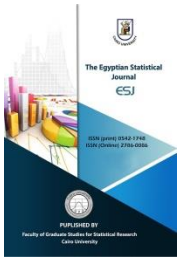
Cairo University

The Egyptian Statistical Journal

Volume(67)-No. 2 -2023
Homepage:mskas.journals.ekb.eg
Print ISSN 0542-1748– Online ISSN 2786-0086



Improved Crow Search Algorithm for Solving Quadratic Assignment Problem

Eman M. Oun[1] and Heba Roshdy[1]

Operations Research & Management
Faculty of graduate studies for statistical research
Cairo University, Giza, Egypt

Submitted: 04-06-2023 Revised: 22-08-2023 Accepted: 23-10-2023

ABSTRACT

In this paper, a crow search algorithm which is a type of swarm intelligence optimization algorithm proposed by simulating the crows' intelligent behavior of hiding and retrieving food is given to solve the Quadratic Assignment Problem (QAP), which is a NP-Complete problem and is assumed one of the complex combinatorial optimization problems related with assigning a set of facilities to a set of locations in condition of minimizing the total assignment cost. The used technique is modified to be allowed to be applied on the desired problem. It is used the Smallest Position Value (SPV) heuristic rule to transform the generated continuous solution from crow search algorithm to a discrete one to be suitable to this kind of problems. Based on ten benchmark problems with different sizes, the computational results explain that the used algorithm is capable to find the optimal and best-known solutions. The proposed method exceeds other illustrated methods. Experimental results show its effectiveness on the quadratic assignment problem.

Keywords:
The Quadratic Assignment Problem, Crow Search Algorithm, the Smallest Position Value Rule, NP- Complete problem.

1. Introduction

The difficulty of locating facilities in specific places where the cost of doing so is determined by distances and interactions between these facilities is known as QAP. In 1957, Koopmans and Beckman presented QAP in an attempt to describe the problem of model facility location. The travelling salesman and graph partitioning problems are examples of classic combinatorial optimization problems that can be expressed as QAPs. It is also appropriate in applications such as layout problems, process communications, computer manufacturing and scheduling [1].

The QAP belongs to the NP-complete problem category and is thought to be one of the most challenging combinatorial optimization problems. Exact solution methods for the QAP have been unsuccessful for large scale problems. As a result, more efforts have been put forth by researchers in getting "inexact" or "heuristic" methods, which give good suboptimal solutions in suitable CPU time.

Recently it is introduced an ideal candidate for meta heuristic approaches [2].

Classical single-solution local search and related algorithms were utilized for the QAP during the early period of heuristic algorithm application (1960–1980) [3, 4]. Later, improved local search algorithms have been presented [5, 6]. Simulated annealing (SA)-based algorithms usually get good results. This is applicable to both the early editions of SA algorithms [7] and improved SA algorithm modifications [8, 9]. The tabu search (TS) methodology based algorithms got more performance. The fast-running tabu search algorithm introduced by Taillard [10] in the early 1990s is still assumed as one of the most successful single-solution-based heuristic algorithms for the QAP Since then, a number of modified variations of TS algorithms have been given: reactive tabu search [11], enhanced tabu search [12], self-controlling tabu search [13], parallel tabu search [14], and other editions [13, 15].

Population-based/evolutionary algorithms are another important class of efficient heuristic algorithms for the QAP. The advantage of this class of algorithms is that these algorithms deal with the sets of solutions instead of single solutions. Genetic algorithms (GA) such as the genetic-local search algorithm [16], the greedy genetic algorithm [17], the genetic algorithm employing cohesive crossover [18], the mimetic algorithm [19], the enhanced genetic algorithm [20], and various GA modifications [21-23] are employed to solve the QAP.

Swarm intelligence algorithms simulate intelligent attitude of physical/biological entities (agents) like particles (particle swarm optimization algorithms [24, 25]), bees (artificial bee colony algorithms [26, 27]) and ants (ant colony optimization algorithms [28]). Finally, the algorithms inspired from real-world phenomena (including those using metaphors) are also used to solve the QAP [29, 30].

The paper is represented as follow. "Methodology" is presented in Section 2, which is divided into three subsections: "Quadratic Assignment Problem," "Crow Search Algorithm," and "Solution Representation." The "Proposed Improved Crow Search Algorithm for Quadratic Assignment Problem" is discussed in Section 3. "Computational findings" is shown in Section 4. Finally, in Section 5, there is a conclusion.

2. Methodology

2.1 Quadratic Assignment Problem

In combinatorial optimization problems, the QAP is a critical challenge. Koopmans and Beckmann first proposed this topic in 1957, and it was classified as an NP-hard problem. The problem of distributing a set of n facilities to a set of n locations in such a way that each location is assigned to exactly one facility and vice versa is referred to as the QAP. Costs of assignment are known for each pair of facilities versus locations. The goal is to reduce the cost of allocating facilities to locations while considering for the sum of distances multiplied by the flows between all facilities. It is described as, an integer linear programming problem, as follows: Consider f_{ij} the flow between facilities i and j, and d_{kp} the distance between locations k and p. The purpose is to determine out [1]:

$$min \sum_{i,j=1}^{n} \sum_{k,p=1}^{n} f_{ij}.d_{kp}x_{ik}x_{jp} \qquad (1)$$

$$s.t. \sum_{i=1}^{n} x_{ij} = 1, \quad 1 \le j \le n \qquad (2)$$

$$\sum_{j=1}^{n} x_{ij} = 1, \quad 1 \le i \le n \qquad (3)$$

$$x_{ij} \in \{0,1\} \qquad 1 \le i,j \le n \qquad (4)$$

A general form for a QAP instance of order n is given by three matrices $F = [f_{ij}]$, $D = [d_{kp}]$ and $B = [b_{ik}]$, the first two matrices define the flows between facilities and the distances between locations, b_{ik} express the allocation costs of facilities to locations. This problem can be defined as:

$$min \sum_{i,j=1}^{n} \sum_{k,p=1}^{n} f_{ij}\, d_{kp}\, x_{ik} + \sum_{i,k=1}^{n} b_{ik}\, x_{ik} \quad (5)$$

$$s.t. \quad (2), (3) and\ (4)$$

Since the linear term of (5) is easy to solve, most authors discarded it. By using permutations, QAP can be formulated as follow: Let S_n be the set of all permutations with n elements and $\pi \in S_n$ Consider f_{ij} the flows between the facilities i and j and $d_{\pi(i)\pi(j)}$ is the distance between location π_i and π_j. If each permutation π represents an allocation of facilities to locations, the problem expression becomes:

$$min_{\pi \in S_n} \sum_{i,j=1}^{n} f_{ij}\, d_{\pi(i)\,\pi(j)} \qquad (6)$$

This formulation is equivalent to the first one presented in Eq. (1) - Eq. (4), since the Eqs. (2) and (3) define permutation matrices $X = [x_{ij}]$ related to S_n elements, as in (6), where, all $1 \le i, j \le n$.

$$x_{ij} = \begin{cases} 1, & if\ \pi(i) = j \\ 0, & if\ \pi(i) \neq j \end{cases} \qquad (7)$$

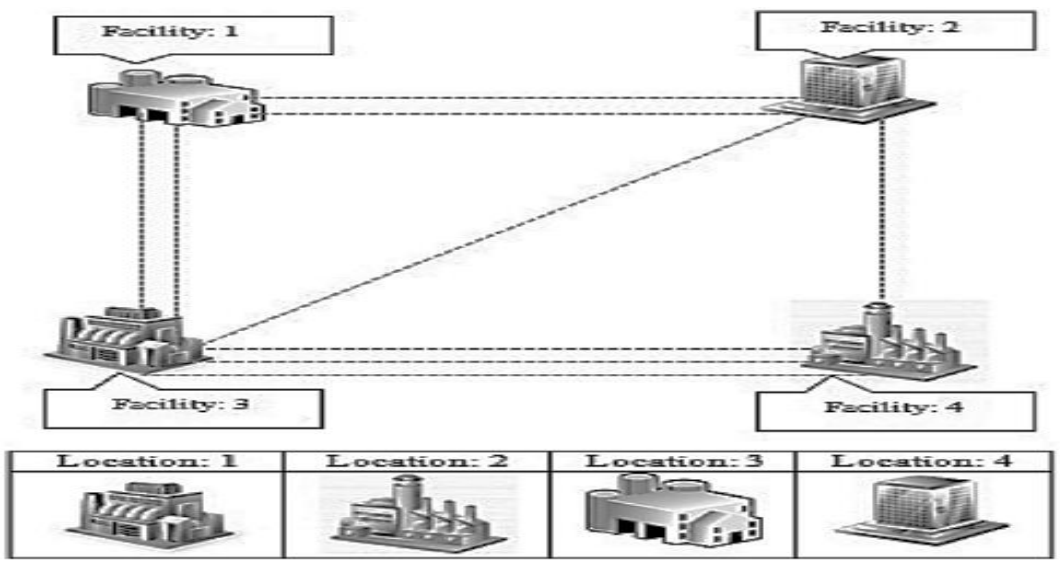


Figure1. Illustrative example for the permutation π = (2, 4, 3, 1) corresponding to the optimal solution of the quadratic assignment problem [31].

Consider the problem of assigning four facilities to four locations and two matrices *A* and *B*, as illustrated in Fig. 1. The dotted line connecting two facilities indicates that there is required flow between them, and the distances between locations are needed to determine assignment cost. One of possible assignment is to place facility 1 to location 3, facility 2 to location 4, and facility 3 to location 1, and facility 4 to location 2. This assignment can be written as the permutation π = (2, 4, 3, 1) but it is not necessarily the optimal one.

A is the flow Matrix: *B* is the distance Matrix:

$$A = \begin{bmatrix} 0 & 2 & 2 & 0 \\ 2 & 0 & 1 & 1 \\ 2 & 1 & 0 & 3 \\ 0 & 1 & 3 & 0 \end{bmatrix} \qquad B = \begin{bmatrix} 0 & 2 & 1 & 3 \\ 2 & 0 & 1 & 2 \\ 1 & 1 & 0 & 3 \\ 3 & 2 & 3 & 0 \end{bmatrix}$$

2.2 Crow Search Algorithm

Crows (or corvids) are the smartest birds. In proportion to their body size, they have the largest brain. Their brain is slightly smaller than a human brain in terms of brain-to-body ratio. There are numerous examples of crows' ingenuity. They have demonstrated self-awareness in mirror tests and have tool-making ability. Crows can remember faces and warn each other when an unfriendly one approaches. Moreover, they can use tools, communicate in sophisticated ways and recall their food's hiding place up to several months later.

Crows have been observed watching other birds, observing where they keep their food, and then stealing it once the owner has left. If a crow has been a victim of theft previously, it will take extra efforts to avoid becoming a victim again, such as changing hiding sites. In fact, they can predict the conduct of a pilferer based on their own experience as a thief and select the best course of action to secure their caches.

Crow Search Algorithm (CSA), a population-based meta-heuristic algorithm, is designed based on the above-mentioned intelligent behaviors. The following are the CSA principles [32]:

- Crows live in groups called flocks.
- Crows remember their hiding spots.
- Crows follow each other to do thievery.
- Crows protect their caches from being stolen by chance.

Let

- *d*: the number of environment dimensions
- *N*: The number of crows (flock size or population size)
- $iter_{max}$: the maximum number iterations
- $x^{i.iter}$: the position of crow *i* at (iteration) *iter* where *i= 1,2, …,N*; *iter* = 1,2,…, $iter_{max}$

$x^{i.iter} = [x_1^{i,\,iter} \;\; x_2^{i,\,iter} \ldots\ldots x_d^{i\,,iter}]$ is the vector of search space.

- $m^{i;iter}$: is the best hiding food place obtained by crow *i* at iteration *iter*.

Actually, the crow keeps its best position experience and moves in the environment to find out better food sources (hiding places).

Assume crow *j* wishes to get to its hiding location, $m^{i;\,iter}$, during iteration *iter*. Crow *i* decides to follow crow *j* to the hiding place of crow *j* in this iteration.

State 1: Crow *j* is unaware of the facts that crow *i* is following it. As a result, crow *i* will find his way to crow *j*'s hiding place. In this case, the new position of crow *i* is calculated as follows:

$$x^{i,iter+1} = x^{i,iter} + r_i * fl^{i,iter} * \left(m^{j,iter} - x^{i,iter}\right) \quad (8)$$

Where r_i is a random number with uniform distribution between 0 and 1 and $fl^{i,iter}$ denotes the flight length of crow *i* at iteration *iter*.

State 2: Crow *j* is aware that crow *i* is chasing it in state 2. As a result, crow *j* will fool crow *i* by travelling to a new position in order to keep its cache safe from theft. The following are the general descriptions of states 1 and 2:

$$x^{i,iter+1} = \begin{cases} x^{i,iter} + r_i * fl^{i,iter} * \left(m^{j,iter} - x^{i,iter}\right), r_j \geq AP^{i,iter} \\ a\ random\ position \qquad otherwise \end{cases} \quad (9)$$

Where r_j is a random number with uniform distribution between 0 and 1 and $AP^{i,iter}$ denotes the awareness probability of crow *j* at iteration *iter*.

Fig. 2 shows the schematic of this state and the effect of *fl* on the search capability. Small values of *fl* leads to local search (at the vicinity of $x^{i,iter}$) and large values results in global search (far from $x^{i,iter}$). If *fl* is less than 1, the next position of the crow *i* is on the dash line between $x^{i,iter}$ and $m^{j,iter}$, as shown in Fig. 2(a). If the value of *fl* is more than 1, as shown in Fig. 2(b), the next position of crow *i* is on the dash line, which may be greater than m. $m^{j,iter}$ [32].

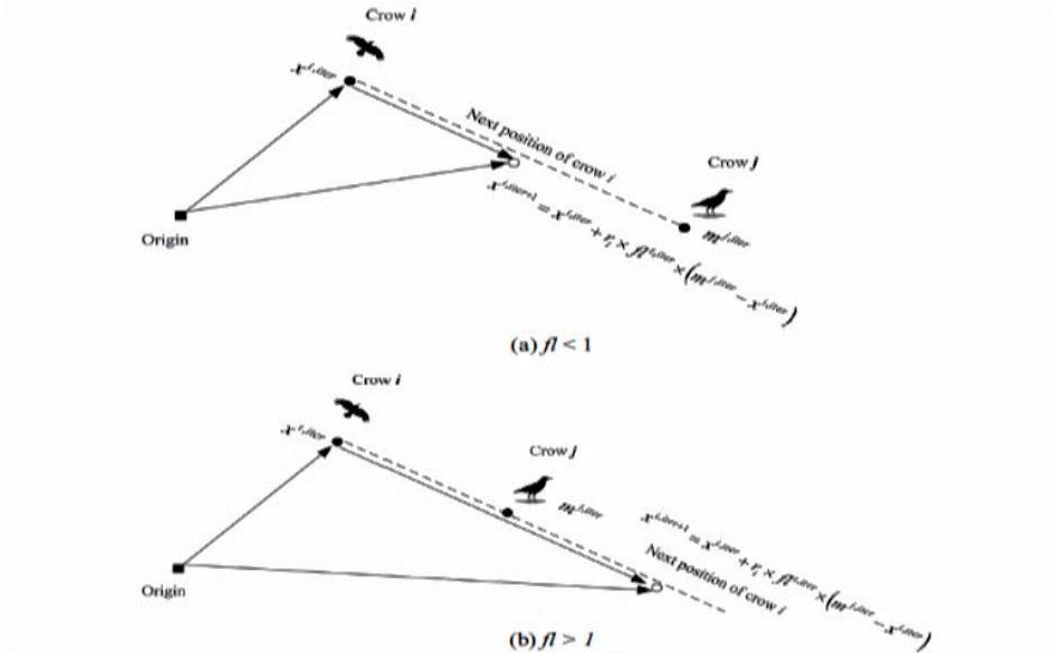


Figure 2. Effect of *fl* on the search capability

Pseudo code of Crow Search Algorithm

- *Randomly initialize the position of a flock of N crows in the search space*
- *Evaluate the position of the crows*
- *Initialize the memory of each crow*
- *While iter < $iter_{max}$*

For i = 1: N (all N crows of the flock)

Randomly choose one of the crows to follow

Define an awareness probability

if $r_j \geq AP^{j,iter}$

$\boldsymbol{x^{i,iter+1} = x^{i,iter} + r_i \times fl^{i,iter} \times (m^{j,iter} - x^{i,iter})}$

else

$\boldsymbol{x^{i,iter+1}}$ ***= a random position of sear space***

end if

end for

- *Check the feasibility of new positions*
- *Evaluate the new position of the crows*
- *Update the memory of crows*

end while

2.3 Solution Representation

When CSA is applied for solving QAP, each dimension represents a typical location. In addition, the crow $X_i^k = [x_{i1}^k \dots x_{in}^k]$ corresponds to the position values for n number of locations in the QAP problem. Each crow has a continuous set of position values. The crow does not present a sequence itself. The SPV rule is used to enable the continuous CSA to be applied to the sequencing problems. As a result, SPV is applied to the discrete QAP problem. The SPV rule is used to determine how facilities are allocated to different locations based on the position values x_{ij}^k of particle X_i^k. The solution representation of crow X_i^k for the CSA is shown in table 1 along with its corresponding sequence. The smallest position value according to the proposed SPV regulation is X_{i3}^k = -0.85, so the dimension j=3 is assigned to be the first facility π_{i1}^k =3 in the sequence π_i^k ; the second smallest position value is X_{i1}^k = -0.72 , so the dimension j=1 is assigned to be the second facility π_{i2}^k =1 in the sequence π_i^k, and so on. To put it another way, dimensions are sorted using the SPV rule, i.e., to x_{ij}^k values in order to create the sequence π_{ik}. This representation is unique in terms of finding new solutions since positions of each crow are updated at each iteration k in the CSA. As a result, each iteration k produces a new sequence.

Table1. Solution Representation of a Crow

j	1	2	3	4	5
x_{ij}^k	-0.72	2.05	-0.85	3.40	1.30
π_{ij}^k	3	1	5	2	4

According to the previous representation, the value of every cell (π_{ij}^k) represents the facility that is assigned to corresponding location. In this example, there are 5 facilities to be placed at 5 locations. For example, the first cell means that facility 3 is placed at location 1; facility 1is placed at location 2 and so on.

3. Proposed Algorithm

QAP is solved using the Improved Crow Search Algorithm (ICSA). It has been demonstrated to be highly competitive and perform well. The suggested algorithm (ICSA) and its essential steps will be briefly described in this section:

Step 1: Initialize Parameters

In this step, the parameters of the ICSA algorithm are set. The number of dimensions (*d*), the number of crows (*n*), the awareness probability (AP), the flight length (*fl*), and the maximum number of iterations ($iter_{max}$) are included.

Step 2: Initialize Position and Memory of Crows

Initialize the n crows at random locations in the search space. Each crow represents a potential solution to the problem, with d denoting the number of decision variables:

$$Crows = \begin{bmatrix} x_1^1 & x_1^2 & \cdots & x_1^d \\ \vdots & & \ddots & \vdots \\ x_n^1 & x_n^2 & \cdots & x_n^d \end{bmatrix} \qquad (10)$$

The memory of each crow has been initialized. Because the crows have no prior experience, it's assumed that they've buried their food in their initial positions as follows:

$$Memory = \begin{bmatrix} m_1^1\, m_1^2 & \cdots & m_1^d \\ \vdots & \ddots & \vdots \\ m_n^1\, m_n^2 & \cdots & m_n^d \end{bmatrix} \quad (11)$$

Step 3: Evaluate Fitness (Objective) Function

By inserting the decision variable values into the objective function for each crow, the quality of each crow position is calculated.

Step 4: Generate New Position

Generating a new position for i^{th} crow at iteration *iter*, $crow_j$ is randomly selected, so that crow *i* follows crow *j* to discover its foods hidden place. This process is repeated for all of the crows, and the new position of each crow is given by Eq.

$$x^{i,iter+1} = \begin{cases} x^{i,iter} + r_i \times fl^{i,iter} \times \left(m^{j,iter} - x^{i,iter}\right), r_j \geq AP^{i,iter} \\ x^{i,iter} + \left(m^{j,iter} - (r_i \times x^{i,iter})\right) \quad otherwise \end{cases} \quad (12)$$

Step 5: Check the Feasibility of New Positions

Based on the awareness probability (AP), the feasibility of each crow's new position is determined. Assume a crow new position is possible. The crow then changes its position and moves to the newly generated position. Otherwise, the crow takes the global best position (x_{best}).

Step 6: Evaluate the Fitness Function of New Positions
Each crow's new position's fitness function value is calculated.

Step 7: Execute Swap Operation.
Swap is to choose two different positions of an operation permutation sequence randomly and swap them.

Step 8: Memory Update
The memory of the i^{th} crow is updated as follows after the fitness value for the new position is computed:

$$m^{i,iter+1} = \begin{cases} x^{i,iter+1}, \ if\ f\left(x^{i,iter+1}\right) is\ better\ than\ f\left(m^{i,iter+1}\right) \\ m^{i,iter} \quad otherwise \end{cases} \quad (13)$$

Where f ($m^{(i,\ iter+1)}$) denotes the fitness value and $m^{i,\ iter}$ is the j^{th} crow's hidden location at iteration *iter*. If a crow's new position is better than the memorized position's fitness function value, the crow updates its memory with the new position using the prior update equation.

Step 9: Check the Termination Criterion

If the current iteration is less than the maximum number of iterations ($iter_{max}$), repeat Steps 4–7; otherwise, the optimization process is ended.

Pseudo code of improved crow search algorithm

Initialize parameters;

- *Randomly initialize the position of the n crows in the search space;*
- *Compute the fitness value of the crows:*
- *While iter < Iter $_{max}$ do*

for i = 1:n (all n crows of the flock)

Randomly choose one of the crows to follow;

Define the awareness probability (AP)

if $r_j \geq AP^{j,\,iter}$

$$x^{i,iter+1} = x^{i,iter} + r_i \times fl^{i,iter} \times \left(m^{i,iter} - x^{i,iter}\right)$$

else

$$x^{i,iter+1} = x^{i,iter} + \left(m^{i,iter} - (r_i \times x^{i,iter})\right)$$

end if

end for

- *Check the feasibility of new positions*
- *Evaluate the new position of the crows*
- *Execute Swap operator*
- *Update the memory of crows*

end while

4. Computational Results

In the experiment, ten instances of the benchmark in QAP data set with various sizes are used to test the behavior of the convergence, and accuracy of the proposed method. The results compared with the other methods in the literature show that the proposed scheme increases more the convergence and the accuracy than the Discrete Bat Algorithm (DBA). The ICSA is tested on problems (tai12a, tai15a, tai17a, tai20a, tai25a, tai30a, tai35a, tai40a, tai50a, tai60a) [31]. These problems are available on the OR-Library web site [33]. ICSA is tested with 30 independent runs; the number of individuals (crow) in the population is fixed to 50.

Maximum iterations for the ICSA are 500 for each run. *AP* is set 0.5 while *fl* is selected to be 1.5. The proposed algorithm is compared with the DBA [31]. The computational results of tai test problems are shown in Table 2. It includes the size of each problem, Best Known Solution (BKS), best solution and Relative Percentage Error (RPE) for each method.

$$RPE = \frac{best - BKS}{BKS} * 100 \qquad (14)$$

The results show that the ICSA is much better than the DBA according to efficiency and time consuming.

The comparison is based on the results for the problems of Taillard quadratic assignment problems as shown in Table 2 and Fig.3. It can be observed that the two algorithms generated the best known solution for the tai12a, tai15a, tai17a, tai20a, tai40a problems, but the solutions in the remaining five problems (tai25a, tai30a, tai35a, tai50a, and tai60a) explain that ICSA gives better results than DBA.

In Fig. 4, the comparison between ICSA and DBA is based on RPE. The two algorithms are able to find the best known solutions for the problems (tai12a, tai15a, tai17a, tai20a, tai40a) of ICSA and DBA, and therefore RPE is zero. The RPE of ICSA is less than RPE of DBA for the problems (tai25a, tai30a, tai35a, tai50a, and tai60a).

Table2. The Comparison between DBA and ICSA

Problem No.	Problem	Size (n*n)	BKS	DBA		ICSA	
				Best	RPE	Best	RPE
1	tai12a	12	224,416	224,416	0	224,416	0
2	tai15a	15	388,214	388,214	0	388,214	0
3	tai17a	17	491,812	491,812	0	491,812	0
4	tai20a	20	703,482	703,482	0	703,482	0
5	tai25a	25	1,167,256	1,172,754	0.471019	1,169,842	0.221545
6	tai30a	30	1,818,146	1,831,272	0.721944	1,825,614	0.410748
7	tai35a	35	2,422,002	2,438,440	0.678695	2,428,120	0.252601
8	tai40a	40	3,139,370	3,139,370	0	3,139,370	0
9	tai50a	50	4,938,796	5,042,654	2.102901	5,040,012	2.049406
10	tai60a	60	7,205,962	7,387,482	2.519025	7,325,135	1.653811

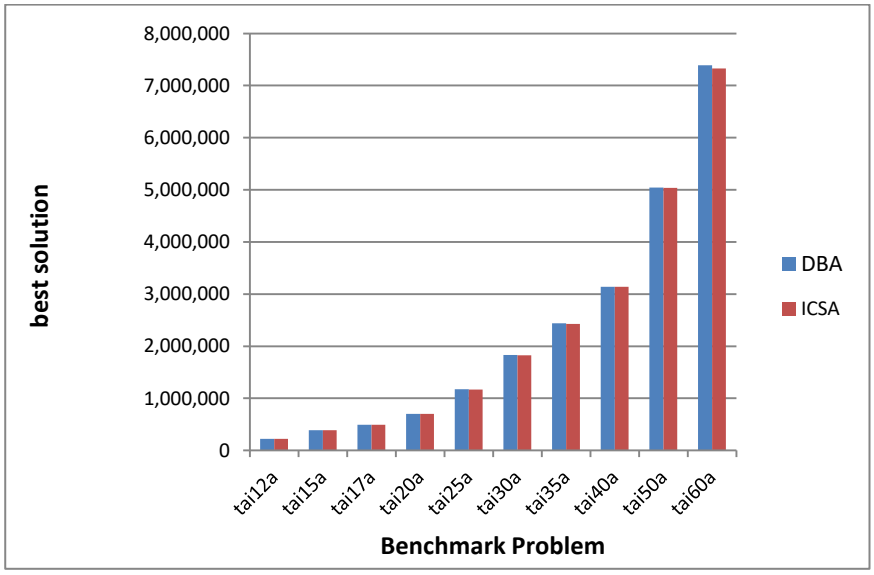


Figure 3. Comparison between DBA and ICSA Using Best Solution

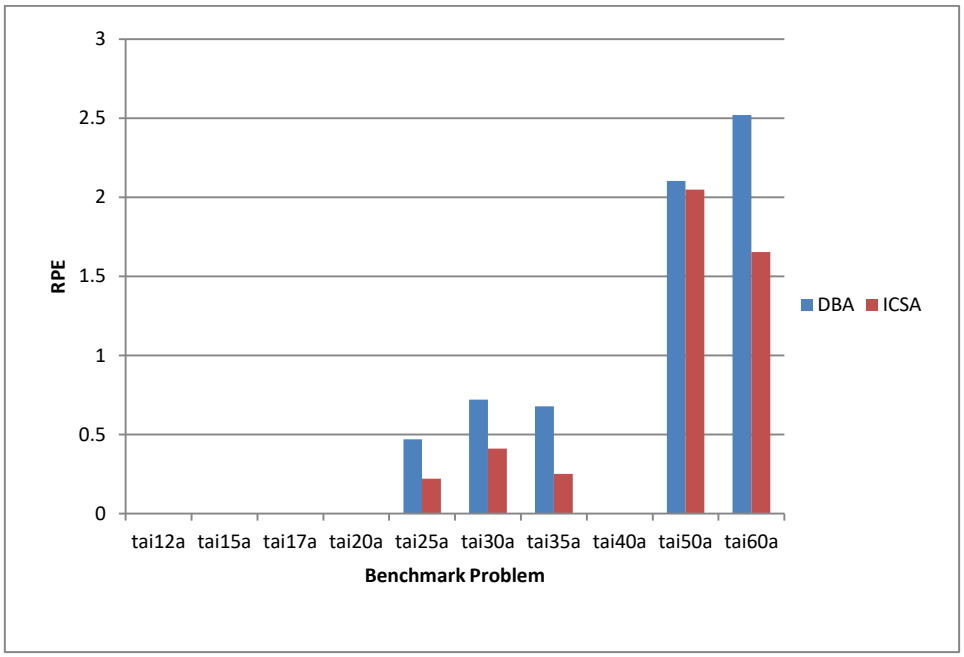


Figure 4. Comparison between DBA and ICSA Using RPE

5. Conclusion

In this paper, a meta-heuristic algorithm, CSA is introduced to solve the QAP. More specifically, an improved version of CSA is named ICSA for QAP. One of the main contributions of this paper is to introduce some new modifications in the original version of CSA. These modifications regard position representation and its update equation to enhance the search strategy and a swap operator is used to enhance global search capability. A collection of benchmark instances from the QAPLIB library are used to test the proposed algorithm. The performance of ICSA is evaluated by comparing the results with DBA results for a number of benchmark instances. The proposed enhanced method is clearly very effective and efficient. It can find optima for a set of test instances, and running time is less than other algorithm.

References

[1] Sed M. Quadratic Assignment Problem. Contributions to Management Science, 2009, 1007/978-3-7908-2151-2.

[2] Dennis H., The Quadratic Assignment Problem: A Note, author profiles for this publication, 1972; https://www.researchgate.net/publication/4813092.

[3] Armour, G.C.; Buffa, E.S. A heuristic algorithm and simulation approach to relative location of facilities. Manag. Sci. 1963, 9, 294–304.

[4] Nugent, C.E.; Vollmann, T.E.; Ruml, J. An experimental comparison of techniques for the assignment of facilities to locations. J. Oper. Res. 1968, 16, 150–173.

[5] Angel, E.; Zissimopoulos, V. On the quality of local search for the quadratic assignment problem. Discret. Appl. Math. 1998, 82, 15–25.

[6] Pardalos, P.M.; Murthy, K.A.; Harrison, T.P. A computational comparison of local search heuristics for solving quadratic assignment problems. Informatica 1993, 4, 172–187.

[7] Burkard, R.E.; Rendl, F. A thermodynamically motivated simulation procedure for combinatorial optimization problems. Eur. J. Oper. Res. 1984, 17, 169–174.

[8] Bölte, A.; Thonemann, U.W. Optimizing simulated annealing schedules with genetic programming. Eur. J. Oper. Res. 1996, 92, 402–416.

[9] Paul, G. An efficient implementation of the simulated annealing heuristic for the quadratic assignment problem. arXiv 2011, arXiv:1111.1353.

[10] Taillard, E.D. Robust taboo search for the QAP. Parallel. Comput. 1991, 17, 443–455.

[11] Battiti, R.; Tecchiolli, G. The reactive tabu search. ORSA J. Comput. 1994, 6, 126–140.

[12] Misevicius, A. A tabu search algorithm for the quadratic assignment problem. Comput. Optim. Appl. 2005, 30, 95–111.

[13] Fescioglu-Unver, N.; Kokar, M.M. Self controlling tabu search algorithm for the quadratic assignment problem. Comput. Ind. Eng.2011, 60, 310–319.

[14] Abdelkafi, O.; Derbel, B.; Liefooghe, A. A parallel tabu search for the large-scale quadratic assignment problem. In Proceedings of the IEEE Congress on Evolutionary Computation, IEEE CEC 2019, Wellington, New Zealand, 10–13 June 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 3070–3077.

[15] Czapi´ nski, M. An effective parallel multistart tabu search for quadratic assignment problem on CUDA platform. J. Parallel Distrib. Comput. 2013, 73, 1461–1468.

[16] Lim, M.H.; Yuan, Y.; Omatu, S. Efficient genetic algorithms using simple genes exchange local search policy for the quadratic assignment problem. Comput. Optim. Appl. 2000, 15, 249–268.

[17] Ahuja, R.K.; Orlin, J.B.; Tiwari, A. A greedy genetic algorithm for the quadratic assignment problem. Comput. Oper. Res. 2000, 27, 917–934.

[18] Drezner, Z. A new genetic algorithm for the quadratic assignment problem. INFORMS J. Comput. 2003, 15, 320–330.

[19] Özçetin, E.; Öztürk, G. A hybrid genetic algorithm for the quadratic assignment problem on graphics processing units. Anadolu Univ. J. Sci. Technol. A Appl. Sci. Eng. 2016, 17, 167–180.

[20] Misevicius, A. An improved hybrid genetic algorithm: New results for the quadratic assignment problem. Knowl.-Based Syst. 2004, 17, 65–73.

[21] Ahmed, Z.H. A hybrid algorithm combining lexisearch and genetic algorithms for the quadratic assignment problem. Cogent Eng. 2018, 5, 1423743.

[22] Baldé, M.A.M.T.; Gueye, S.; Ndiaye, B.M. A greedy evolutionary hybridization algorithm for the optimal network and quadratic assignment problem. Oper. Res. 2020, 1–28.

[23] Chmiel, W. Evolutionary algorithm using conditional expectation value for quadratic assignment problem. Swarm Evol. Comput. 2019, 46, 1–27.

[24] Hafiz, F.; Abdennour, A. Particle swarm algorithm variants for the quadratic assignment problems—A probabilistic learning approach. Expert Syst. Appl. 2016, 44, 413–431.

[25] Szwed, P.; Chmiel, W.; Kadłuczka, P. OpenCL implementation of PSO algorithm for the quadratic assignment problem. In Artificial Intelligence and Soft Computing, ICAISC 2015, Lecture Notes in Computer Science; Rutkowski, L., Korytkowski, M., Scherer, R., Tadeusiewicz, R., Zadeh, L., Zurada, J., Eds.; Springer: Cham, Switzerland, 2015; Volume 9120, pp. 223–234.

[26] Samanta, S.; Philip, D.; Chakraborty, S. A quick convergent artificial bee colony algorithm for solving quadratic assignment problems. Comput. Ind. Eng. 2019, 137, 106070.

[27] Abdel-Baset, M.; Wu, H.; Zhou, Y.; Abdel-Fatah, L. Elite opposition-flower pollination algorithm for quadratic assignment problem. J. Intell. Fuzzy Syst. 2017, 33, 901–911.

[28] Dokeroglu, T.; Sevinc, E.; Cosar, A. Artificial bee colony optimization for the quadratic assignment problem. Appl. Soft Comput. 2019, 76, 595–606.

[29] Dokeroglu, T. Hybrid teaching–learning-based optimization algorithms for the quadratic assignment problem. Comput. Ind. Eng. 2015, 85, 86–101.

[30] Duman, E.; Uysal, M.; Alkaya, A.F. Migrating birds optimization: A new metaheuristic approach and its performance on quadratic assignment problem. Inf. Sci. 2012, 217, 65–77.

[31] Mohammed E. R., Yassine S., Mohammed B., Incorporating a modified uniform crossover and 2-exchange neighborhood mechanism in a discrete bat algorithm to solve the quadratic assignment problem, 2017; Egyptian Informatics Journal, Elsevier.

[32] Alireza A., A novel meta-heuristic method for solving constrained engineering optimization problems: Crow search algorithm, http://dx.doi.org/10.1016/j.compstruc.2016.03.001, 2016; 0045-7949/ Elsevier Ltd.

[33] Beasley, J. E., OR-Library: Distributing test problems by electronic mail. Journal of the Operational Research Society, 1990; pp.1069–1072.