Indonesian Journal of Electrical Engineering and Computer Science
Vol. 11, No. 3, September 2018, pp. 925~935
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v11.i3.pp925-935

Discrete Chicken Swarm Optimization for the Quadratic Assignment Problem

Soukaina Cherif Bourki Semlali[1], Mohammed Essaid Riffi[2], Fayçal Chebihi[3]
[1]LAMAPI Laboratory, Department of mathematics, Faculty of Sciences,
University of Chouaib Doukkali, El Jadida, Morocco
[2,3]LAROSERI Laboratory, Department of Computer Sciences, Faculty of Sciences,
University of Chouaib Doukkali, El Jadida, Morocco

Article Info

Article history:

Received Nov 14, 2017
Revised Jan 8, 2018
Accepted May 26, 2018

Keywords:

Chicken swarm optimization
Combinatorial optimization problem
QAPLib
Quadratic assignment problem

ABSTRACT

The quadratic assignment problem (QAP) is a well-known combinatorial optimization problem, which could be applied to different applications. The main objective of this paper is to present the first discretization of the chicken swarm optimization algorithm (CSO) to solve quadratic assignment problem without using a local search, the adaptation of CSO in discrete case is based on redefining operations and operators of the original version. As known, the CSO is a stochastic method inspired from the behavior of chickens in swarm while searching for food. The experiments are performed on a set of 56 benchmark QAPLIB instances. To prove the choice of the adequate parameters, a study is conduct on CSO using simulations on certain instances. The discussion of different tests obtains competitive results compared with the known metaheuristic of Genetic algorithm based on SCX. The results demonstrate effectiveness of the proposed CSO-QAP to solve the quadratic assignment problem in term of time and quality of solutions. The proposed adaptation can be further applied by using a local search strategy such as 2-opt in order to solve the same problem or another NP-hard combinatorial problem.

Corresponding Author:

Soukaina Cherif Bourki Semlali,
LAMAPI Laboratory, Department of mathematics,
Faculty of Sciences,University of Chouaib Doukkali,
El Jadida, Morocco.
Email: Soukaina.cherif.b.s@ucd.ac.ma

1. INTRODUCTION

Combinatorial optimization occupies an important place in discrete mathematics and computer science, although the combinatorial optimization problems are often easy to define, but they are generally difficult to solve. Indeed, most of these problems belong to the class NP-hard such as Scheduling Workflow in Cloud Computing [1], Traveling Salesman Problem [2] and job shop scheduling problem (JSSP) [3]. Therefore, NP-hard problems don't have an effective solution for all the data, then we need to define a formal framework for many industry in science, engineering and business. In this work, we intend to discuss one of the most interesting combinatorial optimization problem used in the plant layout in order to determine the interaction and the distance between two facilities. The aim of the quadratic assignment problem is to fix the most effective arrangement of departments within the plant when the flow between departments remains constant during the horizontal planning. QAP is a classical NP-hard problem [4] in which it is necessary to find the optimal placement of n objects by taking into account both the cost of allocation of an equipment and its interaction with other equipments. In the field of location science, many practical problems can be

formulated as quadratic assignment problems (QAP) such as the study of Burkard [5] which applied the heuristic procedure proposed by Metropolis et al [6] and the approach of Laporte and Mercure [7], which used the problem of balancing hydraulic turbine runners.

Many methods were adapted to solve the quadratic assignment problem, among these adaptations are metaheuristics. Metaheuristic is a technique that seems to fit with the structure of any problem. Such as simulated annealing by MR Wilhelm in 1987 [8], the taboo search [9], the genetic algorithm [10], the greedy genetic algorithm [11], the ant colony algorithm [12] and Hunting Search Algorithm [13].

The Swarm optimization algorithms offer a new approach to solve several combinatorial optimization problems in engineering and computer sciences. These algorithms are proposed by mimicking the intelligence and the behaviors of the population in nature, such as ant colony optimization algorithm (ACO) [14], bee colony optimization [15], artificial bee colony algorithm (ABC) [16], bat-inspired algorithm [17], Particle Swarm Optimization (PSO) [18], spider monkey [19], firefly algorithms [20] and cuckoo search [21].

This paper applied the Chicken swarm optimization (CSO) which is an algorithm that simulates the behaviors of the chicken swarm. The CSO can effectively harness the intelligence of a chicken swarm to solve practically the quadratic assignment problem. The proposed adaptation of CSO is the first discretization of the chicken swarm optimization algorithm (CSO) to solve quadratic assignment problem, the new approach of CSO in discrete case is based on redefinition of operations and operators in order to to solve the most of QAPLib instances.

This paper is organized as follows: In the second section, we describe the quadratic assignment problem. In the third section, we present the metaheuristic used in this work to solve the QAP. Then, in the fourth section, we provide many simulations to justify the choice of parameters used and the results obtained of this contribution. Finally, in the last section, we close this paper with a conclusion.

2. QUADRATIC ASSIGNMENT PROBLEM

The management of a plant had long been a field of research, as a result, several approaches are developed over the years, but the resource allocation problem is still present. There are different formulations to solve the problem of facility management. One of the simple formulations to be modeled is the quadratic assignment problem (PAQ). Considered as one of the great challenge in combinatorial optimization, Quadratic Assignment Problem is introduced in the first time by koopmans and beekmann in 1957 [22]. The QAP can be reformulated into many combinatorial optimization problems such as the graph partitioning problems and the Travelling Salesman Problem [23].

The idea is to set two matrices of size (n,n), given as A = (a_{ij}) which refers to the flows between pairs of facilities and $B=\left(b_{\phi(i)\phi(j)}\right)$ for the distance of their locations. After the aforementioned descriptions, the purpose of this approach is to find a ϕ^* permutation among a set $\Pi(n)$ of permutations, which minimizes the cost allocation of facilities into locations, the mathematical formulation is given as follows:

$$C(\phi(*))=\min_{\phi\in\Pi}\sum_{i=1}^{n}\sum_{j=1}^{n}a_{ij}b_{\phi(i)\phi(j)} \quad (1)$$

3. CHICKEN SWARM OPTIMIZATION

The Chicken swarm optimization (CSO) is a stochastic algorithm inspired by the behaviors of a chicken swarm; and introduced by Meng, X.B. And al. [24]. The chicken swarm is divided into several groups; each group comprises one dominant rooster and many hens and chicks. Each type of chicken follows different movements.

The roosters, hens and chicks are identified by the fitness value of the chickens themselves. The chickens with the best fitness values would be designated to act as roosters or the leader of the group, they could search for food in a wider range of space,then they would be followed by the others chickens .

The chickens with the worst fitness values would be referred as chicks, which could seek for food around their mother. The others would be the hens, which randomly choose their groups and establish a mother-child relation with chicks. The two relations of dominance and mother-child between the hens and the chicks are all updated after few generations ,a parameter G is used to indicate the number of time steps, if $G\in[2;20]$ then the algorithm could give a good results and could easily fall into a local optimum, otherwise these statuses remain unchanged.

In a combinatorial optimization problem, the position of each chickens represents a solution. The roosters with the best fitness values are the most dominants and they could search for food in a wider range of space, therefore they have priority to access to food than the hens with lower fitness values, which are unstrained to eat food. On the other hand, Chicks move around their mothers to get food.

In [24] the number of roosters, hens, chicks and mother hens, depicted by RN, HN, CN and MN, respectively. The problem is solved in a D-dimensional space, $xt_{i;j}$ stands the positions of all the N chickens at time step t while searching for food.

The position update equation of the rooster can be formulated as:

$$x_{i,j}^{t+1} = x_{i,j}^{t} * (1 + Randn(0, \sigma^2)) \quad (2)$$

$$\sigma^2 = \begin{cases} 1, if\ f_i \le f_j \\ \exp\left(\frac{(f_k - f_i)}{|f_i| + \varepsilon}\right), otherwise\ k \in [1, N], k \ne i \end{cases} \quad (3)$$

Where

- $Randn(0, \sigma^2)$ is a Gaussian distribution, σ^2 is a standard deviation. The rooster index k is randomly selected from the rooster's group, and f is the fitness value of the corresponding x.

The hens can follow their group leaders to forage for food; furthermore, they may randomly locate and steal the good food found by other chickens. These situations can be represented as below:

$$x_{i,j}^{t+1} = x_{i,j}^{t} + S1 * Rand * (x_{r1,j}^{t} - x_{i,j}^{t}) + S2 \times Rand * (x_{r2,j}^{t} - x_{i,j}^{t}) \quad (4)$$

As $S1 = e^{\left(\frac{(f_i - f_r 1)}{(|f_i| + \varepsilon)}\right)}$ And $S2 = e^{(f_{r2} - f_i)}$

Where Rand is chosen randomly from [0, 1], r1 and r2 are two index $r_1 \ne r_2$, the first is the index of the rooster otherwise the second is the index of a random chicken from the swarm (rooster or hen).

At last, the position update equation of the chick is formulated in [24] as follows:

$$x_{i,j}^{t+1} = x_{i,j}^{t} + FL \times (x_{m,j}^{t} - x_{i,j}^{t}) \quad (5)$$

Where $m \in [1, N]$ is the index of the chick's mother and $FL \in [0, 2]$ is a randomly selected parameter to refer to the relationship between the chicks and its mother.

As the chicks only learn and get position from their mothers, they will easily fall into the local optimum as their mothers. In 2015, Dinghui Wu [25] added to the position update equation of the chicks two factors, the first one is the learning factor C which means that the chicks could get information from the rooster, the second one is W a self-learning coefficient for the chicks. The new equation is modified as follows:

$$x_{i,j}^{t+1} = W * x_{i,j}^{t} + FL * (x_{m,j}^{t} - x_{i,j}^{t}) + C * \left(x_{r,j}^{t} - x_{i,j}^{t}\right) \quad (6)$$

Where r is the index of the rooster and m is the index of the mother.

4. DISCRETE CSO ALGORITHM FOR QUADRATIC ASSIGNMENT PROBLEM

After observing each individual in the chicken swarm, it was found that the chicken has a capacity for communication and ability to learn, thus there is a strict hierarchical order, which can be used to model a new stochastic bio-inspired algorithm in order to solve many discrete optimization problem.

In the beginning, the swarm must be structured by declaring a population, its parameter configuration is simple, the fixed identities for each individual is defined after indicating the number of roosters, hens and chickens; then the hierarchical order between the identities of the chickens and the relations between the mother-child's are established; And finally, a certain parameter G must be fixed to

regularly update the relations in the chicken swarm so that the problem of falling into a local optimum is avoided.

Each chicken is represented by position, which is surrogated by a vector of N facilities assigned to locations, thereby a chicken can search food in a set of solutions S (the search space), and the position of a "selected chicken" is the current solution. The purpose of the discretization is to obtain the position of the chicken, which provide the minimum of the objective function $f : S \to \Re$.

This section describes a new adaptation of the original CSO algorithm; CSO-QAP is established by the redefinition of the operators and the operations of CSO (Figure 1 and 2).

4.1. Initialize the population

In this first part, after reading the instances of the QAPLIB, the algorithm begins with the creation of an initial population P of N elements randomly selected among p existing permutation of the search space, which contains all the possibilities (where $p = \frac{(n-1)!}{2}$).

Each possibility is obtained by assigning a facility to its location, and then it is represented by an array of integer of size n where each integer is an index that refer to the facility's location. The following example is a simple example of a QAP solution that represents an assignment of 6 installations to 6 locations; the solution (the table of array) represents a chicken in a known position.

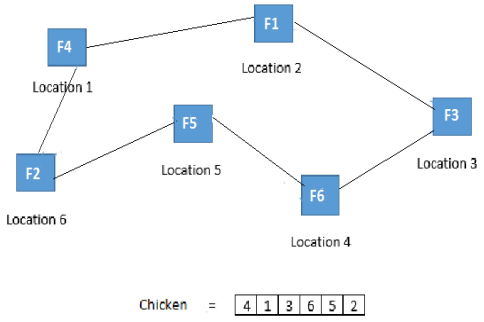

Figure 1. Example of Assigning Facilities to Locations in QAP

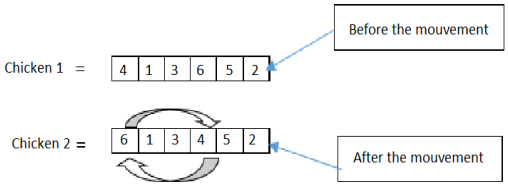

Figure 2. Movement of a Chicken in CSO

4.2. Establish the Hierarchical Order

The selected population is sorted and ranked after evaluating the fitness value of each individual, then a hierarchal order in the swarm is established, the chickens with the minimum fitness values are designated as the roosters, the number of the roosters, the hens, the chicks and the mother hens depicted by RN, HN, CN and MN.

The number of roosters can be expressed by the Equation 7, the number of hens by the Equation 8, otherwise the number of chicks is calculated by the Equation 9, an index is assigned to each type 1, 2 and 3 for the roosters, hens and chicks respectively.

$$RN = N * rp \quad (7)$$

$$HN = N * hp \quad (8)$$

$$CN = N - (RN + HN) \quad (9)$$

Where rp is the proportions of roosters and hp is the proportions of hens. MN represents the number of mother hens, it could be calculated by 10, and furthermore we assign to each mother a random id:

$$MN = HN * mp \quad (10)$$

The sizes of RN, HN, CN and MN directly determine the structure of the swarm, which affect the performance of CSO to solve QAP.

4.3. Create groups

The swarm is randomly divided into different groups according to the proportions obtained at the beginning of the algorithm, the number of groups is equal to number of roosters, we assign randomly to each mother an id hence the relationship between chicks and mother hens is scheduled indiscriminately in each group.

After G iterations, a redistribution of the swarm is required, in order that the roosters find the best solution of the swarm and the other individuals could move towards these solutions until the optimum is found.

4.4. The movements of a chicken

In this adaptation, a movement of a given chicken is represented by the permutation of two of its matrix elements. An example of a permutation is described as follows:

There are three types of chickens (roosters, hens and chicks), each one moves in a structured way according to a hierarchy defined in the previous sections. Thus, three types of movements are possible; each one is characterized by a set of operations and operators. During the discretization, we defined a set of operators, which are $\oplus$ $\ominus$ and $\otimes$ as follows:

4.4.1. Subtraction operation $\ominus$

$x_i \ominus x_j$ Represents a list of possible permutations applied for solution x_i to obtain solution x_j For example : x_i= 1 2 3 4 5 6 7 8 and x_j = 1 8 4 2 7 6 5 3 then $x_i \ominus x_j$= (2; 8)(5; 7)(2; 3)(2; 4).

4.4.2. Multiplication operation $\otimes$

This operation means that a random number of permutation is chosen between 0 and 1 in order to apply it to an operation. For example: R=0,5 and $x_i \ominus x_j$ = (2; 8)(5; 7)(2; 3)(2; 4). Then R$\otimes$ ($x_i \ominus x_j$) = (2; 8)(5; 7).

4.4.3. Addition operation $\oplus$

This operation indicates that the randomly chosen permutation is applied to move from solution i to the new solution j. For example: S1= 1 2 3 4 5 6 7 8 and R$\otimes$ ($x_i \ominus x_j$) = (2; 8) (5; 7) then xj= 1 8 3 4 7 6 5 2

4.4.4. The neighborhood

In the continuous formula of QAP, the objective is to find a permutation p of n elements that minimizes the Equation 1, which uses the fitness value in order to evaluate the solutions. On the other hand, in the discrete concept, we can use another approach to say that a solution is close to another neighboring permutation, then the differences between the solutions instead of the neighborhood by the fitness value.

4.4.5. Rooters movement:

In order to explain why the rooster with the better fitness value can seek for food in a larger space ,the best roosters will have the opportunity to do more permutations until it find a better solution than the one with less fitness value, allowing to search in a wider neighborhood. For that reason, we redefine the operation represented in Equation 2 by the Equation 11 as:

$$x_i^{t+1} = x_i^t \oplus Randn(0,\sigma^2) \otimes x_i^t \quad (11)$$

In another way, a rooster makes a self-permutation and Randn defines a random percentage of permutations between 0 and σ^2 to be chosen in order to apply it in the current solution (Rooster 'position).

Let's in the example as below x_i^t be the current rooster and x_i^{t+1} is the new solution then $Rand = 20\%$,furthermore the current Rooster move to the position x_i^{t+1}.The value of the parameter Randn is proportional to that of σ^2 where σ^2 = exp(*diff*) and *diff* is the percentage of difference between solutions, thus the more the differences persist between the current rooster and that of the new position, the more the probability of applying permutations increases exponentially, which means that the rooster with better fitness values can go to the new position by applying more permutations that is to say search in a wider range of space and the value of Randn is chosen randomly between [0,1] ,while for the other roosters of the swarm, the value is chosen randomly between 0 and a number less than 1.

4.4.6. Hen's movement

It is necessary to take into consideration that hen's solution must be different to the dominant rooster, otherwise in the iterations that follow a sets of roosters will move into the same positions, then the search space will be limited. The equation of hens is redefined as follows:

$$x_i^{t+1} = x_i^t \oplus S1 \otimes Randn(x_{r1}^t \,!\; x_i^t) \oplus S2 \otimes Randn \otimes (x_{r2}^t \,!\; x_i^t) \quad (12)$$

The first part of the Equation 12 represents the movement of hens following their leader of group with a well-defined probability; the second part means that hens could randomly steal food from other chickens (roosters or hens).

4.4.7. Chicks mouvement

In this adaptation, the movement of the chicks is divided into 3 steps, firstly the chicks will use a self-improvement operation to search for better solutions, then the chicks will look for food by learning from their mothers which means that the chicks will have more probability of resemblance between the chicks and their mothers. Finally, the chicks will use the same concept to look for the solution of the dominant rooster. The equation of hens is redefined as follows:

$$x_i^{t+1} = W \otimes x_i^t \oplus FL \otimes (x_m^t \,!\; x_i^t) \oplus C \otimes (x_r^t \,!\; x_i^t) \quad (13)$$

Where W is a parameter of self-learning, FL is a learning-Factor, which means that the chick learns from its mothers and C is a learning factor a parameter of resemblance between the chicks and their dominant rooster, which means that the chick could learn from of the roosters too.

4.5. Discrete CSO Algorithm CSO-QAP

The CSO-QAP in pseudo-code for optimizing the QAP is summarized as follows:

Discrete Chicken swarm optimization algorithm
1. Initialize the size of chicken population
2. Generate a random population of N chickens
3. Initialize the algorithm's parameters: N the number of chickens in the swarm, Rand, r1 is an index of rooster; r2 is an index of chickens, FL, C and w.
4. Evaluate the fitness values at t=0 for each chicken (save the global best solution)
5. Rank the chickens and establish a hierarchal order in the swarm
6. Randomly divide the swarm into different groups
7. Determine the relationship between the chicks and the mother hens in a group.
8. Find a new solution by updating the position of each rooster, hens and chicks using the new equations.
9. Update the new solution when it is better than the previous one.
10. Return to step 5 if G is reached until the maximum number of iterations is checked in.
11. Return results and visualization

5. EXPERIMENTAL RESULTS AND DISCUSSION

The performance of the new algorithm is evaluated by the computational experiments were performed on the QAP instances extracted from QAPLIB and tested 20 times in 100 iterations for each instance. The algorithm was implemented on a DELL in visual studio 2017 using the programing language C# and simulated with Intel(R) Core(TM) i7-6500 U CPU 2.5GHZ (4 CPUs) 2.6 GHz and 16.00 GB of RAM and Microsoft Windows 10 Professional (64-bit) operating system.

5.1. Benchmark instances

In order to evaluate the performance of our new algorithm, we test it on well-known benchmark instances of QAPLIB [26], we choose a set of 56 different instances among the 135 instances from the QAPLIB, then we compare its results with those of the existing methods in literature, the size of each instance is indicated in the instance name. There are many types of instances interpreting and des cribing several real and random problems. The first type is the instances obtained from a practical QAP applications

Real-life instances, the second is the randomly generated instances with a structure similar to real life instances, the third are Grid- based distance matrix instances and finally those in which the distances are calculated by the Manhattan distance.

5.2. Parametric analysis

In order to choose the optimum parameter values, we execute several tests on chr12a and bur26a. As shown in Figure 3, the optimal size of the chicken swarm should be N=500, which provide a better compromise between average of BFS and average of time for each run of the instance bur26a. Otherwise G=2 guarantees convergence of the algorithm, it may achieve a good results in minimum average time as represented in Figure 7 of the instance chr12a, this value ensures the robustness of the algorithm and guarantees the redistribution of the swarm.

The number of roosters, hens and chicks could affect the results; we can observe that the percentage of roosters must be greater than the percentage of hens as the percentage of chicks to ensure faster convergence. In Figure 4 RN should represent 10% of the population with a percentage of 21% for the hens as appears in Figure 5, while the rest of the swarm will be chicks respecting that CN=N-RN-HN, then CN=69% in Figure 6. We note that the gap between 63 and 72 gives better results for the average of BFS and the average of time. The movement of the roosters preserves the exploration; on the other side, the movement of hens and chicks performs the exploitation operations.

Moreover ,as shows in Figure 10, the average of BFS decreases when the self-learning parameter W is between 0.4 to 0,6, then we found that for W=0.5, the chicks could search for the solution in a larger range of space which avoid the problem of falling into local optimal. Furthermore FL is a randomly chosen number between 0,4 and 1, it allows that the chick could learn from the mother which ensure the robustness of the algorithm, in Figure 8 we set the parameter FL=0.4. Eventually the chicks could also learn from the leader of the group using a learning factor C for better results C=0.7 in Figure 9 which guarantees the optimal balance of intensification and diversification for our metaheuristic.

Table 1. The Parameters values for CSO-QAP

Parameters	Values
N (population size)	500
RN (Number of roosters)	10%
HN (Number of hens)	21%
CN (Number of chicks)	69%
G (Number of tours to update the algorithm)	2
C (Rooster learning factor)	0,7
FL (Hens learning factor)	0,4
W (self-learning factor)	0.5

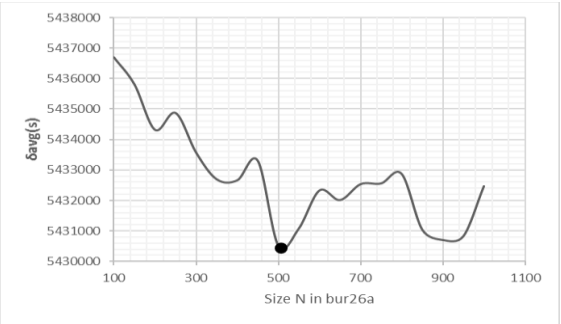

Figure 3. The average time while varying the population size

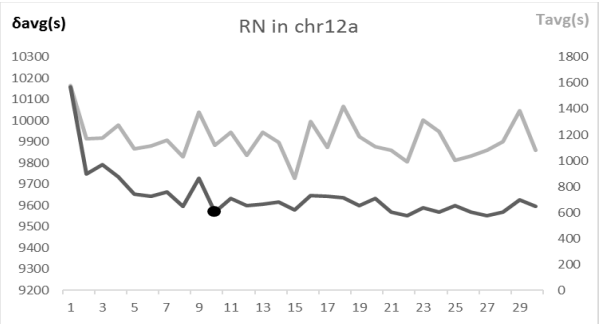

Figure 4. The average time and the average BFS while varying RN

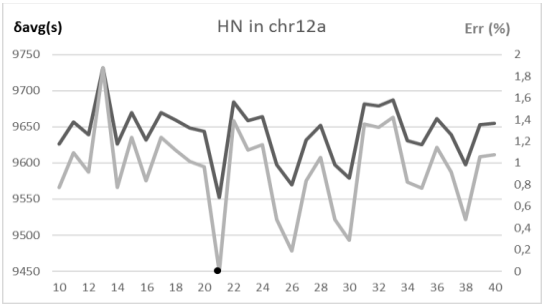

Figure 5. The average time and the percentage of ERR while varying HN

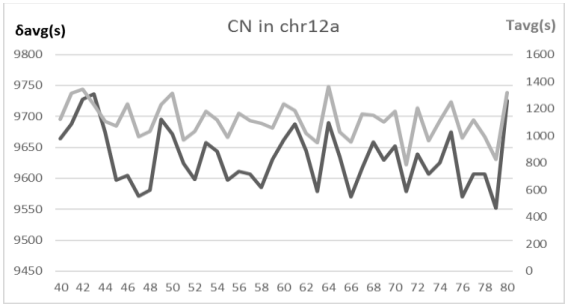

Figure 6. The average time and the average of BFS while varying CN

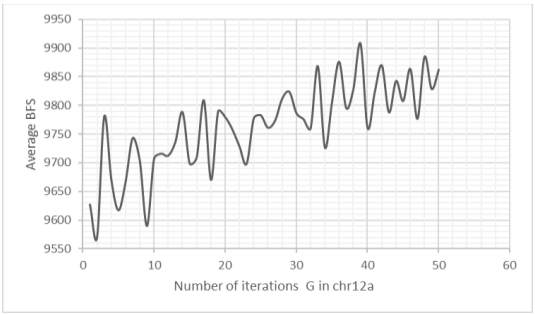

Figure 7. The average of BFS while varying the number of iterations G

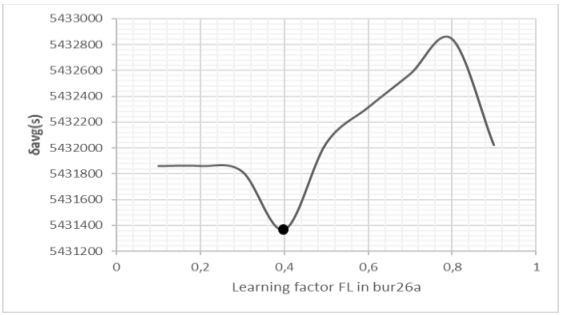

Figure 8. The average of BFS while varying the learning-factor FL

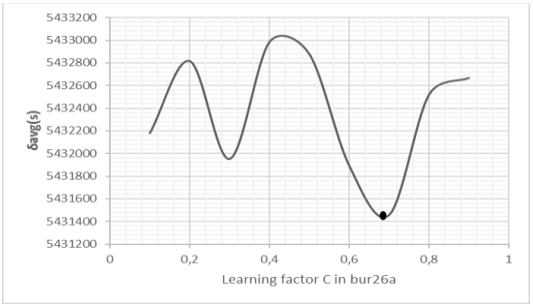

Figure 9. The average of BFS while varying the learning-factor C

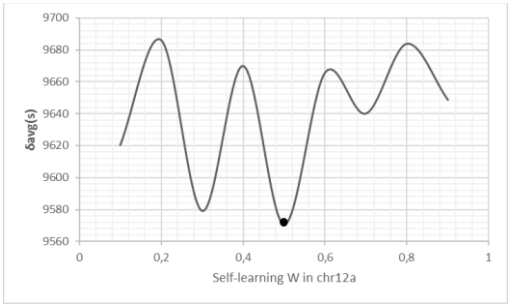

Figure 10. The average of BFS while varying the learning-factor W

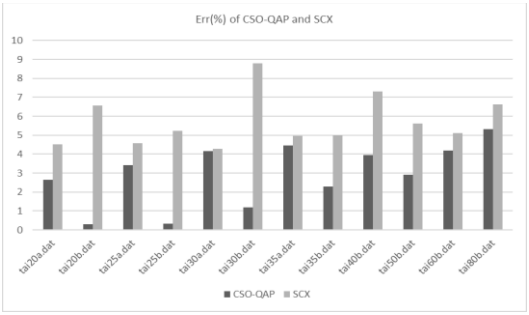

Figure 11. Comparison of the percentage ERR of CSO-QAP and SCX

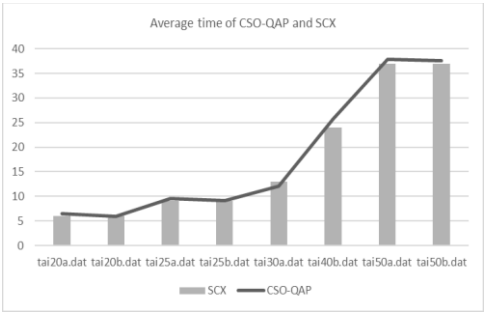

Figure 12. Comparison of the average time of CSO-QAP and SCX

Table 2. Results Obtained By Applying Cso-Qap to Some Qaplib Instances

Num	instance	Sopt	BFS	δ_{avg}	T*best*(s)	T*avg* (s)	ERR (%)	PSD	Succ (%)
1	bur26a	5426670	5426670	5432472,35	8,79	10,38	0,106922846	0,054008614	5
2	bur26b	3817852	3817852	3821399,6	5,47	9,41	0,09292136	0,103166591	20
3	bur26c	5426795	5426795	5427660,7	6,67	9,89	0,015952325	0,012423555	*15*
4	bur26d	3821225	3821239	3821555,25	9,55	10,02	0,008642516	0,007208045	*0*
5	bur26e	5386879	5386879	5387692,05	7,86	10,26	0,015093155	0,0083552	*5*
6	bur26f	3782044	3782044	3782358,25	6,14	9,65	0,008308999	0,009348965	*25*
7	bur26g	10117172	10117172	10118484,4	7,72	9,73	0,012972004	0,006476502	*5*
8	bur26h	7098658	7098658	7099200,15	7,85	9,54	0,007637359	0,005324074	*15*
9	chr12a	9552	9552	9651,8	0,26	1,12	1,04480737	2,099305098	*80*
10	chr15a	9896	10136	10746,3	2,99	3,12	8,59236055	3,32365169	*0*
11	chr15b	7990	7990	8682,5	1,67	2,93	8,667083855	6,727127302	*30*
12	chr18a	11098	11118	12990	4,31	4,67	17,04811678	6,405078609	*0*
13	chr20c	14142	14142	17212,9	5,54	6,09	21,71475039	11,69088172	*5*
14	chr25a	3796	4254	4835,1	9,19	9,4	27,37355111	4,756638722	*0*
15	els19	17212548	17212548	17521363,1	2,08	4,74	1,794127749	1,782569324	*25*
16	esc16a	68	68	68,2	0,05	1,14	0,294117647	0,879765396	*90*
17	esc16b	292	292	292	0,04	0,08	***0***	***0***	*100*
18	esc16c	160	160	160	0,15	0,59	***0***	***0***	*100*
19	esc16d	16	16	16	0,14	0,81	***0***	***0***	*100*
20	esc16e	28	28	28	0,17	1,45	***0***	***0***	*100*
21	esc16g	26	26	26	0,07	0,75	***0***	***0***	*100*
22	esc32a	130	146	154,4	17,86	18,91	18,76923077	2,641979023	*0*
23	esc32e	*2*	***2***	*2*	*0.16*	*0.27*	***0***	***0***	*100*
24	esc32g	6	6	6	0,21	0,84	***0***	*0*	*100*
25	esc32h	438	438	442,2	10,65	18,24	0,95890411	0,533235012	*10*
26	esc64a	116	116	117,2	14,78	57,54	1,034482759	0,995042985	*45*
27	had12	1652	1652	1652,7	0,16	1,32	0,042372881	0,110082927	*80*
28	had14	2724	2724	2724	0,3	1,01	***0***	***0***	*100*
29	had20	6922	6922	6930,5	1,7	6,21	0,12279688	0,101568084	*20*
30	kra30a	88900	90700	92239,5	13,54	14,42	3,756467942	0,821466726	*0*
31	kra30b	91420	92690	93568	13,13	14,1	2,349595275	0,774189402	*0*
32	ste36a	9526	9974	10224,8	18,97	21,01	7,335712786	1,932130437	*0*
33	tai12a	224416	224416	224416	0,27	0,64	***0***	***0***	*100*
34	tai12b	39464925	39464925	39464925	0,12	0,41	***0***	***0***	*100*
35	tai15a	388214	388214	391427,1	3,45	3,64	0,827662063	0,461065545	*5*
36	tai15b	51765268	51765268	51784256,4	0,45	2,24	0,036681738	0,066166326	*75*
37	tai17a	491812	497732	500698,7	4,45	4,66	1,80693029	0,440269737	*0*
38	tai20a	703482	703482	722071,7	5,81	6,49	2,642526746	0,770234121	*5*
39	tai20b	122455319	122455319	122805668,4	2,75	5,95	0,286103824	0,210481236	*30*
40	tai25a	1167256	1195890	1207142,1	9,07	9,58	3,417082457	0,477694806	*0*
41	tai25b	344355646	344355646	345430769,7	5,54	9,14	0,312213162	0,340907718	*20*
42	tai30a	1818146	1875012	1893483,6	11,56	12,04	4,14364963	0,482280256	*0*
43	tai30b	637117113	638977983	644659684,1	11,87	31,91	1,183859442	0,802901585	*0*
44	tai35a	2422002	2515012	2530168,5	49,23	52,48	4,465995486	0,340321432	*0*
45	tai35b	283315445	284893963	289765692,6	50,45	54,15	2,276701717	1,681206138	*0*
46	tai40a	3139370	3246432	3288017,1	22,61	38,88	4,734934079	0,565072522	*0*
47	tai40b	637250948	637409733	662512440,4	23,24	25,78	3,964135703	2,253301507	*0*
48	tai50a	4938796	5163546	5199574,5	36,87	37,79	5,28020392	0,296115161	*0*
49	tai50b	458821517	462945371	472189434,4	36,52	37,61	2,913533227	1,200443005	*0*
50	tai60a	7205962	7516098	7597042	52,28	59,87	5,427172666	0,368302781	*0*
51	tai60b	608215054	611893551	633742259	52,32	62,97	4,197068912	2,615227554	*0*
52	tai64c	1855928	1855928	1857875,9	47,06	75,92	0,10495558	0,084108686	*15*
53	tai80a	13499184	14155210	14196342,5	109,62	115,62	5,164449199	0,161305347	*0*
54	tai80b	818415043	845088215	861842189,4	109	113,17	5,306249778	0,670824532	*0*
55	tai100a	21052466	22064510	22113282,5	199,27	206,54	5,038918006	0,106372369	*0*
56	tai100b	1185996137	1215752011	1228796991	204,57	220,1	3,608852695	0,841996962	*0*

5.3. Discussion of results

We performed several tests on a set of different instances. Moreover, the Table 2 summarizes the numerical results of the CSO-QAP algorithm for 56 instances of 20 runs, we can notice that the optimal solutions is found in 60% of the instances especially with small size. The results in Table 2 summarizes the obtained results by applying the CSO-QAP for 56 instances of QAPLIB over 20 independent runs. The first column is the name of instance; the S*opt* indicated in the column two present the best Known solution of each instance in the QAPLIB documentation, the third column present the best found solution by the CSO-QAP (BFS), the average of the best found solution δ_{avg} is indicated in the fourth column. The remaining columns contains the measures use to perform the quality of the solution which are: the best run time T*best*,

the time average T*avg*, the average percentage of error Err in 16,the percentage of the Standard Deviation PSD in 15 and the last column is the percentage to get the BFS (Succ). The quality of solution is measured by: 1) The percentage of the Standard Deviation as:

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

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

2) The error percentage is calculated by: (15)

$$ERR = \frac{\left(\delta_{avg} - S_{opt} \right)}{S_{opt}} \times 100$$

5.3. Comparison with other metaheuristic

We compare the average percentage of error to get the best Known solution in our proposed work with the results obtained by applying a simple genetic algorithm using sequential constructive crossover for the quadratic assignment problem [27]. In this contribution of Genetic algorithm, SCX is applied to solve the quadratic assignment problem without using the local search to improve the results, therefore the GA has been run on a computer with Intel(R) Core(TM) i7-3770 CPU @3.40GHz and 8.00 GB RAM. Figure 11 shows that the proposed CSO-QAP is much better in term of the average percentage of error than SCX in the GA contribution.

Furthermore, in order to prove the robustness of the algorithm, Figure 12 presents a comparison between CSO-QAP and SCX [27]. The average of time obtained by the CSO-QAP is almost the same of the average time obtained by SCX algorithm in most of QAPLIB instances especially in small instance as shows in Figure 10; it is obvious that CSO-QAP method is very effective to solve the QAP instances (tai20a, tai20b, tai25a, tai25b, tai30a) in a reasonable time.

6. CONCLUSION

In this paper, we applied chicken swarm optimization (CSO), as metaheuristic for solving the QAP without using a local search. We compare as well the solution found with the best-known solution, which is introduced in QAPLIB. The results show that the proposed algorithm have effectively demonstrated the ability to solve QAP and obtain a promising results in terms of the quality of solutions and the computing time .The obtained results are compared with the SCX in the GA contribution. CSO-QAP takes advantage of intensification, diversification approaches, the movement of searching for food could be seen as intensification, and the diversification could be represented by the reorganization of the swarm.

In future research, we can conduct different comparisons between several metaheuristics by using different instances of QAP. We can also add low-level heuristics to solve effectively the quadratic assignment problem then we will aim to applicate the improved CSO-QAP algorithm in other combinatorial optimization problems, especially for problems with high dimensions.

REFERENCES

[1] Sheng-Jun Xue and Wu Wu. "Scheduling workflow in cloud computing based on hybrid particle swarm algorithm". *Indonesian Journal of Electrical Engineering and Computer Science*, 10(7):1560–1566, 2012.
[2] Yabo Luo, Min Liu, Zhongxiao Hao, and Dongbo Liu. "An improved nsga-ii algorithm for multi-objective traveling salesman problem". *Indonesian Journal of Electrical Engineering and Computer Science*, 12(6):4413–4418, 2014.
[3] Fatima Sayoti, Mohammed Essaid Riffi, and Halima Labani. "Optimization of makespan in job shop scheduling problem by golden ball algorithm". *Indonesian Journal of Electrical Engineering and Computer Science*, 4(3):542–547, 2016.
[4] Sartaj Sahni and Teofilo Gonzalez. "P-complete approximation problems." *Journal of the ACM (JACM)*, 23(3):555–565, 1976.
[5] Rainer E Burkard and Franz Rendl. "A thermodynamically motivated simulation procedure for combinatorial optimization problems". *European Journal of Operational Research*, 17(2):169–174, 1984.

[6] Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. "Equation of state calculations by fast computing machines". *The journal of chemical physics*, 21(6):1087–1092, 1953.
[7] Gilbert Laporte and Hélène Mercure. "Balancing hydraulic turbine runners: A quadratic assignment problem". *European Journal of Operational Research*, 35(3):378–381, 1988.
[8] Mickey R Wilhelm and Thomas L Ward. "Solving quadratic assignment problems by 'simulated annealing'". *IIE transactions*, 19(1):107–119, 1987.
[9] Eric Taillard. "Robust taboo search for the quadratic assignment problem". *Parallel computing*, 17(4-5):443–455, 1991.
[10] David M Tate and Alice E Smith. "A genetic approach to the quadratic assignment problem". *Computers & Operations Research*, 22(1):73–83, 1995.
[11] Ravindra K Ahuja, James B Orlin, and Ashish Tiwari. "A greedy genetic algorithm for the quadratic assignment problem". *Computers & Operations Research*, 27(10):917–934, 2000.
[12] Alfredo Reyes Montero and Abraham S´anchez L´opez. "Ant colony optimization for solving the quadratic assignment problem". In *Artificial Intelligence (MICAI), 2015 Fourteenth Mexican International Conference on*, pages 182–187. IEEE, 2015.
[13] Amine Agharghor and Mohammed Essaid Riffi. "First adaptation of hunting search algorithm for the quadratic assignment problem". In *Europe and MENA Cooperation Advances in Information and Communication Technologies*, pages 263–267. Springer, 2017.
[14] Marco Dorigo, Vittorio Maniezzo, and Alberto Colorni. "Ant system: optimization by a colony of cooperating agents". *IEEE Transactions on Systems, Man, and Cybernetics*, Part B (Cybernetics), 26(1):29–41,1996.
[15] Panta Lu˘ci´c and Du˘san Teodorovi´c. "Computing with bees: attacking complex transportation engineering problems". *International Journal on Artificial Intelligence Tools*, 12(03):375–394, 2003.
[16] Dervis Karaboga. "An idea based on honey bee swarm for numerical optimization". Technical report, Technical report-tr06, Erciyes University, engineering faculty, computer engineering department, 2005.
[17] Xin-She Yang. "A new metaheuristic bat-inspired algorithm. Nature inspired cooperative strategies for optimization (NICSO 2010)", pages 65–74, 2010.
[18] James Kennedy. "Particle swarm optimization. In Encyclopedia of machine learning", pages 760–766. Springer, 2011.
[19] Sandeep Kumar, Rajani Kumari, and Vivek Kumar Sharma. "Fitness based position update in spider monkey optimization algorithm". *Procedia Computer Science*, 62:442–449, 2015.
[20] Minmei Huang, Jijun Yuan, and Jing Xiao. "An adapted firefly algorithm for product development project scheduling with fuzzy activity duration". *Mathematical Problems in Engineering*, 2015, 2015.
[21] Amir Hossein Gandomi, Xin-She Yang, and Amir Hossein Alavi. "Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems". *Engineering with computers*, 29(1):17–35, 2013.
[22] Tjalling C Koopmans and Martin Beckmann. "Assignment problems and the location of economic activities". *Econometrica: journal of the Econometric Society*, pages 53–76, 1957.
[23] Panos M Pardalos, Henry Wolkowicz, et al. "Quadratic Assignment and Related Problems: DIMACS Workshop", May 20-21, 1993, volume 16.American Mathematical Soc., 1994.
[24] Xianbing Meng, Yu Liu, Xiaozhi Gao, and Hengzhen Zhang. "A new bio-inspired algorithm: chicken swarm optimization". In *International Conference in Swarm Intelligence*, pages 86–94. Springer, 2014.
[25] Dinghui Wu, Fei Kong, Wenzhong Gao, Yanxia Shen, and Zhicheng Ji. "Improved chicken swarm optimization". In *Cyber Technology in Automation, Control, and Intelligent Systems (CYBER), 2015 IEEE International Conference* on, pages 681–686. IEEE, 2015.
[26] Rainer E Burkard, Stefan E Karisch, and Franz Rendl. "Qaplib–a quadratic assignment problem library". *Journal of Global optimization*,10(4):391–403, 1997.
[27] ZH Ahmed. "A simple genetic algorithm using sequential constructive crossover for the quadratic assignment problem". 2014