JEAS
Journal of Economics and Administrative Sciences
E-ISSN: 2148-1792 P- ISSN: 1302-2024
Volume 3, Supplement Issue 2
www.jeasweb.org

An artificial bee colony meta-heuristic algorithm to solve quadratic assignment problem (QAP)

*Mohammad Amir Fazeli**
Master of Industrial Engineering, Faculty of Industry,
Ershad Damavand Institute of Higher Education, Iran.
**Corresponding Author*

Akbar Alem Tabriz
PhD, Faculty of Management and Accounting,Shahid Beheshti University, Tehran, Iran.

Seyed Mohammad Seyed Hosseini
PhD, Faculty of Industrial Engineering, University of Science and Technology,Tehran, Iran.

Mohammad Reza Lahooti Eshkevari
PhD, Faculty of Industrial Engineering, Ershad Damavand Institute of Higher Education, Iran.

ABSTRACT
The big cases of quadratic assignment problem are still very challenging and no way has yet been discovered to find accurate solutions to this difficult problem. According to the results of artificial intelligence, which is an effective meta-heuristic method for solving many problems, it has been proposed as a suitable candidate to obtain optimal solutions to NP problems. Therefore, the proposed algorithm in this paper is hybrid of bee optimization and artificial intelligence as the first meta-heuristic algorithm with taboo search for quadratic assignment optimization. The powerful taboo search method is used to simulate the exploratory processes and exploitation of bees. According to the obtained results for the standard problems solved from the QAPLIB library by the proposed algorithm, the performance of the proposed optimization algorithm is quite competitive with the optimized algorithms that have been reported to date.
Keywords: Bee Algorithm; Artificial Intelligence Algorithm; Quadratic Assignment; Optimization; Parallel calculations; Taboo Search.

Introduction

QAP in transportation systems, telecommunications, signal processing, typewriters, keyboard design, board wiring, layout, turbine balance, scheduling, data assignment, vendor scrolling, packaging, max group,

linear sorting and the graph partitioning problem [1] has been studied. Despite offering various solutions to this problem, QAP is still one of the most difficult hybrid optimization problems, and there is no precise algorithm for solving large-scale problems in practical computational time.

The quadratic assignment problem (QAP) was first introduced by Koopmans and Beckmann in 1957 [2]. The assignment problem is a set of facilities to a set of locations, so that the total assigned cost is minimized. QAP is in the NP-complete class and is one of the most difficult combinatorial optimization problems. There is no precise algorithm for solving problems larger than 35 locations with practical computational time. The smaller examples of the QAP problem can be solved with precise algorithms per minute/hour. However, due to its adverse behavior, the solution of larger QAP samples can take even hundreds of years to complete with a robust algorithm by a single processor. Therefore, many meta-heuristic algorithms have been proposed to solve QAP. The proposed algorithms are influenced by the fact that which can discover the optimal solutions in practical times. Artificial Bee Colony (ABC) is a meta-heuristic algorithm proposed by Darwish Karboga in 2005 [3].

The idea of using bees' social life to solve optimization problems has been presented by many researchers in different forms. Over the past decade, various algorithms have been developed using bee behavior patterns such as bee system (BS), BCO, ABC, MBO, Bees Algorithm, HMBO, Bee Hive, Artificial Bee Colony, and VBA algorithms to solve various types of optimized problems. ABC uses bee intelligence and population-based search methods. ABC uses common simple parameters such as colony size and maximum number of cycles to solve solvable problems (NP-Complete) such as visitor scrolling, scheduling, and finite optimization problems [5].

There are some new parallel ABC algorithms in the literatures. Subotic et al., have proposed three different parallel ABC algorithms. ABC parallel algorithms are in two independent parallel execution modes and two variations of several parallel particles. Using the independent parallel execution method, they are obtained faster than the ABC algorithm due to the high computability of multi-core processors. They show better results than the consecutive version of the original ABC algorithm. Communication methods improve solution quality with different ratios between exploration and exploitation.

Chmiel and Kwiecien propose an evolutionary algorithm for QAP by quantum. They show how to adapt QAP such as crossover and mutation operators, and introduce quantum principles in specific procedures [6].

Yagmur et al., introduced a parallel version of Breakout Local Search (BLS) [7]. They use a Lunstein metric method to examine the similarity of new starting points. The proposed BLS algorithm (BLS-OpenMP) combines multidisciplinary computations using OpenMP. Dokeroglu proposes Teaching-Learning-Based Optimazation (TLBO) Algorithms to solve QAP [8]. People are trained with recombinant operators and later processed by a powerful Tabu search engine. Hyper Ebtekar has recently introduced an approach to solving the challenging NP-Hard combination optimization problem using a set of low-level innovative methods. Abdul et al., propose a way to improve the Whale Optimization Algorithm.

The proposed algorithm is reinforced with local search. This algorithm has been tested in many cases of QAP and it has been reported that it obtains almost optimal solutions with reasonable execution time [9]. Kola et al., consider new cases of QAP polynomials with a specific diagonal structure. They acquire a new class of special solvable polynomial cases QAP [10]. Bogel et al., assign a linear model to a quadratic model. This is provided by a family of assignment-induced editing paths between nodes and shows that the graph editing distance is equivalent to QAP [11].

In this study, we were proposed a new parallel island algorithm to find better results for samples larger than QAP. Recently, much attention has been paid to the application of parallel metacognitive algorithms in many hybrid problems [12]. Meta-heuristic parallelization can significantly increase the efficiency of the optimization algorithm [13]. In this research, the behavior of worker, observer and scout bees is modeled using the distributed parallel computational paradigm and by selecting the search parameters comparatively, the ABC algorithm consisting of local and global search methods will be optimized.

According to the above, the following objectives were pursued in this study:

1. Solve quadratic assignment problem using an artificial bee colony meta-heuristic algorithm
2. Optimize the search method by comparatively adjusting the parameters of the proposed method

3. Parallelization of the proposed meta-heuristic algorithm to increase efficiency

Materials and Methods

Problem modeling

QAP can be formulated using the nxn, A, B, C matrices.

$$A = (a_{ik}), B = (b_{jl}), C = (c_{ij}), \quad (1)$$

Where a_{ik} is the amount of current from facility i to center k, b_{jl} is the distance from location j to location L, c_{ij} is the cost of installing facilities i at location j. The QAP form of Koopmans and Beckmann can be described as follows.

$$min_{\phi \in S_n}\left(\sum_{i=1}^{n}\sum_{k=1}^{n} a_{ik} b_{\phi(i)\phi(k)} + \sum_{i=1}^{n} c_{i\phi(i)}\right) \quad (2)$$

S_n is the change in the number of numbers 1,2,...n. $a_{ik}b_{\phi(i)\phi(k)}$ is the transfer cost from facility i in location $\phi(i)$ to facility k in location $\phi(k)$.

$c_{i\phi(i)}$ is the cost of installing facility i at location $\phi(i)$ and the cost of transferring to all other locations k, which is determined at location $\varphi(1)$, $\varphi(2)$, ..., $\varphi(n)$. In cases where there is no C expression, Lawler introduced the fourth cost array instead of the three matrices and obtained the general form of QAP [44];

$$min_{\phi \in S_n}\left(\sum_{i=1}^{n}\sum_{k=1}^{n} d_{i\phi(i)\phi(k)}\right) \quad (3)$$

The relationship to Koopmans and Beckmann's problem is as follows:

$$d_{ijkl} = a_{ik}b_{jl}(i, j, k, l = 1,2, \dots, n; i \neq k \text{ or } j \neq l) \quad (4)$$
$$d_{ijij} = a_{ii}b_{jj} + c_{ij}(i, j = 1,2, \dots, n)$$

Proposed Method

ABC is a population-based meta-exploratory optimization method. Bees make up the population of this algorithm and each bee is given the optimal solution (food source) of the QAP sample. It is assumed that each solution is a food source and the amount of nectar in each source indicates the quality of each solution. Our model uses three types of bees: worker bees, observers, and scouts. Scout bees are looking for food sources, and working bees go to the food source and return to the hive to share their information about the flight area. When the task of collecting nectar from the worker bee is completed, it becomes an observer bee and seeks new food sources. The observer bees watch the flight of worker bees and choose food sources depending on the flight. In the first stage, the initial population algorithm is created and after this process, the optimization is repeated using worker, observers, and scouts bees. Scout bees begin the search process. In our developed model, there is a single hive for ABC-QAP. In the parallel version of ABC-QAP, the number of hives is equal to the number of processors in the parallel computing environment. Also, each bee starts out as a scout bee and explores the QAP search space. After spending some time during the exploration process, he returns to the hive and shares his information. Then, by evaluating the results, they return to the best available food sources.

Bees use various taboo search parameters (such as taboo list size and aspiration values) and begin abusing food sources in more detail. The number of scout bees in the exploration phase is 1000. This parameter provides a good balance between the exploration and exploitation steps. The best food source is selected after the exploration phase and exploited by the observing bees.

Adjust the size of the taboo list and the amount of aspiration

The size of a taboo list is an important parameter for finding optimal values.

For each unit and location of QAP, the last iteration that occupies that location is stored in the list.

Small taboo-sized lists can cause searches to be processed in the same areas and be stuck in local optimization, while larger lists can prevent better research and lead to lower quality solutions. Due to the excessive size of the taboo list, optimization may do more repetition than necessary. The lower and upper limits of the taboo list size are set by Taillard to be between [0.9 x n - 1.1 x n] (where n is the size of the problem).

Algorithm 1- ABC-QAP algorithm

1 $start$
2 $scout\ bees\ search\ for\ food$
3 $sout\ bees\ return\ to\ the\ hive\ and\ dance$
4 $onlooker\ bees\ evaluate\ the\ food\ sources$
5 $employeed\ bees\ travel\ to\ the\ food\ sources$
6 $use\ taboo\ search\ for\ nectar\ collecting\ and\ return\ to\ hive$
7 $collect\ the\ solution\ in\ the\ hive$
8 $if(termination\ condition)$
$yes \rightarrow finish$
$No \rightarrow goto\ 2$

These parameters are seen as average values that provide a reasonable execution time during the optimization process. The dynamic size of the taboo list between this value provides a very effective way to optimize the process and the amount of respiration [14]. Therefore, we use a technique that dynamically changes the size of the taboo list in this study. The aspiration value of the taboo search allows that solution to be implemented if a better move is found than the available solutions. The aspiration rate of the taboo search process is usually recommended according to classical methods. However, this value can vary depending on the structure of the problem; therefore, we propose a dynamic aspiration value in our study. Its value changes per 100,000 repetitions of taboo searches and can be a good way to search locally with smaller values. However, once the search process is stuck in the local optimization, the optimization must leave this space while preserving the previous experience (by eliminating the existing position of the current solution). A set amount of aspiration can provide such a mechanism. The value of our aspiration dynamics varies between [n - (n x n x 10)], smaller values of aspiration can search for spaces close to the current solution, while larger values provide a local optimal escape mechanism. In this way, a diverse space search is created with hundreds of processors which it optimized the same problem with different taboo list sizes and aspiration values.

Rapid evaluation of neighbor solutions

Swapping two different locations from an existing QAP solution and creating a new relocation is a very effective way to cross the QAP solution search space. This approach allows us to quickly calculate newly changed costs just by finding the difference cost.

Formally, the calculation of the fit value of a QAP permutation is a function of the second order O(N2) when the calculation is done from the beginning, while when the difference between the two permutations is calculated, it is a function of the first order O (N). This method is used as a method to calculate the performance increase in our proposed algorithm, PABC-QAP. Taboo robust search uses a matrix to store the costs of each possible exchange, and these costs are added to obtain the cost of the new solution.
Starting from solution φ, the neighbor solution π is obtained using units r and s:

$$\pi(k) = \phi(k)\ \forall\ k \neq r, s$$
$$\pi(r) = \phi(s)$$
$$\pi(s) = \phi(r) \qquad (5)$$

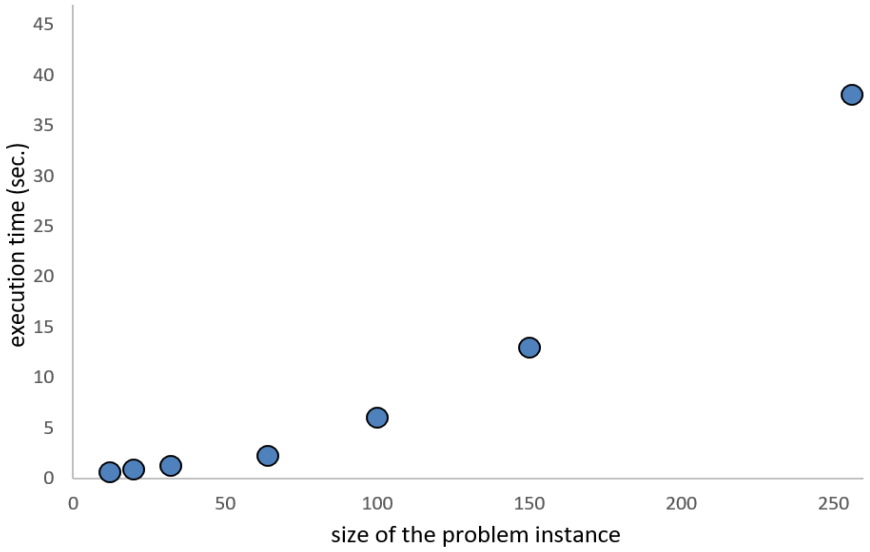

Figure 1: Execution time of problem instance according to their sizes

Figure 1 shows the execution times of the problems, from size 12 to 256. 10,000 neighbors are searched for problem instance using the quick assessment method. During execution, a linear increase versus time is observed, and this method provides a great advantage during the exploration and operation stages of ABC optimization.

Parallel ABC algorithm for QAP

For QAP problem instance, we need to develop a more robust and scalable parallel algorithm. Therefore, we propose the island parallel ABC optimization algorithm for QAP. In this algorithm, there is one main node (processor) and many slave node (sub-nodes) in the parallel computing environment. The name of the proposed algorithm is PABC-QAP. Each slave node in the computing environment works on a separate hive with a different set of bees (scout, observer, and worker). The PABC-QAP algorithm initiates a diverse exploration phase on each processor. This is generated randomly by using QAP permutation with initial onset. The seed mechanism in each processor is initialized with current processor time X#.

This amount of seed is able to provide a good variety of permutations for each processor. During the exploration phase, the taboo search algorithm is executed with a small number of failures (100) and 1000 restarts. This method of performing a taboo search with fewer repetitions provides a good way to explore the QAP search space. One thousand explorations are done with 255 processors. In total, 255,000 exploration processes run in parallel. Due to the minimal connection between the processors, this algorithm is scalable and works almost at a high linear speed. In some problem instance where we cannot obtain the best/optimal values with ABC-QAP, it is possible to obtain the best values even in the exploration phase with the parallel PABC-QAP algorithm. For very difficult problems like tai100a and tai256c, we still have to spend most of our optimization time in the algorithm operation phase. Because the right balance between the exploration and operation stages of a process ensures efficient time optimization, we will perform experiments to understand exploration behavior in terms of time and deviation from the best results.

After completing the reconnaissance phase of the scout bees, the best transfer is sent to the exploitation phase. Then, Taboo search starts optimizing the current solution with more restarts and crashes. After the optimization process in each slave processor, the results and execution time of the optimizations are sent to the main node. Slave nodes may go through different execution times due to the different optimization process in each node. The main node receives the results from the slave nodes and reports the best result as the result of the PABC-QAP algorithm. Algorithm 2 provides the details of the PABC-QAP algorithm.

Algorithm 2: Parallel PABC-QAP algorithm

1 *start*
2 *scout bees search for food*
3 *scout bees return to the hive and dance*
4 *onlooker bees evalute the food sources*
5 *employees bees travel to the food sources*
6 *use taboo search for nectar collecting and return to hive*
7 *collect the solution in the hive*
8 *if (termination condition)*
9 *yes → receive solutions and report the best result → finish*
10 *No → goto* 2

Results of performance evaluation of the proposed algorithms

During the simulation, we use QAPLIB (QAP Standard Problem Library) [48]. 134 problem instance are solved from QAPLIB standard problem instance [4]. Most modern algorithms use this library. Thus, it provides a fair platform for evaluating new algorithms. The problem instance of this library arise from real-life or random applications (such as Manhattan distances from rectangular grids (Head12), hospital layout (kra30), and back wiring (Ste36a)). During the tests, each problem instance is tested 30 times and the average/best test results are reported.

The simulations are performed on the HP ProLiant DL585 G7, which has a 2.6 GHz AMD Opteron 6212 processor with 8 cores. It is possible to create 8 nodes per core (providing 64 possible cores simultaneously). Each processor has a 64-bit computing capacity and an AMD SR5690 chip.

Impact of increasing the number of processors

The number of processors has a major impact on the performance of the proposed parallel PABC-QAP algorithm. Figure 2 shows the deviation of the experiments in the tai50a sample as the number of processors increased (taboo search uses 1,000,000 number of failures and ABC uses 100 bees for the exploration phase). The experiments were repeated 30 times and their average value was reported.

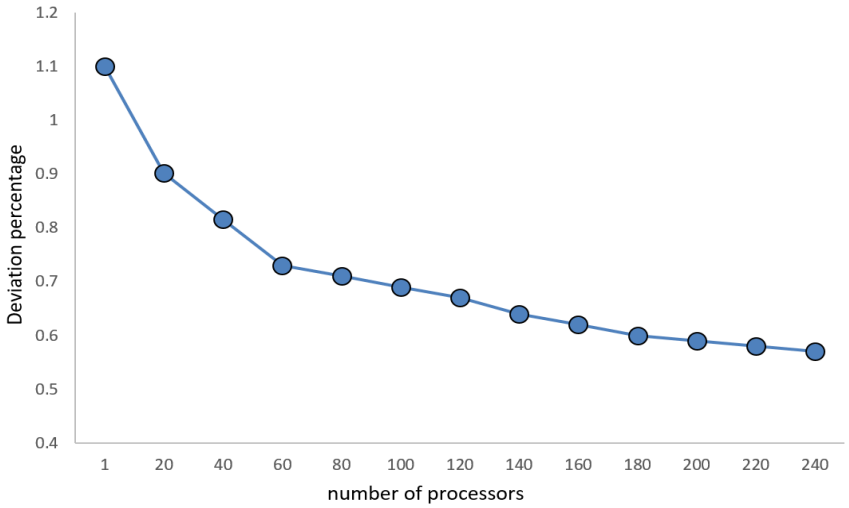

Figure 2: The effect of increasing the number of processors for the PABC-QAP algorithm

The x-axis is the number of searches and the y-axis is the optimization deviation from the best library results. If a proper diversification mechanism is provided in each processor and the possible number of

processors is used, the probability of finding the optimal value is significantly increased. Of course, providing adjusted parameters is another important advantage of optimization.
Our main goal is to ensure that the parallel environment can provide better tools while finding the optimal solution. The first thing to consider when examining parallel algorithms is to accelerate them.

However, the proposed island parallel ABC algorithm works on multiple hives and optimizes independent solution candidates instead of parallelizing the calculation of a solution optimization effort. The latter can often be a very difficult task and cannot provide linear (near-linear) acceleration, while island parallel ABC can effectively consume its computational time. The result has a 1.1% deviation with a single processor for the given settings. The deviation is 0.57% with 255 threads, which work on different hives and are in the memory of each processor. During the tests, obvious improvement is observed and the performance is improved by increasing the number of failures and repetitions of the taboo search.

Adjust the number of discoveries

The number of discoveries has a major impact on the performance of the optimization process. Figure 2 shows the deviation of some of the results for the tai50a problem by increasing the number of discoveries. Each excavation was performed 30 times. The problem instance of tai50a is a very difficult and medium-sized issue in the library. The x-axis is the number of searches and the y-axis is the optimization deviation.

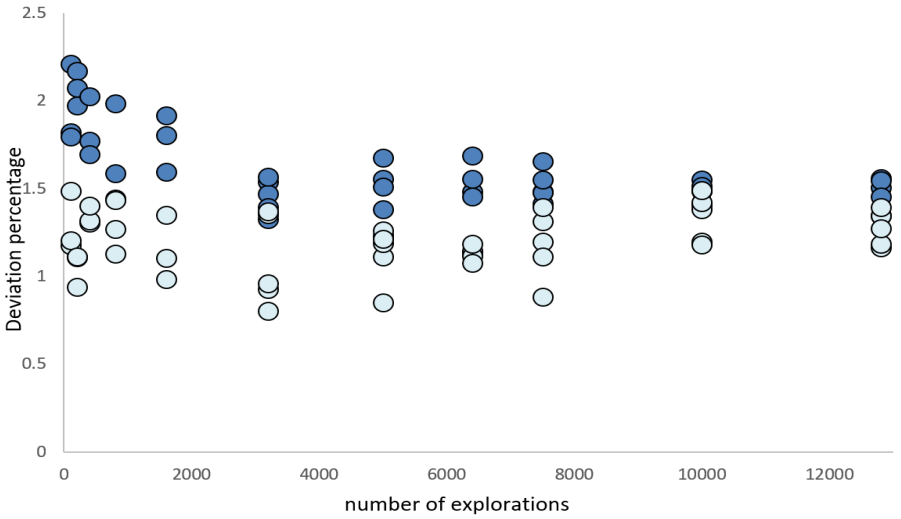

Figure 3. The effect of the number of discoveries for the tai50a problem

Starting from 1 to 130,000 explorations, we see the performance of the exploration process in optimization. In Figure 3, the dark markers are the results of the exploration, while the white markers are the results of the exploitation phase. Even the number of explorations has been done with 130,000 repetitions so that we can start better from the start of operation; and 1000 repetitions with a reasonable execution time is a good value for exploration. With the number of searches increasing by more than 2000 repetitions, no major advantage is observed. Therefore, we set the number of exploration activities as 1000 for all problems. Of course, this optimal value is not difficult for all cases, but a simple parameter adjustment method for the balance between the exploration and exploitation stages. When the list size and taboo aspiration values are well adjusted in the operation phase of ABC-QAP, more failures work better.

Results of implementation of problem instance in QAPLIB

First, we solve a problem instance with ABC-QAP to get the least deviation from the best-known solutions reported in QAPLIB. If the best result is not achieved, we apply different taboo list and aspiration settings to the ABC-QAP algorithm. Next, we run the PABC-QAP algorithm with 255 processors and the settings given in Tables 4-2 to 4-5.

Name of problem instance, best known solution value (BKS) of problem instance reported by QAPLIB, best result (found), average percentage of deviation from best known solution (APD), best percentage of our result deviation from BKS (BPD), algorithm runtime time (s), number of processors used during optimization (#proc.), the amount of aspiration of the taboo search process are given in the tables. If the problem instance is optimized with the PABC-QAP algorithm, the number of processors in the tables is reported to be more than 255. Harder problem instance not solved by the ABC-QAP algorithm are tested with a parallel version with 255 processors. The average execution time and deviation are reported in the tables.

Table 1 shows the four configurations of the taboo search algorithm that we use in our experiments. For small problem instance are applied configuration 1. For more complex problem instance, we mostly use configuration 2. Due to runtime constraints, we have to use configuration 3 for the tai256c problem instance. The size parameter of used taboo list in our experiments is provided by Taylard [15]. The range of aspiration values is used dynamically in the ranges given in Table 1. Setting 4 is used for the exploration step of the ABC-QAP algorithm. All problem instance in the QAPLIB library are solved during the experiments.

Table 1: Taboo search parameters

Configuration	Max. failure	Taboo list size	Scope of aspiration
1	50,000xn	lower limit=(9xn/10) – upper limit=(11xn/10)	[n - nxn]
2	50,000xn	lower limit=(9xn/10) – upper limit=(11xn/10)	[n - nxnx10]
3	20,000xn	lower limit=(9xn/10) – upper limit=(11xn/10)	[n - nxnx10]
4	1,000xn	lower limit=(9xn/10) – upper limit=(11xn/10)	[n - nxn]

The results for the 134 sample issues in QAPLIB are as follows:

125 problems are solved optimally according to the results in the standard library. 105 AND 21 of these results were obtained and solved by ABC-QAP and PABC-QAP parallel algorithm, respectively.

Problem instance bur, had, chr, els, esc, had, kra, lipa, nug, Rou, Scr, Sko (except sko100a) and Ste are solved with optimal ABC algorithms. The larger problems of taia and taib are the most difficult examples to deal with during simulation. The mean of the best reported deviations for tai problem instance is 0.092. In particular, tai50a, tai60a, tai80a, and tai100a have the highest deviation among all problem instance. For tai256c, we report one of the best results in the articles, namely 0.082% deviation. ABC algorithms spend most of their execution time on these problem instance. tai256c, tai150b and tai100a spend 71,129, 16,665 and 10,379 seconds during optimization, respectively.

Name of problem instance, best known solution value (BKS) of problem instance reported by QAPLIB, best result (found), average percentage of deviation from best known solution (APD), best percentage of our result deviation from BKS (BPD), algorithm runtime time (s), number of processors used during optimization (#proc.), the amount of aspiration of the taboo search process are given in the tables.

Conclusion

In this study, a combination of bee optimization algorithms and artificial intelligence for the quadratic assignment problem was proposed and we presented the new ABC optimization algorithms for QAP optimization. Accelerating these calculations using dynamic programming and parallel computing techniques ensures that the proposed algorithm explores/exploits more of the search space and thus has a better chance of finding better results at the same time.

Termination criteria and parameter settings of the proposed algorithms are important issues discussed in our study. Of course, implementing algorithms with more repetition provides a better chance of finding better solutions. However, being stuck around the local optimum and the effect of parameter-adjusted settings are the most important factors in meta-heuristic algorithms when increasing the number of repetitions. In order to provide a mechanism to avoid being stuck in the local optimization, we developed a parallel version of the algorithm. In this way, processors work on various parts of the problem.

This technique reduces the likelihood of our algorithm being clogged in the local optimization. Parameter setting can also be another major problem of meta-heuristic optimization algorithms. Simple parameter setting techniques can provide significant improvements to these algorithms.

The proposed ABC method works well for solving QAP. QAP large problem instance are still challenging. Therefore, we need to develop a parallel version (MPI) of the ABC algorithm. A distributed parallel memory version of ABC is provided. The path optimization method, taboo search, has been adapted to explore and exploit ABC algorithms. During the simulations, the taboo list and aspiration values of this local search method are adjusted to better optimize the results. Most benchmark problems are optimally solved with the new ABC algorithms. The right balance between exploration and exploitation provides better results. There are still good opportunities to get better results with higher number of processors, better tuning parameters, or new introduced exploratory techniques. As a first suggestion for future work, we can use ABC for other known hybrid problems. Many meta-exploratory methods have recently been introduced. Using parallel computing and adjusting their parameters at runtime can be interesting and has the potential to improve the solution of existing NP-Hard hybrid problems. The black box optimization performance is also an effective tool for evaluating the performance of new algorithms. This involves a wide range of benchmark problems. As a recommendation, this tool can be used to evaluate the performance of its new hybrid algorithm in the field of various problems.

References

[1] R. K. Adl, S. M. T. R. Rankoohi, A new ant colony optimization based algorithm for data allocation problem in distributed databases, Knowledge and Information Systems 20 (3) (2009) 349–373

[2] E. Cela, The quadratic assignment problem: theory and algorithms, Vol. 1, Springer Science & Business Media, 2013.

[3] D. Karaboga, An idea based on honey bee swarm for numerical optimization, Tech. rep., Technical report-tr06, Erciyes university, engineering faculty, computer engineering department (2005).

[4] D. Karaboga, B. Gorkemli, A combinatorial artificial bee colony algorithm for traveling salesman problem, Innovations in intelligent systems and applications (2011) 50–53.

[5] B. B. LD. Karaboga, Artificial bee colony (abc) optimization algorithm for solving constrained optimization problems, in: International fuzzy systems association world congress, Springer, 2007, pp. 789–798.

[6] W. Chmiel, J. Kwiecien, Quantum-inspired evolutionary approach for the quadratic assignment problem, Entropy 20 (10) ´ (2018) 781.

[7] Y. Aksan, T. Dokeroglu, A. Cosar, A stagnation-aware cooperative parallel breakout local search algorithm for the quadratic assignment problem, Computers & Industrial Engineering 103 (2017) 105–115

[8] T. Dokeroglu, Hybrid teaching-learning-based optimization algorithms for the quadratic assignment problem, Computers & Industrial Engineering 85 (2015) 86–101.

[9] M. Abdel-Baset, M. Gunsekaran, D. El-Shahat, S. Mirjalili, Integrating the whale algorithm with tabu search for quadratic assignment problem: A new approach for locating hospital departments, Applied Soft Computing.

[10] E. C¸ ela, V. Deineko, G. J. Woeginger, New special cases of the quadratic assignment problem with diagonally structured coefficient matrices, European journal of operational research 267 (3) (2018) 818–834.

[11] S. Bougleux, L. Brun, V. Carletti, P. Foggia, B. Gauz¨ ere, M. Vento, Graph edit distance as a quadratic assignment problem, ` Pattern Recognition Letters 87 (2017) 38–46.

[12] T. Kucukyilmaz, H. E. Kiziloz, Cooperative parallel grouping genetic algorithm for the one-dimensional bin packing problem, Computers & Industrial Engineering 125 (2018) 157–170.

[13] Y. Aksan, T. Dokeroglu, A. Cosar, A stagnation-aware cooperative parallel breakout local search algorithm for the quadratic assignment problem, Computers & Industrial Engineering 103 (2017) 105–115.

[14] F. Glover, Tabu searchpart ii, ORSA Journal on computing 2 (1) (1990) 4–32.

[15] E. Taillard, Robust taboo search for the quadratic assignment problem, Parallel computing 17 (4-5) (1991) 443–455. ´