathematical operators and search mechanisms.

3. QUADRATIC ASSIGNMENT PROBLEM (QAP)

The Quadratic Assignment Problem (QAP) is a mathematical optimization model that was introduced in 1957 by Koopmans and Beckman [1]. It was originally developed to model a plant location problem, and its objective is to find the optimal location of a set of plants taking into account their interactions and distances with other plants.

Since its introduction, the QAP has become a well-known problem in the literature, as it has been studied in a number of research contexts. This is because the QAP presents a generic case that can be applied to other problems.

In the QAP, the data is presented in the form of matrices, the first flow matrix is $F=(f_{i,j})$, where $(f_{i,j})$ is the measure of independence between plant i and plant j, the second matrix

denotes the address of the premises $D=(d_{i,j})$ to which the factories can be assigned, where $d_{i,j}$ represents the distance between premises i and premises j.

Formally, in a QAP problem of size n, i.e., assigning n factories in n location, with $F=(f_{i,j})$ and $D=(d_{i,j})$ are the flow and distance matrices, its solution amounts to optimizing the following function:

- Sets $N=\{1,2,3,\ldots, n\}$
- $Sn= Ø : N \rightarrow M$ represents the set of all permutations.
- Parameters
- $F=(f_{i,j})$ Matrix of flow between facilities i and j
- $D=(d_{i,j})$ Matrix of the distance between locations i and j.

$$\boldsymbol{Min}\ \emptyset \in \boldsymbol{S_n}\ \sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} \sum_{\boldsymbol{j}}^{\boldsymbol{n}} \boldsymbol{f_{i,j}} \times \boldsymbol{d_{\emptyset(i)\emptyset(j)}} \tag{1}$$

A permutation, where i is the place to which facility i is assigned, is used to indicate the assignment of facilities to locations. The cost of assigning facility i to location $Ø(i)$ and facility j to location $Ø(j)$ equals the cost of each individual product $f_{i,j} \times d_{Ø(j)\,Ø(j)}$.

The solution is a permutation Ø of n elements of the search space; each element $Ø(i)$ indicates the location of the proposed assignment for plant i in the assignment Ø. The interest is to find the permutation $Ø(i)$ that minimizes the objective function. For each problem of size n, the number of possible permutations is $n!$ e.g., for a problem of size 10 the number of possible solutions is 3628800, as the size of increases, the computational time increases in parallel, which presents complexity in this NP-hard classified problem [40]. However, QAP has a variety of applications in real life, for example [41] have applied QAP in a university campus, the interest is to find the ideal allocation of buildings to decrease the traffic of steps in the campus.

Another example is [42] to use the same concept to find an assignment of blocks of different departments of a hospital to decrease the patient's route, another use presented by [43] the detection of the right places to install the services of a city such as supermarkets and police stations. QAP could be applied not on geometric assignment problems but on any kind of assignment, for example the problem of assigning electronic components on computer panels, the objective is to reduce the total length of cabling used for interconnecting components.

Example:

Consider the possibility of an installation location issue with four installations (4 emplacements). The following illustration depicts a potential impact: Installation 2 is affected at position 1, Installation 1 is affected at position 2, Installation 4 is affected at position 3, and Installation 3 is affected at position 4. This impact can be written as the permutation $p=2,1,4,3$, which denotes that installations 2 and 1 are both affected at position 1, installations 4 and 3 are both affected at position 3, and installations 4 and 3 are both affected at position 4.

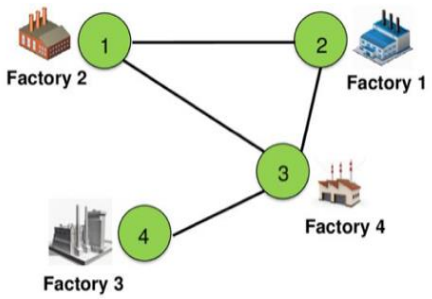

Fig. 2 Facility location problem with four facilities

Tables 1 and 2 provide descriptions of the distances between facilities and the necessary flows between facilities. These distance values are essential for computing the assignment cost of the permutation, as they help determine the overall cost associated with different facility assignments for the given problem.

Table 1 The movements between facilities

Facility i	Facility j	Flow (i,j)
1	2	3
1	4	2
2	4	1
3	4	1

Table 2 Distances between sites

Location *i*	Location *j*	Distance (*i*,*j*)
1	3	53
2	1	22
2	3	40
3	4	55

The assignment cost of the permutation may then be calculated using the formula:

Function_objective*=flow(1,2)×distance(2,1)+flow(1,4)×distance(2,3)+flow(2,4)×distance(1,3)+flow(3,4)×distance(3,4)= 322+240+153+455.*

The Quadratic Assignment Problem (QAP) can be used to optimize the layout of a manufacturing facility by assigning different machines or processes to different locations in the facility in a way that minimizes the total cost of the assignment.

To use the QAP for the installation of machines in a manufacturing facility, you would need to define the set of facilities (machines) and the set of locations (available positions in the facility) and specify the cost matrix C that represents the cost of assigning each machine to each location. The cost matrix can include various factors that contribute to the total cost of the assignment, such as the distance between locations, the setup time or changeover time between different products, and the capacity or output of the machines.

It is important to note that the QAP is a combinatorial optimization problem and is known to be NP-hard, meaning that it is computationally difficult to solve optimally. Therefore, it may be necessary to use approximate solutions or heuristics to find a good, but not necessarily optimal, solution to the problem.

4. RSO ALGORITHM

The rat swarm optimization algorithm is a nature-inspired metaheuristic that mimics the hunting and attacking behavior of a group of rats. This algorithm is based on the collective intelligence of rats and their ability to adapt to their environment. To model this behavior, the algorithm uses a population of "rats" that move through a search space. Each rat has a certain position, which determines its movement in the search space. The rats are also assigned a fitness value, which indicates their probability of finding a solution to the problem at hand. The rats in the swarm are able to communicate with each other and share information about their position and fitness value. This allows them to collaborate and search for the optimal solution to the problem. When a rat finds a good solution, it becomes a "leader" and the other rats follow it, which increases the chances of finding a better solution. Rats also exhibit "offensive" behavior, that is, they aggressively search for solutions in areas of the search space that have not yet been explored. This combination of collaborative and aggressive search behavior allows the rat swarm optimization algorithm to efficiently explore the search space and find good solutions to complex optimization problems. Rats are social predators that prefer to live in groups and perform their many tasks together, including hunting, attacking, and foraging.

The two behaviors that serve as the basis for this bio-inspired algorithm are:

Hunting behavior: in which rats hunt their prey in packs. To locate the prey, the group members designate a captain each time they think they have located it, and they follow him. However, each time they change captains, they cover the entire area.

The behavior of dispute with the prey: in order to hunt their prey, the rats enter into dispute with these last ones, this dispute can cause in several cases the death of certain rats which can translate to the cancellation of a certain solution.

4.1. Mathematical and logical modeling of behavior

This section explains the chasing and fighting behavior of rats.

- **Prey Pursuit:**

Rats typically hunt their prey in packs due to their agonistic social behavior, which makes them sociable. We assume that the finest searcher knows the location of the prey in order to define this behavior quantitatively. The best searcher found so far can be updated by other searchers. The following equations are proposed to model this mechanism:

$$Loc = \delta \times Loc_i + \beta \times (loc_{Best} - loc_i) \tag{2}$$

Here, loc_{Best} is the best optimal solution and loc_i specifies the locations of the rats, while the parameters δ and β are determined as follows:

$$\delta = \theta - \rho\left(\frac{\theta}{Max_{Iteration}}\right), 1 \le \theta \le 5, \rho = 1,2,3, \dots, Max_{Iteration} \tag{3}$$

Therefore, the two parameters δ and β are sensitive to good exploration and exploitation throughout the iteration, while δ and β are random values between [1, 5] and [0, 2].

- **combating a prey**

In many cases, the chase ends with the death of some rats. The following equation was presented as a mathematical definition of this process:

$$Loc_{i+1} = |Loc_{Best} - Loc_i| \tag{4}$$

Where Loc_{i+1} specifies the rat's next updated position. The best solution is preserved, and the positions of the other search agents in relation to the best search agent are updated.

In general, the execution steps of the standard RSO algorithm are presented as follows:

Algorithm 1: Basic Discrete RSO

Output: Optimal solution
Input: The initial rat population P,
Initialize RSO parameters: A, C, and R.
Initialize the rat's population P_i where i = 1, 2
Now, calculate the fitness value of each search agent.
Choose the best agent fitness value P_{Best}.
while (k < MaxIteration) **do**
  for each agent search **do**
    Update the positions of current search agents using Equation (4)
  end for
    Update RSO parameters: A, C, and R.
    Check whether any search agent goes beyond the boundary limit of the search space and then amend it.
    Calculate the fitness of each search agent.
    Update P_{Best} if there is a better solution than the previous optimal solution.
  k ← k + 1.
end while
Return P_{Best}

5. Proposed Discrete RSO Algorithm for QAP

The Quadratic Assignment Problem (QAP) is a mathematical optimization problem that involves finding the optimal placement of a set of machines in a manufacturing facility. Standard rat swarm optimization (RSO) is a continuous optimization method that is used to optimize continuous nonlinear functions, but it cannot be used to directly solve discrete problems. To address this, a modified version of RSO, called discrete rat swarm optimization (DRSO), has been developed to solve discrete combinatorial problems, including the QAP. In order to apply DRSO to the QAP, the fundamental equations of RSO must be modified to include position representation, position update equations, and RSO parameters and operators. Additionally, neighborhood search techniques are often used in DRSO to improve the quality of the solution for combinatorial problems. One such technique is the 2-exchange neighborhood function, which is appropriate for use in QAPs.

This function involves exchanging the positions of two machines in the current solution and evaluating the resulting sum of distances between the facilities and the machines. If the sum of distances is improved, the new solution is accepted as the current solution. By utilizing the 2-exchange neighborhood function, the DRSO algorithm is able to effectively search for the optimal solution to the QAP by making small changes to the current solution and evaluating the resulting improvement in the sum of distances. This allows the algorithm to find a high-quality solution to the QAP in a relatively short amount of time.

The subsection that follows will go into further information about this function.

i. Update position

The rat swarm optimization algorithm simulates the movement of virtual rats in an n-dimensional search space (where n is the size of the problem) according to the location described in the basic RSO algorithm. As the rat's search for the optimal solution to the problem, they tend to move towards the best solution found so far, updating their positions (P_i) at each time step (t). During the search process, some rats may be eliminated due to conflicts with other rats.

In the context of the Quadratic Assignment Problem (QAP), each rat represents a potential solution and loc_{Best} represents the best solution found by the i^{th} rat. This solution represents the optimal placement of the machines in the manufacturing facility, minimizing the sum of distances between the facilities and the assigned machines.

ii. Check the position quality

During the search process, the rats may engage in "hunting and fighting" against other rats as they compete for the optimal solution. In some cases, this competition may result in the elimination of weaker rats, or solutions. This process can be modeled as follows: each rat represents a potential solution, and the elimination of a rat corresponds to the abandonment of that solution. This process is analogous to the death of a rat during the search process.

iii. Operator of Discrete RSO

In continuous optimization problems, logical and mathematical operators are applied to real and natural numbers. However, in discrete optimization problems, these operators cannot be used in the same way because they are designed for continuous scenarios. Discrete optimization problems, such as order, sequencing, or permutation optimization, require the use of discrete operators. Therefore, it is necessary to modify the operators in order to apply them to discrete optimization problems.

The addition operator is used to move the current position by one step. In the discrete case, this operator can be represented as a set of permutations that alter the placement of the facilities. This allows the operator to be applied to discrete optimization problems such as the Quadratic Assignment Problem (QAP)

The subtraction operator locBest – loci in the rat swarm optimization algorithm is used to calculate the set of permutations needed to transform the current position of a rat loc_i into the position of the "best" rat loc_{Best}. This is done by subtracting the position of the "best" rat from the position of the current rat, resulting in a new position loc_{Best}.This operator is used to guide the rats towards the optimal solution by allowing them to follow the movements of the "best" rat.

The multiplication operator in the rat swarm optimization algorithm is used to reduce the number of permutations needed to transform the current position of a rat (loc_i).) into the position of the "best" rat loc_{Best}. This operator is defined as the multiplication of a real

number (β) with the set of permutations calculated by the subtraction operator rat ($loc_{Best} - loc_i$).

By multiplying these values, the number of permutations needed to reach the "best" position is reduced, allowing the rats to more efficiently search for the optimal solution.

iv. The objective function

The objective function is used to evaluate the quality of a given solution. It is defined as the sum of the distances between the locations of the facilities and the machines assigned to them, multiplied by the flow between those locations (as seen in Eq. (4)).

Neighborhood search techniques are often used in combinatorial optimization problems to enhance the quality of the solutions. The two-exchange neighborhood function is a reliable technique for use in the Quadratic Assignment Problem (QAP). It involves starting from a random placement of the machines in the facilities and repeatedly exchanging the positions of two machines as long as it results in a more optimized placement.

v. The 2-opt

The algorithm is a general-purpose optimization algorithm that can be applied to a wide range of problems, including the QAP. It works by iteratively improving the current solution by making swaps between pairs of facilities and their assigned machines. By making swaps that reduce the overall cost of the assignment, the algorithm is able to progressively improve the solution and find the optimal placement of the machines in the facilities.

Here is an example of pseudo-code for the 2-opt algorithm for solving the QAP:

Algorithm 2: 2-opt algorithm for the Quadratic Assignment Problem

```
Function 2_OPT_QAP (Solution):
  - solution = initial solution
  - best_solution = solution
  - improved = True
 while improved do
   improved = False
    for i = 0 to n-1 do                    // n: number of facilities
      for j = i+1 to n do
        if cost_of_swapping (solution, i, j) < 0 then
          solution = swap (solution, i, j)
          improved = True
        end if
      end for
    end for
    if cost_of_solution(solution) < cost_of_solution(best_solution) then
      best_solution = solution
    end if
 end while
 return best_solution
```

The final version of the Discrete Rat algorithm presented as follows:

Algorithm 3 Discrete Rat Swarm Optimization with 2-opt for QAP

Require: Distance matrix D, Flows matrix F , number of rats N , maximum number of iterations Imax

Initialize rats positions X randomly ·X, representing an installation of facility

Calculate fitness values F for each rat using Distance matrix D and Flows matrix F.

for t = 1 to Imax **do**

  for i = 1 to N **do**

    Update rat position using equation (3): $X_{new} = a \cdot X_i + B \cdot (X_{best} - X_i)$

    Calculate new fitness value Fnew using updated position

    if $F_{new} < F$ **then**

      if 2-Opt(X_{new}) < X_{new} **then**

        $X_{new} \leftarrow$ 2-Opt(X_{new})

      end if

        $F \leftarrow F_{new}$

        $X_{best} \leftarrow X_{new}$

    end if

  end for

end for

Output: Best solution founded X_{best}

6. Experimental Results and Analysis

In this study, the rat swarm optimization (RSO) algorithm was implemented in the C++ programming language using a quad-core Intel Core i5 processor with 4 GB of RAM. Objects from the QABLIB library were used for testing the algorithm on various instances of the Quadratic Assignment Problem (QAP). The instances used for testing ranged in size from 12 to 100, as indicated by the number in the instance name (e.g. the instance "els19" represents an instance with 19 facilities).

To compare the performance of the RSO algorithm with other metaheuristics in the domain, we defined a set of parameters and comparison criteria, as shown in the Table 3:

Table 3 Parameters of Discrete RSO

Parameter	Value
The population of rat size: N	60
δ	A random value between [1, 5]
β	A random value between [0, 1]
Nb iteration	300

Tables 4-7 present the results of the discrete rat swarm optimization (DRSO) algorithm applied to 49 selected benchmark instances from the QAPLIB dataset, including some of the most difficult instances to solve such as tai20a, tai30a, tai40a, tai80a, and tai100a. The DRSO algorithm was run 20 times independently for each dataset to obtain the experimental results. The "Best", "Worst", "Average", and "Dev (%)" values represent the best, worst, average, and deviation of the solution after running the algorithm 30 times, respectively. The deviation of the solution, represented by Dev (%), is calculated using the following formula:

$$\text{Dev}\ (\%) = \frac{\text{average-opt}}{\text{opt}} \times 100 \tag{5}$$

The convergence speed of an optimization algorithm is a measure of how quickly the algorithm converges to a solution. A higher convergence speed means that the algorithm reaches a solution in a shorter amount of time. Population diversity, on the other hand, refers to the variation within a population of solutions. A more diverse population is characterized by a greater average distance between individuals, indicating a wider range of possible solutions.

To test the performance of our optimizer, we will choose some of the most well-known metaheuristics in solving Quadratic Assignment Problem to make a comprehensive comparison.

We provide tests on more than 48 instances, but to make the comparison more meaningful and important, we will compare only the most difficult instances, as well as the instances proposed in the articles of the methods, we are going to compare with them, i.e., the instances of the "Tai...." and "Sko..." families.

The comparison is made with recently developed bio-inspiring metaheuristics more known in the solution of combinatorial optimization problems.

DSSO: The SSO (swallow swarm optimization) [38] method is a swarm intelligence approach that is based on the behavior of swallow swarms.

PHCSO: The parallel hybrid chicken swarm optimization (PHCSO) [44] method is a nature-inspired optimization approach that combines the behavior of chicken swarms with parallel computing techniques.

GBSA: The GBSA (generalized binomial search algorithm) [45] is an optimization algorithm that is based on the principle of binary search.

DBA: The discrete bat algorithm (DBA) [10] is a nature-inspired optimization method that is based on the behavior of bats.

To enrich the comparison and make it real and meaningful, it is necessary to use parametric or non-parametric statistical tests. In this study, we will choose the parametric test most commonly used in this type of study, namely the Wilcoxon signed ranks test. The Wilcoxon signed ranks test a non-parametric statistical technique used to compare two related samples. The Wilcoxon signed rank test is often used when data are not normally distributed and can be applied in situations where parametric tests such as t-tests cannot be used. This article discusses the assumptions underlying this method, how it works, its advantages and disadvantages compared to other methods, and some examples of its application.

In Tables 4-7, the sig column is added to indicate the sign of the average time difference of the DRSO with each other metaheuristic.

In Figs. 3-10 we present a comparison on of deviation and average time between DRSO and others methods.

Table 4 Comparison between DRSO and PHCSO

		PHCSO			DRSO			
Instances	Best-know	Best	Dev	Time	Best	Dev	Time	Sig
Sko42	15812	**15812**	0,355	31,02	**15812**	0.04	3.40	=
Sko49	23386	24124	0.765	78.96	**23386**	0.01	5.31	+
Sko81	90998	91113	0.567	215.91	91034	0.01	36.38	+
Tai12a	224416	**224416**	0	0	**224416**	**0**	0.01	=
Tai15a	388214	**388214**	0	0.33	**388214**	**0**	0.02	=
Tai17a	491812	**491812**	0	0.21	**491812**	**0**	0.05	=
Tai20a	703482	**703482**	0	0.12	**703482**	**0.12**	0.23	=
Tai25a	1167256	**1167256**	0	8.09	**1167256**	**0.34**	0.16	=
Tai30a	1818146	1824318	0.673	8.44	**1818146**	**0.49**	1.68	+
Tai35a	2422002	2428322	0.563	16.19	2429278	**0.23**	9.03	=
Tai40a	3139370	**3139370**	0.6275	17.15	3168134	**0.02**	12.11	=
Tai40b	637250948	**637250948**	0.556	9.61	**637250948**	**0**	1.27	=
Tai50a	4938796	5090356	0.176	12.19	**4938796**	0.78	17.06	+
Tai50b	458821517	458845260	1.344	37.61	**458821517**	0.03	10.01	+
Tai60a	7205962	7351256	0.837	59.87	**7231162**	0.57	27.13	+
Tai80a	13499184	13657560	0.863	62.13	13505690	0.77	19.05	+
Tai100a	21052466	21503812	0.136	86.13	**21052466**	0.12	69.45	+

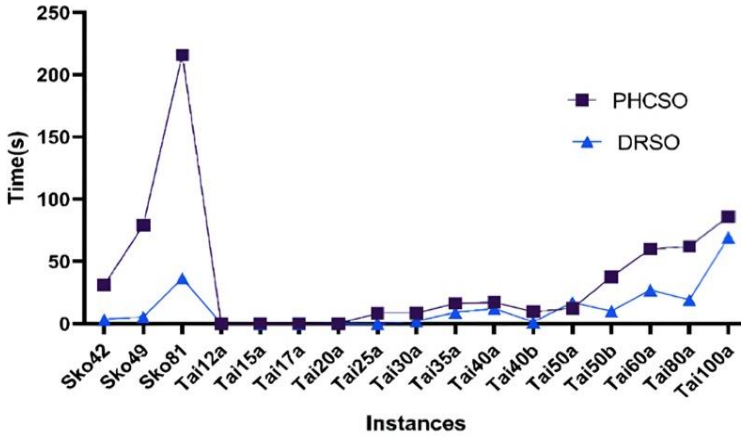

Fig. 3 Comparison of average time between DRSO and PHCSO

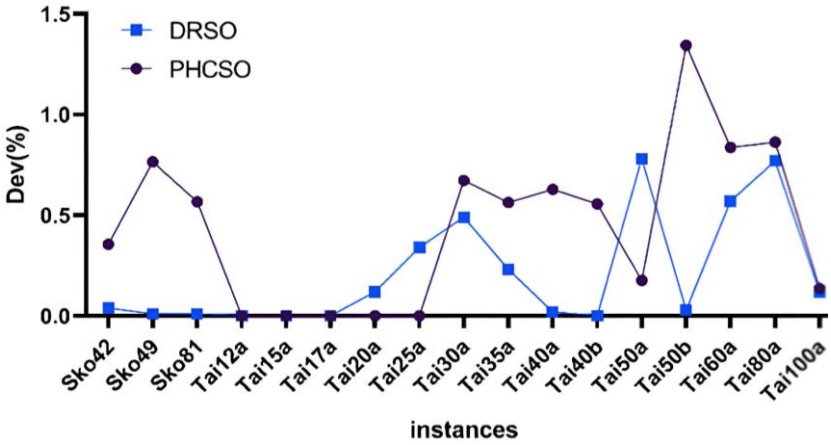

Fig. 4 Comparison of deviation between DRSO and PHCSO

Table 5 Comparison between DRSO and DSSO

		DSSO.			DRSO			
Instances	**Best-know**	**Best**	**Dev**	**Time**	**Best**	**Dev**	**Time**	**Sig**
Sko42	15812	**15812**	0.02	2.80	**15812**	0.04	3.40	=
Sko49	23386	**23386**	0.09	5.50	**23386**	0.01	5.31	+
Sko81	90998	91008	0.08	41.37	91034	0.01	36.38	-
Tai12a	224416	**224416**	0.00	0	**224416**	0	0.01	=
Tai15a	388214	**388214**	0.05	0.05	**388214**	0	0.02	=
Tai17a	491812	**491812**	0.36	0.06	**491812**	0	0.05	=
Tai20a	703482	**703482**	0.62	0.13	**703482**	0.12	0.23	=
Tai25a	1167256	**1167256**	0.95	0.48	**1167256**	0.34	0.16	=
Tai30a	1818146	1825384	0.89	1.29	**1818146**	0.49	1.68	+
Tai35a	2422002	2435966	1.05	4.36	2429278	0.23	9.03	+
Tai40a	3139370	3165320	1.17	9.67	3168134	0.02	12.11	-
Tai40b	637250948	**637250948**	0.00	0.56	**637250948**	0	1.27	=
Tai50a	4938796	4995292	2.00	34.88	**4938796**	0.78	17.06	+
Tai100a	21052466	21044752	2.36	545.31	**21052466**	0.12	69.45	+

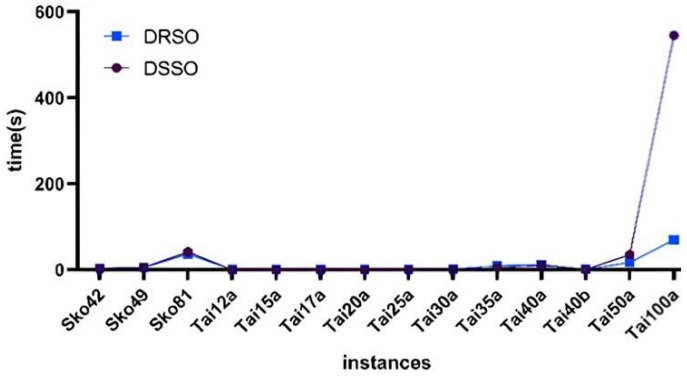

Fig. 5 Comparison of average time between DRSO and DSSOSO

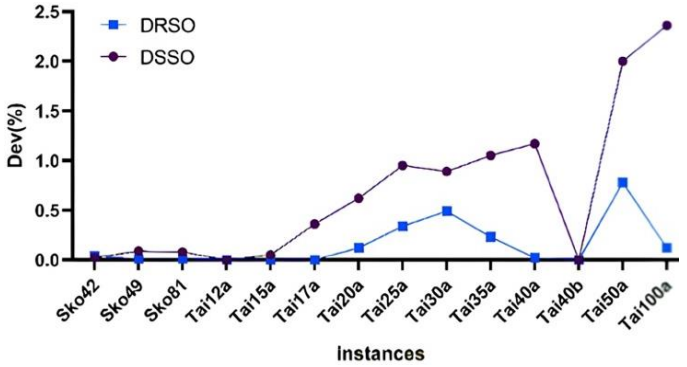

Fig. 6 Comparison of deviation between DRSO and DSSO

Table 6 Comparison between DRSO and GBSA

		GBSA			DRSO			
Instances	Best-know	Best	Dev	Time	Best	Dev	Time	Sig
Sko42	15812	15880	0.99	240	**15812**	0.04	3.40	+
Sko49	23386	23582	1.13	240	**23386**	0.01	5.31	+
Tai12a	224416	**224416**	0.00	0	**224416**	**0**	0.01	=
Tai15a	388214	**388214**	0.00	0	**388214**	**0**	0.02	=
Tai17a	491812	**491812**	0.00	0.12	**491812**	**0**	0.05	=
Tai20a	703482	**703482**	0.37	32	**703482**	0.12	0.23	=
Tai30a	1818146	1841180	2.22	240	**1818146**	0.49	1.68	+
Tai40a	3139370	3215360	3.01	240	**3168134**	0.02	12.11	+
Tai50a	4938796	5084020	3.53	240	**4938796**	0.78	17.06	+

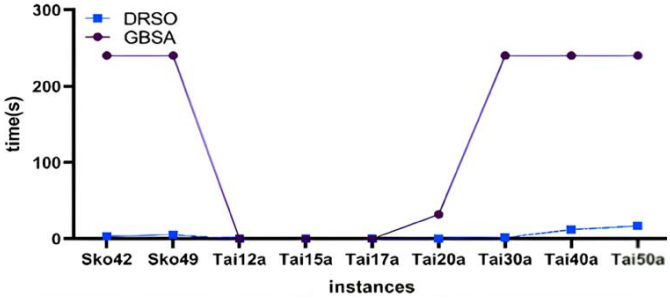

Fig. 7 Comparison on of average time between DRSO and GBSA

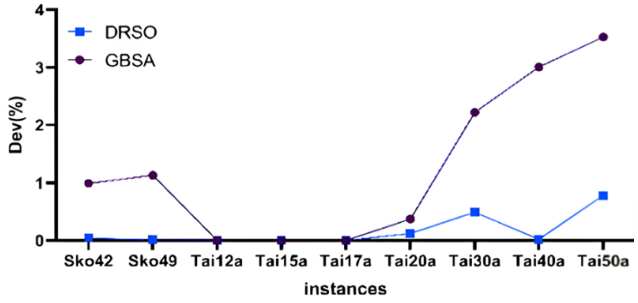

Fig. 8 Comparison on of deviation between DRSO and GBSA

Table 7 Comparison between DRSO and DBA

		DBA			DRSO			
instance	Best-know	Best	Dev	Time	Best	Dev	Time	Sig
Sko42	15812	**15812**	0.30	42.87	**15812**	0.04	3.40	=
Sko49	23386	23421	0.34	97.71	**23386**	0.01	5.31	+
Sko56	34458	34524	0.46	159.8	**34458**	0.01	17.33	+
Sko64	48498	48656	0.44	265.63	**48498**	0.10	27.05	+
Sko72	66256	66422	0.46	361.59	**66259**	0.01	38.52	+
Sko81	90998	91252	0.45	512.73	**90998**	0.01	36.38	+
Tai12a	224416	**224416**	0.00	0.00	**224416**	0	0.01	=
Tai15a	388214	388214	0.00	0.50	**388214**	0	0.02	=
Tai17a	491812	**491812**	0.00	0.39	**491812**	0	0.05	=
Tai20a	703482	**703482**	0.85	0.18	**703482**	0.12	0.23	=
Tai25a	1167256	1172754	1.51	12.10	**1167256**	0.34	0.16	+
Tai30a	1818146	1831272	1.34	20h25	**1818146**	0.49	1.68	+
Tai35a	2422002	2438440	1.79	35.43	**2429276**	0.23	9.03	+
Tai40a	3139370	3139370	2.02	51.12	**3168124**	0.02	12.11	-
Tai40b	637250948	**637250948**	0.00	14.86	**637250948**	0	1.27	=
Tai50a	4938796	5042654	2.69	100	**4983176**	0.78	17.06	+
Tai50b	458821517	458830119	0.11	126.34	**458821517**	0.03	10.01	+
Tai60a	7205962	7387482	2.73	166.23	**7231162**	0.57	27.13	+
Tai80a	13499184	13810130	2.67	420.62	**13505690**	0.77	19.05	+
Tai100a	21052466	21541326	2.5	1045.27	**21052466**	0.12	69.45	+

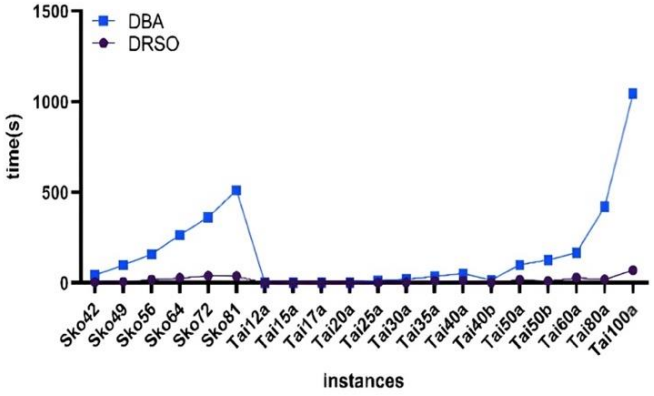

Fig. 9 Comparison on of avg time between DRSO and DBA

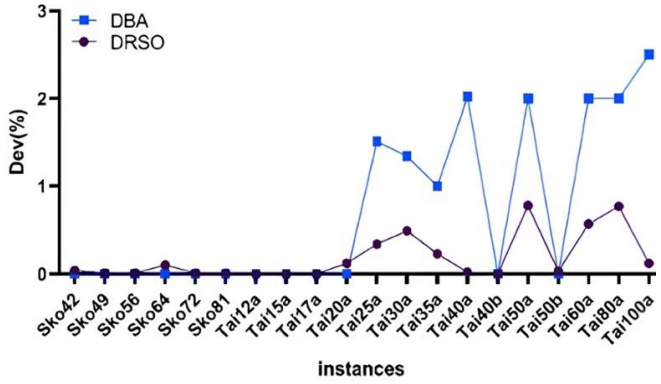

Fig. 10 Comparison on of deviation between DRSO and DBA

6. Discussion and Analysis

To compare the performance of our optimization algorithm with the other metaheuristics, we will apply the Wilcoxon test [46] with a 95% confidence interval (α=0.05). This test will be conducted twice: first, to compare the difference in Dev (%) values between the two algorithms for comparison and ranking; and second, to compare the average execution time and justify the comparison of convergence speed. Instances with similar values or that are easy to solve for both algorithms will not be considered.

N denotes the number of test cases, and W+ represents the scores of cases where the proposed algorithm performs the best. W- represents the sum of the scores of the cases where the proposed algorithm performs worse than the comparative algorithm. The *p*-value is compared to the critical value $\delta = 0.05$ in the Wilcoxon signed-rank test. If the *p*-value $\leq \delta$, it indicates a significant difference in performance between the two algorithms. However, if the p-value $> \delta$, there is no significant difference in performance between the two algorithms.

In Tables 8 and 9, we conducted the Wilcoxon signed rank tests to assess and compare the deviation and average time of different metaheuristics.

Table 8 Wilcoxon signed rank test applied to the deviation of metaheuristics

Comparison	N	W-	W+	P-value	Significantly
DRSO vs DBA	20	-16.00	173,0	<0,001	YES
DRSO vs HPCSO	17	0	200	0,044	YES
DRSO vs GBSA	9	0	39	0,031	YES
DRSO vs DSSO	14	-3	99	<,001	YES

Table 9 Wilcoxon signed rank test applied to the Avg time of metaheuristics

Comparison	N	W-	W+	P-value	Significantly (P < 0.05)?
DRSO vs DBA	20	-3,00	187	<0,001	YES
DRSO vs HPCSO	17	-8	145	<0,001	YES
DRSO vs GBSA	9	-3	42	0,020	YES
DRSO vs DSSO	14	-54,50	50,50	0,915	NO

In order to study the quality of DRSO, we will describe each comparison separately according to the tables above.

The comparison of DRSO with other methods is based on three important factors: the deviation of each algorithm, the average time of convergence to the optimum, and the number of times each algorithm reaches the known optimum of QAPLIB. Each comparison will be related to the curves described above from each table.

- DRSO vs HPCSO

Starting with the comparison with PHCSO in Table 4, we found that the results found by DRSO are 50% better (9 out of 18 tests) than PHCSO and are 50% equal (9 out of 18 tests), while the convergence is 100% better (18 out of 18 tests), which means that DRSO requires less time and iteration to converge to the optimum. On the other hand, the deviation of DRSO in the 18 tests is less than that of HPCSO at 100% which shows that the gap between the results obtained by this method and the best-known value of QABLIB is very large.

Concerning the ability of the two algorithms to attract the optimum value of QABLIB: DRSO was able to attract the best-know value in 13 instances among the 18 test instances at 72.22% while HPCSO found the optimum for 8 instances among 18 i.e. at 44.44% with a deference of 27%. The curves of Deviation and Time in Fig. 3 and 4 show a large difference between the two algorithms, and also show that the Dev (%) value of DRSO is very close to 0 in almost all instances which justifies that the latter was able to find the optimum or almost in most of the tested instances.

- DRSO vs DSSO

In the comparison with DSSO in Table 5, we found that the results found by DRSO are 35.71% better (5 out of 14 tests) than DSSO, 50% equal (7 out of 14 tests), and 14.28% (2

out of 14 tests) weak, while the convergence is 57% better (8 tests out of 14) than DSSO while it is weak at 42.85% (6 tests out of 14) which means that the methods converge quite close and require less time and iteration to converge to the optimum.

On the other hand, the difference in deviation between the two methods is significant and we can see that the deviation values of DRSO are better in almost all instances at 92.8% (13 instances out of 14) which shows that the difference between the results obtained by these methods and the best-known value of QABLIB is very large.

Regarding the ability of the two algorithms to attract the optimal value of QABLIB: DRSO was able to attract the best-known value in 13 instances among the 18 test instances at 72.22% while DSSO found the optimum for 8 instances among the 18, i.e. at 44.44% with a deference of 27%.

The curves id Deviation and Time in Fig. 5 and 6 show a significant difference between the two algorithms, and also show that the value Dev (%) of DRSO is lower than those of DSSO in almost all the instances what justifies that this last one was able to find solutions in the neighborhood or equal to the optimum in the majority of the tested instances, on the other hand shows the curve avg time mounted that the two converge perfectly in a reasonable time and that the two curves are very close in almost all the test.

- DRSO vs GBSA

In the comparison with GBSA in Table 6, the results found by DRSO are 55.55% better (5 tests out of 9) than GBSA, 44.44% equal (4 tests out of 9).

While the convergence is 77.77% better (7 tests out of 9) than GBSA while it is low at 22.22% (2 tests out of 9) which means that DRSO converges quickly and requires less time and iterations to converge to the optimum than GBSA.

On the other hand, the difference in deviation between the two methods is significant and we can see that the deviation values of DRSO are better in almost all instances at 100% (9 instances out of 9) which shows that the difference between the results obtained by these methods and the best-known value of QABLIB is very large and the solutions obtained by DRSO are all closer to the best-know value of QABLIB than those of GBSA this can be clear also in Fig. 7.

Regarding the ability of the two algorithms to attract the optimal value of QABLIB: DRSO was able to attract the best-known value in 8 instances among the 9 test instances at 88.88% while GBSA found the optimum just for 4 instances among the 9 instances, that is at 44.44% with a deference of 44.44%.

The curves of deviation and avg time of Fig. 7 and 8 show a very significant difference between the two algorithms, and also show that the value Dev (%) of DRSO is lower than those of GBSA in almost all the instances which justifies that the latter was able to find solutions in the neighborhood or equal to the optimum in almost the majority of the tested instances on the other hand the curve avg time showed that GBSA is so slow in.

- DRSO vs DBA

Finally, we compare our method with DBA in Table 7, the results found by DRSO are 65% better (13 tests out of 20) than DBA, 30% equal (6 tests out of 20) and 5% (1 test out of 20) weak than DBA.

While the convergence is 100% better (20 tests out of 20) than DBA, which means that DRSO converges quickly and requires fewer time and iterations to reach the optimum compared to DBA.

On the other hand, the difference in deviation between the two methods is significant, and we can observe that the deviation values of DRSO are better in almost all cases at 95%

confidence level (19 cases out of 20). This indicates that the difference between the results obtained by these methods and the best-known value of QABLIB is substantial. Moreover, the solutions obtained by DRSO are all closer to the best-known value of QABLIB compared to those of DBA. This clarity is also evident in Fig. 10.

Regarding the ability of the two algorithms to attract the optimal value of QABLIB: DRSO was able to attract the best-known value in 15 instances among the 20 test instances at 75% while DBA found the optimum only for 7 instances among the 20 instances, i.e. at 35% with a deference of 40%.

Curves of Deviation and Time in Fig. 9 and 10 show a very significant difference between the two algorithms, and also show that the Dev (%) value of DRSO is lower than those of DBA in several instances which justifies that the latter was able to find solutions the neighborhood or equal to the optimum in almost the majority of the tested instances; on the other hand, the avg time curve showed that DBA is very slow in the large instances which contain more facility.

The superior performance of DRSO is attributed to the behavior of the rats and their ability to efficiently share and exploit information, as well as rapidly explore the entire search space.

These statistical analyzes can be evaluated and confirmed by the results of the Wilcoxon signed rank tests carried out and cited in the Tables 8 and 9, in these tests, it will be found that the difference in a deviation between DRSO and the other methods is very significant. at α=0.05 in a 95% confidence interval, which confirms the results of the analyzes carried out below, and justifies that the results and the solutions obtained by DRSO are much closer or equal to the optimum.

On the other hand, the WSR test described in Table 9 confirms that the difference in mean times between DRSO and DBA, GBSA, PHCSO are so significant at α=0.05 in a 95% confidence interval, whereas the difference with DSSO and insignificant and almost zero, which confirms the analyses, and approves that the DRSO converges quickly and requires months of iteration time to find solutions equal to or closer to the optimum, and at the same level as the DSSO but just in convergence time.

7. CONCLUSIONS

In conclusion, the discrete rat swarm optimizer (DRSO) algorithm has demonstrated great potential in solving the discrete quadratic assignment problem (QAP). The algorithm has shown effectiveness in finding high-quality solutions and has outperformed existing algorithms in numerous instances. The algorithm's advantages, such as its simplicity, access to the entire search space, and a balanced approach between exploration and exploitation, contribute to its success. Additionally, the algorithm has proven its capability by solving various discrete problems, including the well-known discrete traveling salesperson problem.

This study's contributions are significant in several aspects. Firstly, it introduces the DRSO algorithm as a novel solution for the QAP, addressing its combinatorial nature. The study proposes a mapping strategy for converting real values into discrete values, redefines mathematical operators suitable for solving combinatorial and discrete optimization problems, and incorporates local search heuristics to enhance solution quality. The effectiveness of the algorithm is demonstrated through extensive simulations and

comparisons using the QAPLIB test library. Furthermore, the study employs statistical analysis, including the Wilcoxon parametric test, to validate the algorithm's performance.

Comparisons with other algorithms, namely PHCSO, DSSO, GBSA, and DBA, further highlight the superiority of DRSO. The results consistently indicate that DRSO outperforms these algorithms in terms of solution quality, convergence speed, and deviation from the best-known values of QABLIB. The curves and statistical tests provide strong evidence supporting the superior performance of DRSO, with its convergence curve consistently outperforming other algorithms.

The success of DRSO can be attributed to the efficient information sharing and exploitation capabilities of the rat swarm. The algorithm effectively explores the entire search space, allowing for rapid convergence to optimal or near-optimal solutions. The statistical analyses, including the Wilcoxon signed rank tests, confirm the significant differences in deviation between DRSO and other methods, reinforcing the notion that DRSO produces solutions that are closer to the optimum.

Future research opportunities for the DRSO algorithm include incorporating additional heuristics or metaheuristics to improve performance, expanding its applicability to larger instances or other combinatorial optimization problems, and studying its behavior in greater detail to understand its strengths and limitations. Moreover, exploring potential applications in domains such as logistics, scheduling, and resource allocation could further contribute to the development and advancement of the DRSO algorithm.

REFERENCES

1. Koopmans, T.C., Beckmann, M., 1957, *Assignment Problems and the Location of Economic Activities*. Econometrica, 25(1), pp. 53-76.
2. Lberni, A., Marktani, M.A., Ahaitouf, A., Ahaitouf, A., 2020, *Adaptation of the Whale Optimization Algorithm to the Optimal Sizing of Analog Integrated Circuit: Low Voltage Amplifier Performances*, Proc IEEE 2nd International Conference on Electronics, Control, Optimization and Computer Science ICECOCS 2020, Kenitra, Morocco, pp. 1-6.
3. Lin, H., Li, P., 2015, *Circuit performance classification with active learning guided sampling for support vector machines*, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 34(9), pp. 1467-1480.
4. Domschke, W., Krispin, G., 1997, *Location and layout planning*, OR Spektrum, 19(3), pp. 181 -194.
5. Mittal, S., Katal, A., 2016, *An Optimized Task Scheduling Algorithm in Cloud Computing*, Proc. IEEE 6th International Conference on Advanced Computing IACC 2016, Bhimavaram, India, pp. 197-202.
6. Mzili, T., Mzili, I., Riffi , M.E., 2023, *Optimizing production scheduling with the Rat Swarm search algorithm: A novel approach to the flow shop problem for enhanced decision making*, Decision Making: Applications in Management and Engineering, 6(2), pp.16 - 42.
7. Silva, A., Coelho, L.C., Darvish, M., 2021, *Quadratic assignment problem variants: A survey and an effective parallel memetic iterated tabu search*, European Journal of Operational Research, 292(3), pp. 1066-1084.
8. Hafiz, F., Abdennour, A., 2016, *Particle Swarm Algorithm variants for the Quadratic Assignment Problems - A probabilistic learning approach*, Expert Systems with Applications, 44, pp. 413-431.
9. Jiyue, E., Liu, J., Wan, Z., 2023, *A novel adaptive algorithm of particle swarm optimization based on the human social learning intelligence*, Swarm and Evolutionary Computation, 80, 101336.
10. Riffi, M.E., Saji, Y., Barkatou, M., 2017, *Incorporating a modified uniform crossover and 2-exchange neighborhood mechanism in a discrete bat algorithm to solve the quadratic assignment problem*, Egyptian Informatics Journal, 18(3), pp. 221-232.
11. Agharghor, A., Riffi, M.E., Chebihi, F., 2019, *Improved hunting search algorithm for the quadratic assignment problem*, Indonesian Journal of Electrical Engineering and Computer Science, 14(1), pp. 143-154.
12. Xu, G.N., Zhang, T., Lai, Q., 2021, *A new firefly algorithm with mean condition partial attraction*, Applied Intelligence, 52(4), pp. 4418-4431.

13. Mirjalili, S.Z., Mirjalili, S., Saremi, S., Faris, H., Aljarah, I., 2017, *Grasshopper optimization algorithm for multi-objective optimization problems*, Applied Intelligence, 48(4), pp. 805-820.
14. Mirjalili, S., Gandomi, A.H., Mirjalili, S., Saremi, S., Faris, H., Mirjalili, S., 2017, *Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems*, Advances in Engineering Software, 114, pp. 163-191.
15. Medjahed, S.A., Ouali, M., 2018, *Spectral band selection using binary Gray Wolf optimizer and signal to noise ration measure*, Lecture notes in networks and systems, pp. 75-89.
16. Dhiman, G., Garg, M., Nagar, A.K., Kumar, V., Dehghani, M., 2020, *A novel algorithm for global optimization: Rat Swarm Optimizer*, Journal of Ambient Intelligence and Humanized Computing, 12(8), pp. 8457-8482.
17. Dowsland, K.A., Thompson, J., 2012, *Simulated annealing*, Handbook of natural computing, Springer, Berlin, Heidelberg, pp. 1623-1655.
18. Sabri, N.M., Puteh, M., Mahmood, M.R., 2013, *A review of gravitational search algorithm*, Int. J. Advance. Soft Comput. Appl, 5(3), pp. 1-39.
19. Abualigah, L., Elaziz, M. A., Sumari, P., Khasawneh, A.M., Alshinwan, M., Mirjalili, S., Gandomi, A.H., 2022, *Black hole algorithm: A comprehensive survey*, Applied Intelligence, 52(10), pp. 11892–11915.
20. Golabian, H., Arkat, J., Tavakkoli-Moghaddam, R., Faroughi, H., 2022, *A multi-verse optimizer algorithm for ambulance repositioning in emergency medical service systems*, Journal of Ambient Intelligence and Humanized Computing, 13(1), pp. 549-570.
21. Pereira, J.L.J., Francisco, M.B., Diniz, C.A., Oliver, G.A., Cunha Jr, S.S., Gomes, G.F, 2021, *Lichtenberg algorithm: A novel hybrid physics-based meta-heuristic for global optimization*, Expert Systems with Applications, 170(15), 114522.
22. Zhao, F., Li, G., Yang, C., Abraham, A., Liu, H, 2014, *A human–computer cooperative particle swarm optimization based immune algorithm for layout design*. In Neurocomputing, 132, pp. 68–78.
23. Samareh Moosavi, S. H., Bardsiri, V. K, 2019, *Poor and rich optimization algorithm: A new human-based and multi populations algorithm*. Engineering Applications of Artificial Intelligence, 86, pp. 165–181.
24. Arnold, D. V., 2002, *Noisy Optimization With Evolution Strategies*, in Genetic Algorithms and Evolutionary Computation, Springer US.
25. Koza, J.R., Poli, R., 2005, *Genetic programming, Search methodologies*, pp. 127-164, Springer, Boston, MA.
26. Simon, D., 2008, *Biogeography-based optimization*, IEEE transactions on evolutionary computation, 12(6), pp. 702-713.
27. Opara, K. R., Arabas, J., 2019, *Differential Evolution: A survey of theoretical analyses*, Swarm and evolutionary computation, 44, pp. 546-558.
28. Katoch, S., Chauhan, S.S., Kumar, V., 2021, *A review on genetic algorithm: past, present, and future*, Multimedia Tools and Applications, 80(5), pp. 8091-8126.
29. Mzili , T., Riffi , M.E., Mzili, I., Dhiman, G., 2022, *A novel discrete Rat swarm optimization (DRSO) algorithm for solving the traveling salesman problem*, Decision Making: Applications in Management and Engineering, 5(2), pp. 287-299.
30. Clerc, M., 2004, *Discrete Particle Swarm Optimization, illustrated by the Traveling Salesman Problem*, in Onwubolu, G.C., Babu, B.V., *New optimization techniques in engineering*, pp. 219-239.
31. Di Blanco, Y. E., Desbiez, A. L. J., Jiménez-Pérez, I., Kluyber, D., Massocato, G. F., Di Bitetti, M. S., 2017, *Habitat selection and home-range use by resident and reintroduced giant anteaters in 2 South American wetlands*, Journal of Mammalogy, 98(4), pp. 1118–1128.
32. Cheng, M.Y., Prayogo, D., 2014, *Symbiotic Organisms Search: A new metaheuristic optimization algorithm*, Computers & Structures, 139, pp. 98-112.
33. Jain, M., Singh, V., Rani, A., 2019, *A novel nature-inspired algorithm for optimization: Squirrel search algorithm*, Swarm and evolutionary computation, 44, pp. 148-175.
34. James, P.C., Verbeek, N.A., 1983, *The food storage behaviour of the northwestern crow*, Behaviour, 85(3-4), pp. 276-291.
35. Geem, Z.W., 2009, *Music-inspired harmony search algorithm*, Studies in Computational Intelligence, Springer Berlin Heidelberg.
36. Mzili, I., Riffi, M. E., Benzekri, F., 2017, *Penguins search optimization algorithm to solve quadratic assignment problem*, Proc. 2nd international Conference on Big Data, Cloud and Applications, pp. 1-6.
37. Bouzidi, A., Riffi, M.E., 2014, *Discrete cat swarm optimization algorithm applied to combinatorial optimization problems*, Proc. 2014 5th Workshop on Codes, Cryptography and Communication Systems WCCCS, El Jadida, Morocco, pp. 30-34.
38. Bouzidi, S., Bouzidi, M., Riffi, M.E., 2019, *Solving the Quadratic Assignment Problem using the Swallow Swarm Optimization Problem*, International Journal of Engineering and Advanced Technology, 8(6), pp. 3116-3120.
39. Gao, X.Z., Govindasamy, V., Xu, H., Wang, X., Zenger, K., 2015, *Harmony search method: theory and applications*, Computational intelligence and neuroscience, 2015, 258491.
40. Krishnamoorthy, M.S., 1975, *An NP-hard problem in bipartite graphs*., ACM SIGACT News, 7(1), pp. 26-26.

41. Loiola, E. M., De Abreu, N. M. M., Boaventura-Netto, P.O., Hahn, P., Querido, T., 2007, *A survey for the quadratic assignment problem*, European Journal of Operational Research, 176(2), pp. 657–690.
42. Tseng, L.Y, Liang, S.C., 2006, *A hybrid metaheuristic for the quadratic assignment problem*, Computational Optimization and Applications, 34(1), pp. 85-113.
43. Mohassesian, E., Karasfi, B., 2017, *A new method for improving the performance of fast local search in solving QAP for optimal exploration of state space*, Proc. 2017 Artificial Intelligence and Robotics IRANOPEN, Qazvin, Iran, pp. 64-72.
44. Semlali, S.C.B., Riffi, M.E., Chebihi, F., 2019, *Parallel hybrid chicken swarm optimization for solving the quadratic assignment problem*, International Journal of Electrical and Computer Engineering, 9(3), 2064.
45. Riffi, M.E., Sayoti, F., 2019, *Hybrid Algorithm for Solving the Quadratic Assignment Problem*, International Journal of Interactive Multimedia & Artificial Intelligence, 5(4), 68.
46. Taheri, S. M., Hesamian, G., 2012, *A generalization of the Wilcoxon signed-rank test and its applications*, Statistical Papers, 54(2), pp. 457–470.