Indonesian Journal of Electrical Engineering and Computer Science
Vol. 14, No. 1, April 2019, pp. 143~154
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v14.i1.pp143-154

Improved hunting search algorithm for the quadratic assignment problem

Amine Agharghor, Mohammed Essaid Riffi, Fayçal Chebihi
Laroseri Laboratory, Department of Computer Sciences, Faculty of Sciences, University of Chouaib Doukkali, El Jadida, Morocco

Article Info

Article history:

Received May 1, 2018
Revised Aug 10, 2018
Accepted Dec 25, 2018

Keywords:

Combinatorial optimization problem
Hunting search algorithm
Metaheuristic
QAPLib
Quadratic assignment problem

ABSTRACT

Nowadays, the metaheuristics are the most studied methods used to solve the hard optimization problems. Hunting Search algorithm is a metaheuristic inspired by the method of group hunting of predatory animals like wolves. Created for solving continuous optimization problems, recently, it is adapted and evaluated to solve hard combinatorial optimization problems. This paper proposes an improved hunting search algorithm to solve the quadratic assignment problem. No local search method is used. To evaluate the performances of this work, the improved Hunting Search is checked on a set of 36 instances of QAPLib and it outperforms the results obtained by the well-known metaheuristics.

Corresponding Author:

Amine Agharghor,
Laroseri Laboratory, Department of Computer Sciences,
Faculty of Sciences, University of Chouaib,
Doukkali, El Jadida, Morocco.
Email: amine.agharghor@gmail.com

1. INTRODUCTION

Metaheuristics are the best algorithms used to solve the NP-Hard optimization problems for which there is no exact effective known method. Sometimes, they can be the only possible methods to find a good solution to these hard problems. The most important advantage of using a metaheuristic is that it uses a high abstraction level. Therefore, it can be applied to wide different problems. New metaheuristics are proposed recently such as the hunting search [1], the chicken swarm optimization [2], [3] and the golden ball algorithm [4].

Hunting Search (HuS) is a developed metaheuristic for solving continuous optimization problems. It starts to be adapted to solve combinatorial optimization problems such as The Traveling Salesman Problem [5], [6] and the no-wait flow shop scheduling [7]. It is also proposed as a first adaptation for the Quadratic Assignment Problem [8]. HuS is a method inspired by group hunting of some animals such as wolves that organize their position to surround the prey. Each of them is relative to the position of others and especially in relation to the position of their leader.

Quadratic Assignment Problem (QAP) [9] is the well-known discrete optimization problem of the category of the facilities location problems NP-hard. The problem is to assign a set of facilities to a set of locations in order to minimize the total cost of assignments. The QAP has several applications in combinatorial optimization problems such as Backboard Wiring [10] and scheduling [11]. In order to solve the QAP, several metaheuristics have been proposed [12]. The present paper proposes an Improved Hunting Search algorithm (IHuS) adapted as a combinatorial optimization method to solve the QAP in a minimum CPU run time.

This paper is divided into six main sections. The first one is an introduction of the given method and the benchmark problem. The second section presents the metaheuristic HuS. The third section provides a detailed description of the QAP. The fourth section presents the proposed adaptation and improvement of HuS for the QAP. Numerical results obtained by the use of IHuS on the QAPLib instances [13] are demonstrated in the five section, and the last section concludes the whole work.

2. HUNTING SEARCH ALGORITHM

Hunting Search algorithm is an evolutionary method inspired by cooperative hunting of some carnivores that hunt bigger and faster preys than themselves with fewer energy. It is a population-based stochastic metaheuristic. The population is the hunting group that contains a set of solutions of the studied problem; each hunter represents a solution. The leader represents the best solution defined by an objective function. A hunter is characterized by its position that defines the distance between it and the other hunters.

The hunting process in nature represents the search of optimum in the algorithm. The movements of the hunters to encircle the prey represents the operations of improvement of the initial solutions. Hunters do three main movements. Two movements of intensification operations. These are the movement toward the leader and the movement toward the other hunters. The third one is a diversification operation the reorganization of the hunters when they become very close to each other. HuS algorithm is presented in Figure 1:

- **HG** (**H**unting **G**roup) is the population of hunters used for the search of the optimum.
- **NE** (**N**umber of **E**pochs) is the number of loops to make for the search.
- **IE** (**I**teration per **E**poch) is the number of times per epoch where hunters make a move.

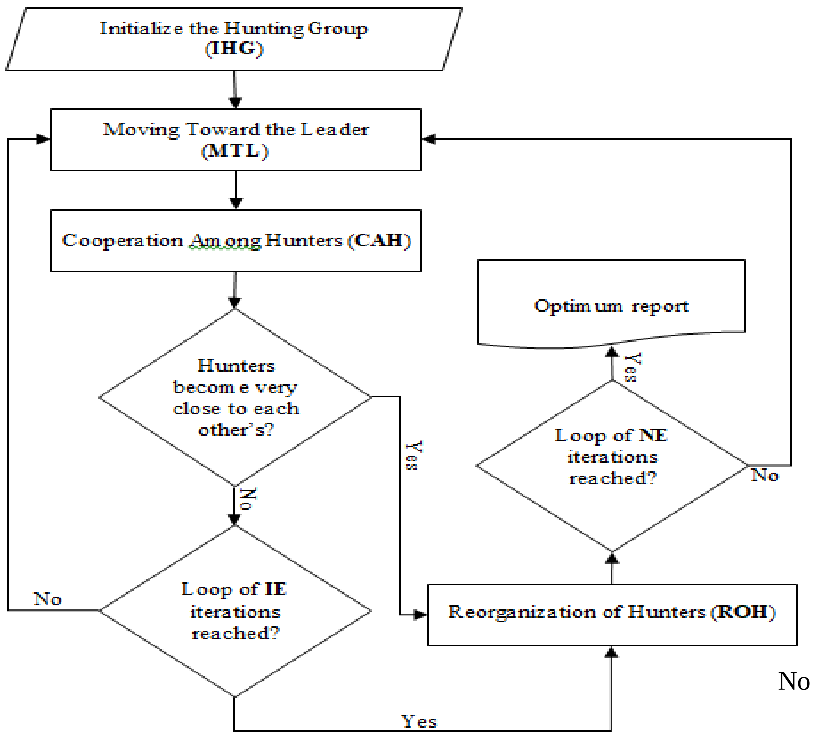
Initialize the Hunting Group (IHG)
Moving Toward the Leader (MTL)
Cooperation Among Hunters (CAH)
Hunters become very close to each other's?
Yes
No
Loop of IE iterations reached?
No
Yes
Reorganization of Hunters (ROH)
Loop of NE iterations reached?
No
Yes
Optimum report
No

Figure 1. Flowchart of hunting search algorithm

The main operations of HuS are:

a) Initialize the Hunting Group: generate a set of random solutions for a problem; each hunter represents a solution.
b) Moving Toward the Leader: hunters move toward the leader, which makes them stay closer to the leader until finding a better solution and becoming the new leader.
c) Cooperation among Hunters: each hunter moves toward the other hunters or just makes a random move.

d) Reorganization of hunters: at some point, hunters become very close to each other, which means that they represent almost the same solution. In this case, they have to be regenerated, except the leader.

3. QUADRATIC ASSIGNMENT PROBLEM

3.1. Definition

QAP is a combinatorial optimization problem of the class NP-Hard whose computational complexity increases exponentially by increasing the number of facilities. No exact method can solve the problem when the number of facilities is upper then twenty.

Given a set of facilities to assign to a set of locations, there is a required flow between every two facilities and a required distance between every two locations. The problem is to find the best assignment of the facilities to the locations to have the minimum total cost of flows and distances.

Figure 2 is an example of assignment of three facilities to three locations. **D1** and **D2** are the distances between the locations, and **F1** and **F2** are the flows between the facilities.

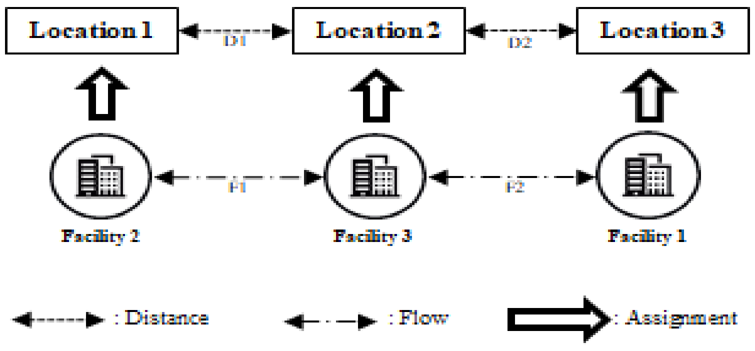

Figure 2. Assignment of three facilities

The optimal solution is defined by an objective function from a discrete subset of the feasible solutions.

Let *E* be the set of the feasible solutions; *S* is a subset of *E*; and $g: S \to \mathbb{R}$ is the objective function. The problem is to find:

$$\min\{g(s) : s \in S\} \tag{1}$$

Where *s* is a solution from *S*. It is a vector of $\mathbb{N}$, which contains indexes of facilities assigned respectively to locations 1, 2,…,n. n is the number of the locations.

The objective function gives the cost of the assignment, defined as follows:

$$g(s) = \sum_{i=1}^{n}\sum_{j=1}^{n} f_{s(i)s(j)} \times d_{ij} \tag{2}$$

Where $(\boldsymbol{f_{ij}}) \in \mathbb{R}$ and $(\boldsymbol{d_{ij}}) \in \mathbb{R}$ are two matrices. $(\boldsymbol{f_{ij}})$ is the square matrix that represents the required flow between facility *i* and *j*. $(\boldsymbol{d_{ij}})$ is the square matrix that represents the distance between location *i* and *j*.

s ∈ S, s(*i*) is the location to which facility *i* is assigned and *n* is the dimension of the two matrices.

4. IMPROVED HUNTING SEARCH ALGORITHM

HuS as a metaheuristics uses intensification and diversification operations. The move towards the leader and the cooperation among hunters are the intensification operations; and the reorganization of hunters is the diversification operation. HuS uses also a population of individuals (the hunters) in the search space that undergoes mutation operations. It is an evolutionary algorithm.

4.1. Adaptation

4.1.1 Initialize the hunting group

To represent a solution of QAP in a program, an array of integer is used, where the array indexes refer to the locations respectively *1, 2, ..., n* (*n* is the number of locations) and the array contents refer to facilities assigned to each location. Figure 3 is an example of a QAP solution that represents an assignment of five facilities to five locations; the solution is represented by a hunter.

In this first step, a set of random solutions called the Hunting Group (HG) is created. Each solution is represented by a hunter and the size of the set of solutions is called the Hunting Group Size (HGS). Figure 4 is an example of a HG of HGS=3.

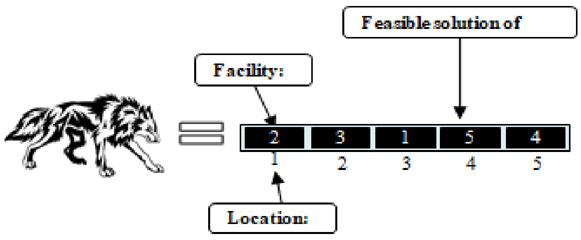

Figure 3. An example of an assignment facilities solution

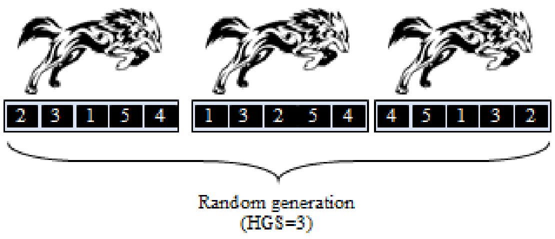

Figure 4. An example of a HG initialization

4.1.2 The movements of a hunter

The only possible representation of the movement of a given hunter in the program is the permutation of two of its array contents. The algorithm is used to define the strategy and the sizes of permutations to do. Therefore, a movement of a hunter is equal to a permutation as it is shown in (3).

$$1\, movement = 1\, permutation \tag{3}$$

The movement of a hunter H_i toward another hunter H_j is also a permutation in a given array index k. One search for the location of the content array in H_i that is similar to the content of array H_j in the array index k and permute them in H_i as it shown in (4). H_i is the updated hunter and H_j is from which the update is inspired.

$$1\, movement(H_i, H_j, k) = 1\, permutation(H_i, H_j, k) \tag{4}$$

Figure 5 is an example of movement of the hunter **H_1** toward the hunter **H_3** in the array index **k=2**.

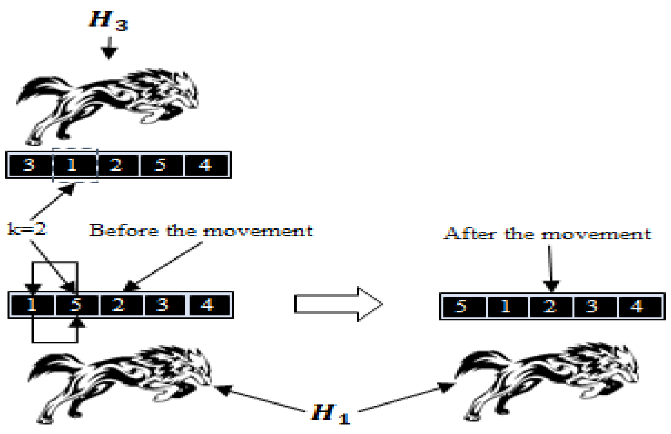

Figure 5. An example of a movement toward a hunter

4.1.3 Moving towards the leader

After the initialization of the HG starts the loop of the NE iterations. In this loop starts the second loop of IE iterations, where comes the move towards the leader (MTL). In this step, each hunter moves toward the leader by copying a part from the best solution; the size of the movement towards the leader (SML) which is the size of the copied part is calculated as follows:

$$SML = rand \times MML \times D(H_L, H_i) \tag{5}$$

Where:

- Rand is a uniform random number that varies between 0 and 1.
- MML (Maximum Movement toward the Leader) is a number between 0 and 1 representing the maximum closer rate of a hunter to the leader.
- Function D refers to the distance between two hunters. It is the number of the different assignments in the two represented solutions. HL is the leader and Hi is the hunter number i.

Each movement toward a leader (MTL) is characterized by a start array index that is chosen randomly and a size of movements toward the leader that is SML.

$$1\ MTL = SML \times permutation \tag{6}$$

Figure 6 is an example of a movement of the hunter number one towards the leader, the SML=2.

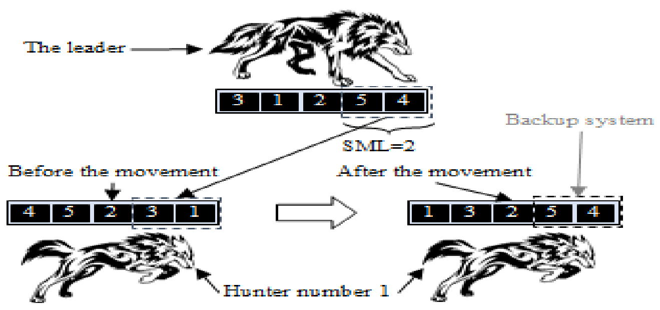

Figure 6. An example of a movement towards the leader

4.1.4 Cooperation among hunters

Always in the loop of IE iterations, after the move towards the leader comes the cooperation among hunters, where each hunter moves towards the other hunters by copying a part from their solution with the probability HGCR, or by changing their position relatively to themselves with the probability (1-HGCR). HGCR (Hunting Group Consideration Rate) is a number between 0 and 1. Figure 7 is an example of the movement of the hunter number 1 towards three different hunters (the case of the probability HGCR). And Figure 8 is an example of changing the position of a hunter relatively to itself (the case of the probability 1-HGCR).

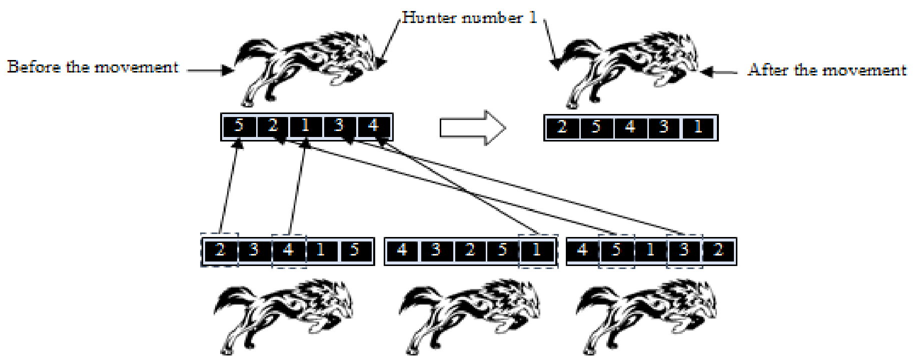

Figure 7. An example of a movement towards others hunters (the case of HGCR)

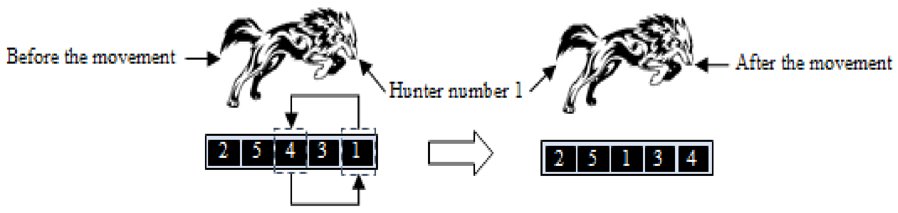

Figure 8. An example of changing the position (the case of 1-HGCR)

4.1.5 Reorganization

The intensification operations reduce the distances among the hunters. So, the solutions presented by these hunters become almost similar, which can lead to a blockage in a local optimum. Then, a diversification operation is needed: the reorganization hunters. Figure 9 is an example of reorganization of a population of hunters of HGS=3.

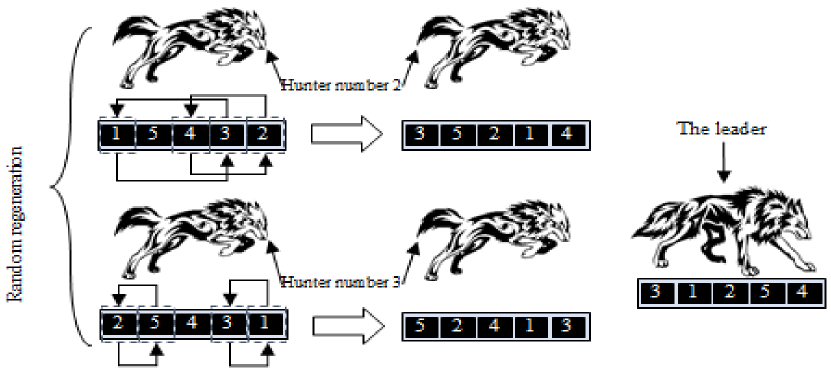

Figure 9. An example of reorganization hunters

4.1.6 Strategies

a) Backup system

At the end of every intensification operation, a backup system is applied and only the good new solutions are taken. If the movement towards the leader or towards other hunters gives worse solution, one returns to the previous position. This strategy helps a rapid convergence towards local optimum.

b) End criteria

The end of the search process is determined by the number of epochs *NE* or by its half in the case of a stationary optimum.

4.2. Improvements

4.2.1 One by one backup system

The first improvement is applied at the strategy of the backup system. Therefore, a backup is made for every change during the operation of intensification and not until the end. Figure 10 is an example of the improved backup system during the movement towards leader.

4.2.2 Leader moving towards hunters

In the initial HuS algorithm, the leader does not move along the search. The new proposed improvement is to move the leader towards the other hunters (LMTH) while using the backup system for preserving its quality of the best-found position. Figure 11 is an example of a leader moving towards three hunters using the improved backup system.

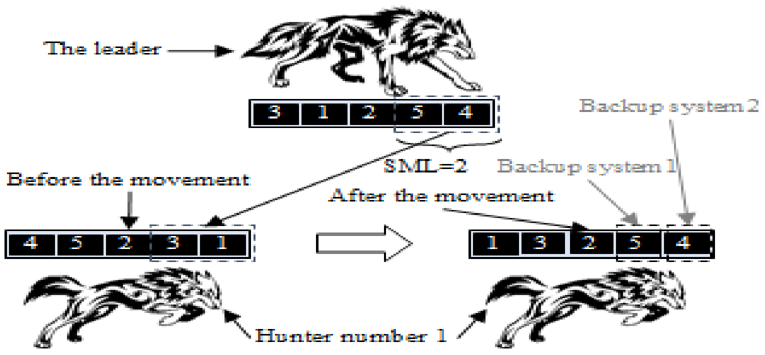

Figure 10. An example of an improved backup system

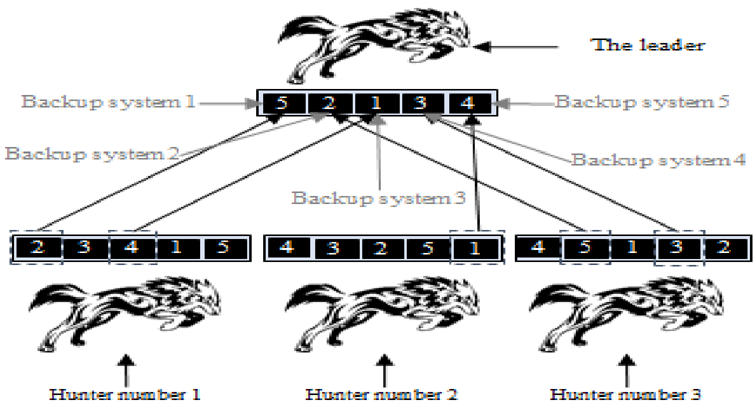

Figure 11. An example of a movement of the leader towards the other hunters

4.2.3 Dynamic parameters

The final proposed improvement is at the level of changing parameters dynamically during the search of optimum. This improvement helps better control the intensification and the diversification operations. Parameters are classify in three categories: parameters of static value, parameters of dynamic value and parameters of hybrid value. The parameters of static value are *HGS*, *HGCR* and *NE*; they have the same value from the beginning of the search until the end. The parameter of dynamic value is *MML*; it changes its value during the search. And the parameter of hybrid value is *EPS*; it changes its value under certain conditions.

The proposed definitions for the static parameters are as follows:

$$HGS = N \tag{7}$$

$$NE = 100 \tag{8}$$

Where N is the size of the evaluated QAPLib instance.
The proposed definitions for the dynamic parameters are as follows:

$$RLW = \frac{(EN - TN)}{EN} \tag{9}$$

$$MML = 0.2 + 0.3 \times RLW \tag{10}$$

$$EPS = \min\{g(s_W) - g(s_L) : s \in S\} \tag{11}$$

$$EPS = \begin{cases} EPS + 0.01 \times EPS, & RLW > 0.5 \\ EPS - 0.01 \times EPS, & RLW < 0.3 \end{cases} \quad (12)$$

$$IE = \begin{cases} IE + 5 \times (EN - TN), & RLW > 0.5 \\ IE - 5 \times (EN - TN), & RLW < 0.3 \end{cases} \quad (13)$$

Where:

1. $\boldsymbol{RLW}$ (**R**ate of the distance between the **L**eader and the **W**orst hunter).
2. ***TN*** (**T**rapped **N**umber) is the number of epoch that the distance between the leader and the worst hunter is lower than ***EPS***.
3. ***EN*** (**E**poch **N**umber) is the number of epochs reached during the search.
4. ***EPS*** (**Eps**ilon) is the minimum distance between the leader and the worst hunter.
5. ***s*** is a solution from the subset of solutions **S.**
6. $\boldsymbol{g(s_L)}$ is the value of the objective function of the best solution represented by the leader during the search of the optimum.
7. $\boldsymbol{g(s_W)}$ is the value of the objective function of the worst solution during the search of the optimum.

According to **(10)** the value of ***MML*** is decreasing while the value of ***TN*** is increasing and vice versa. The minimum value of ***MML*** is ***0.2*** and the maximum value is ***0.5***.

The value of ***EPS*** in **(11)** is determined as the minimum distance during the search, then, in **(12)** is an improvement of ***EPS***.

According to **(13)** if the ***RLW*** is superior to 0.5, which means that the distance between the leader and the worst hunter is 50% superior to ***EPS***, the value of ***IE*** is increasing to intnsify the search. And if the ***RLW*** is inferior to 0.3, which means that the distance between the leader and the worst hunter is 30% infirior to ***EPS***, the value of ***IE*** is decreasing to reduce the intensification during the search. The minimum value of ***IE*** is ***30*** and the maximum value is ***100***.

5. EXPERIMENTAL RESULTS

5.1. Experimental results of the proposed improvements

This section presents the performance of IHuS algorithm on instances of QAPLib. The tests are performed on a computer processor Intel(R) Core(TM) i5-4300 CPU @ 1.9GHz @ 2.50 GHz and 4 GB of RAM. 20 times tested for each instance. The most important collected data from the obtained results over the 20 runs are:

1. $\boldsymbol{\delta_{best}}$: The **B**est-**F**ound **S**olutions (**BFS**).
2. $\boldsymbol{\delta_{avg}}$: The average of the **BFS**.
3. **PSD**: **P**ercentage of the **S**tandard **D**eviation.
4. **Suc.**: The percentage of **Suc**cess in getting the **B**est-**K**nown **S**olution (**BKS**).
5. $\boldsymbol{\theta_{avg}}$: The average percentage of error in getting the BFS.
6. $\boldsymbol{\Gamma_{avg}}$: The average run time of the program in getting the BFS.
7. $\boldsymbol{\Gamma_{best}}$: The best run time of the program in getting the BFS.

PSD is calculated as follow:

$$\text{PSD} = \frac{SD}{\delta_{avg}} \times 100 \quad (14)$$

$$\text{SD} = \sqrt{\frac{\sum_{i=1}^{20}\left(BFS_i - \delta_{avg}\right)^2}{20}} \quad (15)$$

Where

- SD is the Standard Deviation.
- *BSF$_i$* is the best-found solution in the test number i.

θ_{avg} is calculated as follow:

$$\theta_{avg} = \left(\frac{\delta_{avg} - BKS}{BKS}\right) \times 100 \quad (16)$$

5.1.1 One by one backup system

Table 1 shows the used parameter values for the tests of the new proposed backup system on the QAPLib instance Bur26a. Table 2 shows the obtained results compared to the old backup system (backup system for all the modified part in the solution).

Table 1. Parameter Values

HGS	MML	IE	NE
26	0.3	30	100

Table 2. One By One Backup System Compared To Full Part Backup System

Backup system	Suc. (%)	θavg (%)	Γbest (sec)	Γavg (sec)
Full part	1	0.2049	2.145	2.911
One by one	87	0.0107	0.901	24.634

According to the table, the success of finding the optimum by the new proposed backup system has increased considerably compared to the one obtained by the old backup system, exactly 86 times better. The percentage of error is reduced more than 20 times, and the best time is reduced more than 2 times. Finally, the average time is increased more than 10 times because of the additional tests.

5.1.2 Leader moving toward hunters

Table 3 shows the obtained results for the tests of the new proposed operation: the move of the leader toward hunters (LMTH) on the QAPLib instance Bur26a.

‘

Table 3. Results For The Lmth Operation

Proposed operation	Suc. (%)	θavg (%)	Γbest (sec)	Γavg (sec)
LMTH	100	0	0.954	16.669

According to the table, the success to find the optimum by the new proposed LMTH operation has reached 100%, and the average time is improved by 33%.

5.1.3 Dynamic parameters

5.1.3.1 Hunting Group Size

Figure 12 shows the obtained results from the tests of the proposed values of HGS applied to the QAPLib instance Bur26a. The proposed values of HGS are:

$$HGS = N \tag{17}$$

$$HGS = N \times 2 \tag{18}$$

$$HGS = N \div 2 \tag{19}$$

Where $\boldsymbol{N}$ is the size of the evaluated QAPLib instance. $\boldsymbol{N}(\text{Bur26a}) = \mathbf{26}$

According to Figure 12, **(18)** is better than **(19)** in all the obtained results. **(17)** is better than **(18)** in all the obtained results except for $\boldsymbol{\Gamma_{avg}}$ by less than 10%. The most important result is $\boldsymbol{\theta_{avg}}$. So, **(17)** is the better value for HGS.

5.1.3.2 Maximum Movement toward the Leader

Figure 13 shows the obtained results from the tests of the fixed maximum and minimum values of **MML** applied to the QAPLib instance Bur26a. According to Figure 13, the average error is equal to 0 and the average time is the least when the **MML** value is between 0.2 and 0.5.

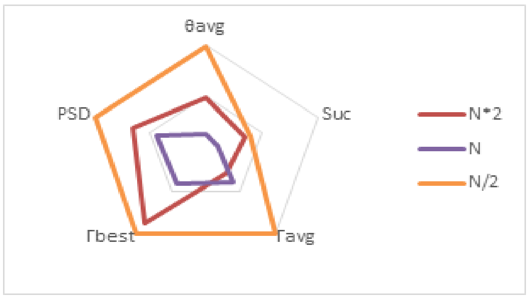

Figure 12. The results obtained by changing the value of HGS

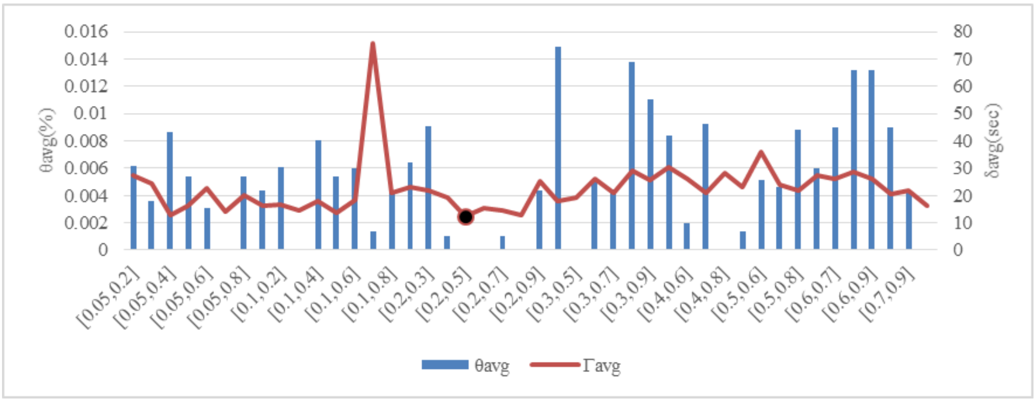

Figure 13. The results obtained by changing the maximum and minimum values of MML

5.1.3.3 Epsilon and the iteration per epoch

Figure 14 shows the obtained results from the tests of the **RLW** values that increase or decrease the values of **Eps** and **IE** applied to the QAPLib instance Bur26a.

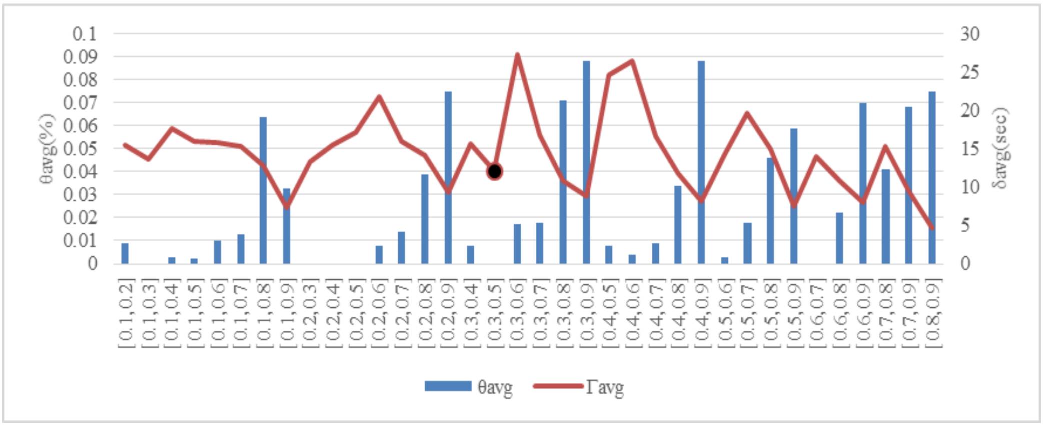

Figure 14. The results obtained by changing the Eps and the IE values

According to Figure 14, the average error is equal to 0 and the average time is the least when the **Eps** and the **IE** values are improved when the value of **RLW** is inferior to 0.3 or superior to 0.5. Table 4 shows the obtained results for the tests of the dynamic parameters compared to the static parameters on the QAPLib instance Bur26a. According to the table, the dynamic parameters have improved the best time and the average time to find the optimum.

Table 4. Results For The Dynamic Parameters

Parameters	Suc. (%)	θavg (%)	Γbest (sec)	Γavg (sec)
Static	100	0	0.455	17.720
Dynamic	100	0	0.317	14.639

5.1.4. IHuS applied to QAPLib instances

Table 5 shows the obtained results by applying IHuS to 36 instances of QAPLib. Among 36 instances of QAPLib evaluated in the Table 5, the proposed IHuS has solved 20 instances with no error, 13 instances with error and has not solved 3 instances.

Table 5. Numerical Results Obtained By Ihus Applied To Some Qap Instances Of Qaplib

N°	Instance	BKS	δavg	θavg (%)	Γavg (sec)	PSD	Suc. (%)	Γbest (sec)
1	bur26a	5426670	5426670	0	14.640	0	100	0.317
2	bur26b	3817852	3817852	0	13.547	0	100	2.824
3	bur26c	5426795	5426795	0	3.218	0	100	0.688
4	bur26d	3821225	3821225	0	7.951	0	100	1.579
5	bur26e	5386879	5386879	0	4.221	0	100	0.653
6	bur26f	3782044	3782044	0	3.159	0	100	0.297
7	bur26g	10117172	10117172	0	15.829	0	100	2.031
8	bur26h	7098658	7098658	0	3.842	0	100	0.923
9	tai10a	135028	135028	0	0.542	0	100	0.049
10	tai10b	1183760	1183760	0	0.197	0	100	0.037
11	tai12a	224416	224416	0	0.640	0	100	0.073
12	tai12b	39464925	39464925	0	0.717	0	100	0.058
13	tai15a	388214	388291	0.0198	4.855	0.08433	95	0.531
14	tai15b	51765268	51765268	0	0.415	0	100	0.089
15	tai17a	491812	493948.7	0.4340	26.937	0.41487	40	2.037
16	tai20a	703482	709926.3	0.9160	23.782	0.38367	10	6.083
17	tai20b	122455319	122510738.4	0.0453	9.483	0.13570	90	0.305
18	tai25a	1167256	1187017.9	1.6930	56.834	0.38947	0	49.033
19	tai25b	344355646	344367602.6	0.0035	18.168	0.01513	95	0.883
20	tai30a	1818146	1851329.6	1.8251	112.737	0.37674	0	96.424
21	tai30b	637117113	637271686	0.0243	91.640	0.03810	20	11.169
22	tai35a	2422002	2477518.6	2.2922	204.162	0.49207	0	173.891
23	tai35b	283315445	283487663.15	0.0608	118.488	0.08008	60	27.158
24	tai40b	637250948	637286051.4	0.0055	176.889	0.01111	70	29.569
25	tai64c	1855928	1855928	0	107.5595	0	100	33.971
26	lipa30a	13178	13178	0	17.403	0	100	1.091
27	lipa40a	31538	31777.75	0.7602	280.597	0.46111	20	181.749
28	lipa70b	4603200	4603200	0	529.561	0	100	279.521
29	esc16a	68	68	0	0.0615	0	100	0.011
30	esc32a	130	132.1	1.6154	123.5548	1.12026	25	41.088
31	esc64a	116	116	0	1.6071	0	100	0.306
32	esc128	64	64	0	507.5493	0	100	23.036
33	had20	6922	6922	0	0.9235	0	100	0.141
34	kra30a	88900	89279.5	0.426	57.006	0.58714	65	4.080
35	kra30b	91420	91590	0.186	120.770	0.10775	5	17.204
36	chr25a	3796	4201	10.669	57.264	4.24970	5	44.783

5.2. Comparison with other metaheuristic

Figure 15 compares average percentage of error in getting the BKS of ten QAPLib instances obtained by the proposed IHuS and the existing Genetic Algorithm (GA) [14]. The adaptation of GA for the QAP is the only contribution found in literature of an evolutionary algorithm that does not use the local search to improve the results. Therefore, it is the only available work in literature to compare results with the proposed work.

According to Figure 15, it is clear that the average percentage of error in getting the BKS obtained by IHuS is much better compared to that obtained by the GA. Knowing that the GA has been run on a better computer with Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz and 8.00 GB RAM.

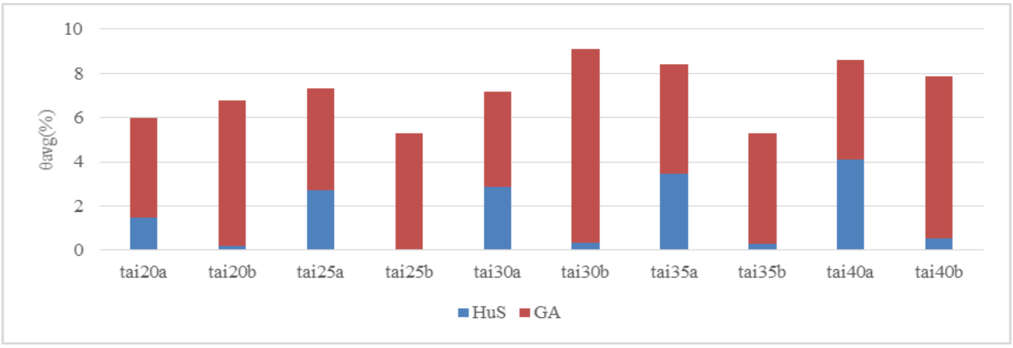

Figure 15. The average percentage of error in getting the BKS obtained by IHuS and GA for some instances of QAPLib

6. CONCLUSION

In this work, a new Improved Hunting Search algorithm is presented. It is the best evolutionary algorithm proposed to solve the quadratic assignment problem. QAPLib is the used library of the QAP instances to assess the performance of IHuS. To show the quality of IHuS, no local search is used. Three effective improvements are proposed: the one by one backup system, the leader moving toward hunters and the dynamic parameters. Very good results are obtained by applying IHuS to 36 instances of QAPLib instances. Finally, it is shown that the results obtained by IHuS are much better to that obtained by the existing evolutionary algorithm in literature.

REFERENCES

[1] R. Oftadeh, M. J. Mahjoob, and M. Shariatpanahi, “A novel meta-heuristic optimization algorithm inspired by group hunting of animals: Hunting search,” Computers & Mathematics with Applications, vol. 60, no. 7, pp. 2087–2098, 2010.
[2] F. Chebihi, M. E. Riffi, A. Agharghor, S. C. B. Semlali, “Improved Chicken Swarm Optimization algorithm to solve the Travelling Salesman Problem,” Indonesian Journal of Electrical Engineering and Computer Science, vol. 12, no. 3, 2018.
[3] S. C. B. Semlali, M. E. Riffi, F. Chebihi, “Discrete Chicken Swarm Optimization for the Quadratic Assignment Problem,” Indonesian Journal of Electrical Engineering and Computer Science, vol. 11, n. 3, 2018.
[4] F. Sayoti, M. E. Riffi, H. Labani, “Optimization of Makespan in Job Shop Scheduling Problem by Golden Ball Algorithm,” Indonesian Journal of Electrical Engineering and Computer Science, vol. 4, n. 3, 2016.
[5] A. Agharghor and M. E. Riffi, “Hunting search algorithm to solve the traveling salesman problem,” Journal of Theoretical & Applied Information Technology, vol. 74, no. 1, 2015.
[6] A. Agharghor, M. E. Riffi, and F. Chebihi, “A memetic hunting search algorithm for the traveling salesman problem,” 2016, pp. 206–209.
[7] B. Naderi, M. Khalili, and A. A. Khamseh, “Mathematical models and a hunting search algorithm for the no-wait flowshop scheduling with parallel machines,” International Journal of Production Research, vol. 52, no. 9, pp. 2667–2681, May 2014.
[8] A. Agharghor and M. E. Riffi, “First Adaptation of Hunting Search Algorithm for the Quadratic Assignment Problem,” in Europe and MENA Cooperation Advances in Information and Communication Technologies, vol. 520, Á. Rocha, M. Serrhini, and C. Felgueiras, Eds. Cham: Springer International Publishing, 2017, pp. 263–267.
[9] T. C. Koopmans and M. Beckmann, “Assignment Problems and the Location of Economic Activities,” Econometrica, vol. 25, no. 1, p. 53, Jan. 1957.
[10] L. Steinberg, “The Backboard Wiring Problem: A Placement Algorithm,” SIAM Review, vol. 3, no. 1, pp. 37–50, Jan. 1961.
[11] A. M. Geoffrion and G. W. Graves, “Scheduling Parallel Production Lines with Changeover Costs: Practical Application of a Quadratic Assignment/ LP Approach,” Operations Research, vol. 24, no. 4, pp. 595–610, Aug. 1976.
[12] M. Bashiri and H. Karimi, “Effective heuristics and meta-heuristics for the quadratic assignment problem with tuned parameters and analytical comparisons,” Journal of Industrial Engineering International, vol. 8, no. 1, p. 6, 2012.
[13] R. E. Burkard, S. E. Karisch, and F. Rendl, “QAPLIB–a quadratic assignment problem library,” Journal of Global optimization, vol. 10, no. 4, pp. 391–403, 1997.
[14] H. A. Zakir, “A Simple Genetic Algorithm using Sequential Constructive Crossover for the Quadratic Assignment Problem,” Journal of Scientific and Industrial Research, vol. 73, no. 12, pp. 763–766, 2014.