Highlights

- ❖ An attempt is made to solve the NP-hard type combinatorial problems using Whale Optimization Algorithm (WOA).
- ❖ The proposed algorithm integrates the WOA with a Tabu search for solving the Quadratic Assignment Problem.
- ❖ The proposed algorithm is validated using the benchmark problems addressed in the literature.
- ❖ Results proved the effectiveness of the proposed algorithm.

Integrating the Whale Algorithm with Tabu Search for Quadratic Assignment Problem: A New Approach for Locating Hospital Departments

Mohamed Abdel-Baset^{*1}, M. Gunsekaran², Doaa El-Shahat³ and Seyedali Mirjalili⁴

^{1,3}Faculty of Computers and Informatics, Department of Operations Research, Zagazig University, Egypt; Email: ielhenawy@zu.edu.eg; analyst_mohamed@zu.edu.eg.

²University of California, Davis, United States; E-mail: gunavit@gmail.com

⁴School of Information and Communication Technology, Griffith University, Nathan Campus, Brisbane, QLD 4111, Australia; Email: seydali.mirjalili@griffithuni.edu.au

Abstract

The Quadratic Assignment Problem (QAP) is a combinatorial NP-hard optimization problem that is not solvable in a polynomial time. It has a large number of real-world applications in diverse fields (e.g. facility arrangement in a hospital). **The Whale Optimization Algorithm is a new meta-heuristic that achieves a great success in solving the continuous problems.** In this paper, we propose a memetic algorithm using the Whale optimization Algorithm (WA) Integrated with a Tabu Search (WAITS) for solving QAP. **In fact, this work employs Tabu Search to improve the quality of solution obtained by WA for QAP problem as a local search algorithm. This is an attempt to improve the convergence speed and local search of WA as its main drawbacks.** Due to the combinatorial nature of QAP, the continuous values generated from the standard WA were converted to discrete values by the largest real value mapping. The WAITS algorithm is enhanced by a local search that defines a set of neighborhood solutions to improve the accuracy of the obtained solutions. Fourteen different case studies including 122 test problems are employed for analyzing the performance of the proposed WAITS. The results show that the proposed memetic algorithm finds near-optimal solutions with an acceptable computational time. WAITS is compared to several algorithms in the literature. The results show that the proposed algorithm outperforms similar algorithms in the literature.

Keywords

Quadratic assignment problem, whale algorithm, tabu search, local search, **hospital facility layout**, combinatorial problems, facilities.

1. Introduction

The QAP is a challenging problem with a wide range of applications in diverse fields including backboard wiring problem [1], hospital layout problem [2], campus planning problem, airport gate assignment [3], turbine runner in electricity generation [4], statistical analysis [5], travelling salesman problem, and optimal placing of letters on touchscreen devices [6]. The problem of locating hospital departments so as to minimize the total distance traveled by patients can be formulated as a Quadratic Assignment Problem. As such, several attempts have been made to solve this difficult problem that can be categorized into exact, heuristic, and meta-heuristic methods.

Exact methods suffer from high computational complexity such as dynamic programming, branch-and-bound, and cutting planes or combinations of among these methods. There is currently no solution for this problem that runs in polynomial time unless $P=NP$. The heuristic algorithms are problem dependent and require modifications when applying to different problems. Osman and Laporte [7] considered that the heuristic methods are procedures that are devoted to the search for the good quality solutions. The meta-heuristic algorithms are applied to a wide range of problems. Furthermore, many new meta-heuristic algorithms are launched to mimic the behavior of many species as insects, flowers, water, whales which leads to a new tsunami of meta-heuristic algorithms [8, 9]. Some of the recent algorithms use metaphors to give a new name to an improvement to an existing algorithm with little novelty. The algorithm chosen in this work, WOA, is a well-regarded algorithm with a unique inspiration and mathematical mode. It has been widely used in both science and industry with a large number of citations since proposal. In the following sub-sections, we will explore some of the sequential and parallel versions of the heuristic and meta-heuristic algorithms developed for tackling QAP. Firstly, the sequential algorithms are surveyed in Table 1. Many researchers began to develop parallel processing programs to overcome the computational complexity of QAS after a large number of works using sequential algorithms. The parallel algorithms are explored in Table 2.

Table 1. The sequential algorithms for QAP

Author	Year	reference	Algorithm	Summary
Pradeepmon et al.	2016	[10]	Discrete Particle Swarm Optimization (DPSO)	- Taguchi's robust design approach is used to select the best combination of parameters used in DPSO.
Pradeepmon et al.	2018	[11]	Modified DPSO	- In DPSO, four different positions for each particle were generated and the best of them was selected rather than updating the standard velocity and position.
Mamaghani and Meybodi	2012	[12]	Particle Swarm Optimization with Hill Climbing (PSO _{HC})	- PSO is incorporated into hill climbing method. - Smallest Position Value (SPV) maps continuous values into discrete ones. - A repair method is suggested to handle any violation to SPV rule.
Liu and Clerc	2007	[13]	Variable Neighborhood Particle Swarm Optimization (VNPSO)	- The representation of the standard velocity and position is converted from real vectors to fuzzy matrices.
Riffi et al.	2017	[14]	Discrete Bat Algorithm (DBA)	- BA is integrated with a 2-exchange neighborhood method and a modified uniform crossover to enhance the search strategy of BA.
Shukla	2015	[15]	BAT	- SPV rule is used. - The hill climbing approach is applied to the best bat through iterations.
Ahmed	2015	[16]	Improved Genetic Algorithm (IGA)	- GA is improved using a modified sequential constructive crossover (SCX).
Ahmed	2015	[17]	Multi-Parent Genetic	- GA employs three-parent crossover

			Algorithm (MPGA)	operation instead of two-parent crossover in SCX and the results have been improved.
Tosun	2014	[18]	GA based on order-1 crossover	- A new recombination operator for GA based on order-1 crossover algorithm is used.
Azarbonyad and Babazadeh	2014	[19]	Proposed genetic algorithm	- Two-point crossover and the roulette wheel are used.
Ahuja	2000	[20]	Greedy genetic algorithm	- The 2-exchange neighborhood method improves the quality of the population.
Ahmed	2018	[21]	Lexisearch and Genetic Algorithm (LSGA)	- Lexisearch can give an optimal and an exact solution. - Lexisearch is used to generate the initial population. - Starting with good quality solutions leads to faster convergence in GA.
Taillard	1991	[22]	Tabu Search (TS)	- TS has fewer parameters and less complexity compared to previous versions of TS.
James	2009	[23]	Diversified best-solution-found TS (DivTS)	- In DivTS, a multi-start TS method executes multiple times from different initial settings.
Misevicius	2012	[24]	Iterated Tabu Search (ITS)	- ITS is combined with a special procedure of mutation and local search.
Lim et al.	2016	[25]	Biogeography-Based Optimization with Tabu Search (BBOTS)	- Biogeography-based optimization algorithm is combined with TS by replacing the mutation operator with TS procedure.
Shylo	2017	[26]	Repeated Iterated Tabu Search (RITS)	- RITS reduced the number of computations used for the investigation of a neighborhood in $O(n^3)$.
Gambardella et al.	1999	[27]	Hybrid Ant colony System for QAP (HAS-QAP)	- Ant Colony System is hybridized with a local search.
Pardalos and Resende	1994	[28]	Greedy Randomized Adaptive Search Procedure (GRASP)	- GRASP produces good quality solutions.
Stützle and Dorigo	1999	[29]	AS-QAP, AS2-QAP, and ANTS-QAP	- These algorithms gave good results compared to robust TS and GRASP.
Duman et al.	2012	[30]	Migrating Birds Optimization (MBO)	- MBO is inspired from the v flight information of the migrating birds for tackling QAP.
Benlic and Hao	2015	[31]	population-based memetic algorithm (BMA)	- BMA was combined with a Breakout Local Search (BLS).
Abdel-Baset et al.	2017	[32]	Elite-Opposition Flower Pollination Algorithm (EOFPA)	- EOFPA uses chaotic Switching Probability Strategy (CSPS).
Xia and Zhou	2018	[33]	Ant Colony Optimization (ACO)	- ACO algorithm outperforms 2-exchange local search method.

Table 2. The parallel algorithms for QAP

Author	Year	reference	Algorithm	Summary
Munera et al.	2016	[34]	Extremal Optimization (EO-)	- The parallel version of extremal optimization procedure performs

			QAP)	better than the sequential one.
Chmiel and Szwed	2016	[35]	(Bee Algorithm) BA	<ul style="list-style-type: none"> - BA is implemented on CUDA platform. - Time performance is improved if large swarms were processed in parallel.
Aksan et al.	2017	[36]	BLS-OpenMP	<ul style="list-style-type: none"> - A parallel version of BLS is proposed using OpenMP. - Levenshtein Distance metric is used for checking the similarity of new starting points to previously explored QAP permutations.
Amudha and Shivakumar	2012	[37]	Harmony Search Integrated with Consultant Guided Search (HICGS)	<ul style="list-style-type: none"> - The harmony search is combined with the consultant guided search. - The hybridization obtained better results than using consultant guided search only.
Chakrapani and Skorin	1993	[38]	Parallel tabu (par tabu)	<ul style="list-style-type: none"> - Number of explored neighborhoods is restricted in par tabu.
Talbi et al.	2001	[39]	ANTabu	<ul style="list-style-type: none"> - The exploitation phase in the parallel model of the ant colonies algorithm is upheld by tabu local search.
James et al.	2009	[40]	Cooperative Parallel Tabu Search (CPTS)	<ul style="list-style-type: none"> - The parallel processing of TS reduced the computational time.
Brown et al.	1989	[41]	parallel genetic heuristic	<ul style="list-style-type: none"> - A parallel version of the genetic algorithm is suggested.
Mohammadi et al.	2015	[42]	Genetic Algorithm with CUDA	<ul style="list-style-type: none"> - A parallel version of GA is proposed based on Graphics Processing Unit (GPU).
Tosun	2015	[43]	Parallel Hybrid Algorithm (PHA)	<ul style="list-style-type: none"> - PHA can go through three stages using GA, diversification phase, and TS to obtain near-optimal solutions.
Cárdenas et al.	2017	[44]	Parallel Genetic Algorithm (PGA)	<ul style="list-style-type: none"> - An improved PGA based Grid uses a local search 2-opt on GPU.
Ng and Tran	2018	[50]	Water Flow Algorithm (WFA)	<ul style="list-style-type: none"> - The parallel computing strategy is integrated with WFA to reduce the computational time.

All the above works show the popularity of heuristics and meta-heuristics in solving QAP problems. Now the question is as to why we need new or improve technique for this problem. According to the No Free Lunch theorem in the field of optimization, there is and will be no algorithm to solve all optimization problems. As an optimization problem, QAP can be better solved by a new memetic or improved algorithm when considering a specific dataset. This motivated our attempt to develop a solution for QAP using the recently proposed Whale Optimization Algorithm (WOA).

The motive of this paper is to propose a method for tackling QAP and find **near-optimal solutions** to alleviate high computational cost as the main drawback of exact methods. The meta-heuristic algorithms can't guarantee to obtain the optimal solutions but they have an additional advantage over the exact methods which is obtaining **acceptable results within a reasonable time**. As such, an integrated version of the meta-heuristic whale optimization algorithm with tabu search (WAITS) is proposed. The tabu search is characterized by its ability to ameliorate the quality of solutions.

WAITS algorithm has been compared with other meta-heuristic algorithms and the experimental results have revealed that WAITS presented superior results compared to these meta-heuristic algorithms. Many comparisons and statistical analyses are done such as best, mean, worst, Standard Deviation (SD), time, average error, boxplots, and drawing charts. Also, we will deal with the problem of hospital facility layout as QAP to minimize the total distance traveled by the patients.

The rest of this paper is organized as follows. Section 2 introduces the problem definition of QAP and the transformation of the hospital facility layout problem into QAP. Section 3 provides the description of the standard whale algorithm. In section 4, the proposed algorithm (WAITS) is more investigated and elaborated. The numerical and comparison results are given in section 5. Finally, some conclusions are introduced and some future works are suggested in section 6.

2. Quadratic assignment problem

QAP is NP-hard problem which was first introduced by Koopmans and Beckmann in 1957 for assigning plants to locations. At first, we need to know what the problem of QAP means and what its objectives are. QAP can be defined as follows:

- Given two matrices of facilities (F) and locations (L). Each matrix has a size of $N * N$ where $N = 1, 2, \dots, n$.
- The flow or weight between each pair of facilities is denoted by f_{ij} which represents the flow from facility i to facility j .
- The distance between each pair of locations is denoted by d_{ij} which represents the distance from location i to location j .
- The objective is to assign n facilities to n locations with a minimum cost which can be computed as follows:

$$\text{Minimize: } \sum_{i=1}^n \sum_{j=1}^n f_{ij} d_{\pi(i)\rho\pi(j)} \quad (1)$$

- The assignment of facilities to locations is represented by a permutation π where $\pi(i)$ represents the location to which facility i is assigned.
- Accordingly. The cost is the individual product of assigning facility i to location $\pi(i)$ and assigning facility j to location $\pi(j)$.

Thus, QAP can be summarized as the problem of assigning a set of facilities to a set of locations regarding the distance between any two locations and the flows between any two facilities to minimize the cost. Locating the departments of the hospital can be modeled as QAP to reduce the average distance traveled by the patients among the departments. Building new hospitals needs very expensive costs. So, renovating the existing hospitals and optimizing the reallocation of the departments may lead to better patient service. Furthermore, the time consumed by each patient can be decreased and the efficiency of the hospital in serving a larger number of patients will increase. If we consider a simple example for locating four

departments (D1, D2, D3, D4) to four locations (1, 2, 3, 4) in a single-floor of a hospital as follows:

		D1	D2	D3	D4			1	2	3	4
F =	D1	0	1	1	0	L =	1	0	7	14	12
	D2	4	3	0	1		2	8	0	16	9
	D3	0	2	2	0		3	22	13	0	8
	D4	0	4	0	3		4	16	15	20	0

The flows between any two departments are given in matrix F and the distances between any two locations are given in matrix L. The next step is to assign the set of departments to the set of locations in the aim of minimizing the distance traveled by the patients among the departments. The hospital facility layout problem can be formulated as QAP [54-57].

3. The whale algorithm

The whale algorithm (WA) [45] is a meta-heuristic algorithm that simulates the hunting mechanism of humpback whales in nature. During the hunting process, two or more of the whales swim in a decreasing circle puffing the bubbles to surround the prey along a spiral-shaped path. Some of the whales try to push the prey towards the surface, but the others push the prey towards the net. WA is characterized by the following strategies: looking for the prey and bubble-net attacking. The pseudo code of WA is provided in Algorithm 1.

Algorithm 1. The standard WA

size = number of search agents in the population, *max_iter* = number of maximum iterations

Initialize a population of random search agents $x_i (i = 1, 2, \dots, size)$

Evaluate each search agent

B^* = the best search agent

While ($t < \text{max_iter}$)

for each search agent in the population

 Update WA parameters (a , A , C , L , and p)

if ($p < 0.5$)

if ($|A| < L$)

 Update the current search agent by $x^{t+1} = B^* - A \cdot D$

else

if ($|A| \geq L$)

 Select a random search agent (x_{rand})

 Update the current search agent by $x^{t+1} = x_{rand} - A \cdot D$

end if

else

if ($p \geq 0.5$)

 Update the current search agent by $x^{t+1} = D' \cdot e^{bl} \cdot \cos(2\pi l) + B^*$

end if

end for

 Evaluate the search agent

 Update B^* if there is a better solution in the population

```

     $t = t + 1$ 
end while
return  $B^*$ 

```

3.1 Bubble-net attacking strategy

In fact, whales have the ability to identify their prey and attack them. The best search agent (or whale) is determined by the one close to the target prey or the optimum. The remaining search agents update their positions in the direction of the best search agent. To simulate the strategy of encircling the prey, the following equations are used:

$$D = |C \cdot B^* - x^t| \quad (2)$$

$$x^{t+1} = B^* - A \cdot D \quad (3)$$

$$A = 2a \cdot r - a \quad (4)$$

$$C = 2 \cdot r \quad (5)$$

where B^* represents the best search agent found and $| |$ is the absolute value, x^t is the current search agent and the distance between the current search agent and the best search agent at iteration t is denoted by D , A is a random value in the range $[-a, a]$, a decreases from 2 to 0 to indicate that the new position of the search agent can be updated anywhere between the current position of the search agent and the position of the best search agent, and C is a coefficient. To mimic the behavior of the spiral-shaped path, it can be defined as:

$$x^{t+1} = D' \cdot e^{bl} \cdot \cos(2\pi l) + B^* \quad (6)$$

$$D' = |B^* - x^t| \quad (7)$$

Where B^* shows the position of prey (best solution found so-far) b is a constant that defines the shape of the logarithmic spiral, l is a random number in $[-1,1]$, and $\cdot$ is a pairwise multiplication (dot product).

The absolute value for the distance between the current search agent and the best search agent at t iteration is denoted by D' . Because of the collective intelligence of humpback whales during the hunting process, WA implements the two behaviors (shrinking circle and spiral movement) with equal probability of 0.5 as follows:

$$x^{t+1} = \begin{cases} B^* - A \cdot D & p < 0.5 \\ D' \cdot e^{bl} \cdot \cos(2\pi l) + B^* & p \geq 0.5 \end{cases} \quad (8)$$

This equation allows switching between the two position updating equation with 50% equal chance. As such, the algorithm constantly converge and diverge from the prey (best solution) in an attempt not to trap in local solutions.

3.2 Searching for the prey

A is used with random values greater than 1 or less than -1 to simulate the random searching for the prey. Choosing a random search agent (x_{rand}) supports the exploration of the search space. Choosing the best search agent as in the bubble-net strategy supports the exploitation. Searching for the prey can be expressed as:

$$D = |C \cdot x_{rand} - x^t| \quad (9)$$

$$x^{t+1} = x_{rand} - A \cdot D \quad (10)$$

Despite the merits of the whale algorithm, it might not be suitable for a particular set of problems as per the No Free Lunch theorem discussed above. Our initial investigation showed that this algorithm requires improvement in the accuracy of the solution obtained. In addition, the original version of WA is not suitable to form combinatorial problems and required modifications. This motivated us to propose a new algorithm that can treat with the combinatorial nature of QAP.

4. The proposed algorithm (WAITS)

The basic steps of the proposed algorithm will be addressed in the following subsections. The flow chart of WAITS algorithm is shown in [Figure 1](#). The details of each step are discussed in the following sub-sections.

Figure 1. Flowchart of WAITS algorithm

4.1 Initialization

The first step begins with initializing the population with random search agents or solutions. Each solution has a dimension equal to the number of locations or facilities of QAP (n). The solution represents a different permutation of locations in which the different set of facilities will be assigned. If we assume that we have a QAP problem with 10 facilities which needed to be assigned to 10 locations. The solution can be represented as shown in Figure 2.

Facility (index)	1	2	3	4	5	6	7	8	9	10
Location	5	4	8	7	9	10	3	6	2	1

Figure 2. Solution representation

It can be seen that facility 1 is assigned to location 5, facility 2 is assigned to location 4, facility 3 is assigned to location 8, and so on. WA deals with the continuous space in which the search agents are represented by real values. As QAP is considered a permutation discrete problem, the largest real value is used to map the real values into permutation sequence as in Figure 3.

1.5	0.99	2.25	3.2	8.9	7.1	0.46	14.3	8.4	5.5
			↓			↑			
8	9	7	6	2	4	10	1	3	5

Figure 3. Largest real value mapping

4.2 Evaluation

After initializing the population, the random solutions are evaluated. The best solution is determined which has the minimum cost. For the simple example of the hospital facility layout problem mentioned before in section 2, it can be solved as QAP. To evaluate a random solution found in the population such as (4, 2, 3, 1), we perform the following steps shown in Figure 4. This solution means that facility D1 is assigned to location 4, facility D2 is assigned to location 2, facility D3 is assigned to location 3, and facility D4 is assigned to location 1. As can be seen in the figure the cost of this solution is 148. After evaluating all the search agents in the population, the best search agent is selected. WA will be applied for each search agent in the population through a number of iterations to ameliorate the quality of solutions.

Figure 4. Evaluation of QAP

4.3 Tabu search

WA improves the quality of the search agents through iterations. Per iteration, a new search agent is generated and it is replaced with the one in the population if it is better. Also, the best search agent is updated. TS is employed for better and more effective improvement in the quality of solutions. We perform a modification on a version of TS developed by Taillard [22] as follows:

- Our TS procedure is not applied for a specific number of iterations, but it is combined in the WA.
- The new search agent from WA is first compared with the current solution in the population and it is replaced with the one in the population if it is better.
- After that, we can apply TS procedure on the current solution in the population.

The swap operator is computationally expensive, so to generate neighborhood search agents for a specific search agent, TS is supported with a local search. Suppose we have a current search agent with a specified permutation (π), if we want to generate a neighborhood search agent the new neighborhood, permutation is denoted by ($\emptyset$). The difference cost for any move of two locations (j, k) in the current permutation π if the matrices (locations and facilities) are symmetric is computed as follows:

$$\Delta_{cost}(\pi, j, k) = \sum_{j=1}^N \sum_{k=1}^N (f_{jk} d_{\pi(j)\pi(k)} - f_{jk} d_{\emptyset(j)\emptyset(k)}) \quad (11)$$

If the matrices are not symmetric the difference cost can be computed as:

$$\Delta_{cost}(\emptyset, a, b) = \Delta(\pi, a, b) + 2 \cdot (f_{ja} - f_{jb} + f_{kb} + f_{ks}) (d_{\pi(k)\pi(a)} - d_{\pi(k)\pi(b)} + d_{\pi(j)\pi(b)} - d_{\pi(j)\pi(a)}) \quad (12)$$

The selected moves are tabu or forbidden for a random number of iterations proportional to the QAP size. The aspiration criterion allows us to accept a move that may be forbidden but the cost of that move is more interesting. The detailed pseudo code of the WAITS algorithm is given in [Algorithm 2](#).

5. Computational results and discussion

This section is dedicated for demonstrating the effectiveness and the strength of the proposed algorithm. The technical specification of the hardware used to conduct experiments is windows 7 Ultimate 64-bit operating system; 4 GB of RAM; processor Intel Core i5-3317U CPU, 1.70 GHz processor speed. The algorithm is implemented in JAVA platform.

The proposed algorithm (WAITS) was run on 14 different instances which consist of 122 test problems known as follows Chr, Els, Esc, Had, Kra, Lipa, Nug, Rou, Scr, Sko, Ste, Tai, Tho, and Wil. The size of QAP ranges from 12 to 256. The dataset is available online [58]. One advantage of WAITS algorithm is having few parameters to set their values. Setting the algorithm parameters may affect the quality of the generated solutions. Several experiments were done to find the best combination of the parameters values that can give competitive results. Only one test problem is selected from each instance to ensure the efficient selection of the parameters values. Finally, the values of the parameters of WAITS algorithm are set as shown in [Table 3](#). Note that the same parameters are used for other algorithms too (particularly the population size and iteration) to ensure fair comparisons. This number of population size and iteration in [Table 3](#) are obtained after an extensive experiment with different values. We observed that all algorithms show their best performance with such values.

Table 3. The parameters of WAITS

Parameter	Value
Population size	20
Number of iterations	5000
Number of evaluations	30
b	1

Algorithm 2. WAITS algorithm

```

size = number of search agents in the population
max_iter = number of maximum iterations
Initialize a population of random search agents  $x_i (i = 1, 2, \dots, size)$ 
Cost = Evaluate each search agent
 $B^*$  = the best search agent
Define a matrix Tabu_list with size  $N \times N$ 
Define a matrix  $\Delta_{cost}$  with size  $N \times N$ 
Define  $min\_size = (3 * N) / 10$ 
Define  $max\_size = (7 * N) / 10$ 
While ( $t < max\_iter$ )
    for each search agent  $x_i$  in the population
        Update WA parameters ( $a, A, C, L$ , and  $p$ )
        if ( $p < 0.5$ )
            if ( $|A| < L$ )
                Update the current search agent by  $x^{t+1} = B^* - A.D$ 
            else
                if ( $|A| \geq L$ )
                    Select a random search agent ( $x_{rand}$ )
                    Update the current search agent by  $x^{t+1} = x_{rand} - A.D$ 
                end if
            end if
        else
            if ( $p \geq 0.5$ )
                Update the current search agent by  $x^{t+1} = D'.e^{bl}.\cos(2\pi l) + B^*$ 
            end if
            if there are any repeated values in the same search agent
                Replace it with a random value that isn't found in the search agent
             $x_{seq}$  = apply the largest real value mapping on the new search agent  $x^{t+1}$ 
            Evaluate the new search agent  $x_{seq}$ 
            if ( $cost(x_{seq}) < cost(x_i)$ )
                 $x_i = x_{seq}$ 
            Generate a set of neighborhood solutions to  $x_i$  using  $\Delta_{cost}$ 
             $\Delta_{cost}$  = store the difference cost of swapping any two locations ( $j, k$ ) in the sequence  $x_i$ 
            Accept the move ( $j, k$ ) with minimum  $\Delta_{cost}$  if it isn't forbidden ||
            accept forbidden move ( $j, k$ ) if it obeys the aspiration criterion
             $x_{swap}$  = swap two locations ( $j, k$ )
             $x_i = x_{swap}$ 
             $Cost(x_i)$  = Evaluate the new search agent  $x_i$ 
            Forbid the chosen move ( $j, k$ ) for a specific iterations  $\in [min\_size, max\_size]$ 
            Update  $B^*$  if  $x_i$  is better
        end for
    t = t + 1
end while
return  $B^*$ 

```

In The first experiment, we will show the effect of integrating the local search and the tabu search with the whale algorithm. Table 4 records the computational results for six test problems originated from different six instances among the three variants of WA as follows:

1. The standard whale algorithm (WA).
2. The whale algorithm combined with local search (WAL).
3. The whale algorithm combined with local search and tabu search (WAITS).

The parameters values of the three algorithms are set as recorded in Table 3. It can be seen that WAITS algorithm outperforms the other two algorithms in all the test problems. Also, WAL outperforms WA in four problems. We can conclude that integrating the local search and the tabu search improves the results obtained by WAITS algorithm. Figs. 5-18 show the boxplots for the test problems Chr12a, Els19, Esc64a, Had20, Kra30a, Lipa80b, Nug27, Rou20, Scr20, Sko100b, Ste36b, Tai64c, Tho30, and Wil50 respectively. A random test problem is selected from each instance. The boxplots draw the difference in WA, WAL, and WAITS algorithms regarding the results for 20 independent runs. It can be deduced that WAITS algorithm has lower values compared to WA and WAL.

Table 4. Comparison among variants of the whale algorithm

Problem	Best-known	WA	WAL	WAITS
Chr12a	9552	11248	9916	9552
Els19	17212548	24642284	17212548	17212548
Esc64a	116	116	116	116
Had20	6922	6922	6922	6922
Sko100b	153890	156884	155522	153890
Tai64c	1855928	1857646	1857646	1855928

Bold font indicates the best results

Figure 5. Boxplot for Chr12a problem

Figure 6. Boxplot for Els19 problem

Figure 7. Boxplot for Esc19a problem

Figure 8. Boxplot for Had20 problem

Figure 9. Boxplot for Kra30a problem

Figure 10. Boxplot for Lipa80b problem

Figure 11. Boxplot for Nug27 problem

Figure 12. Boxplot for Rou20 problem

Figure 13. Boxplot for Scr20 problem

Figure 14. Boxplot for Sko100b problem

Figure 15. Boxplot for Ste36b problem

Figure 16. Boxplot for Tai64c problem

Figure 17. Boxplot for Tho30 problem

Figure 18. Boxplot for Wil50 problem

A non-parametric Friedman test of differences in the data is conducted. In Table 5, the Friedman test compares the mean ranks between the algorithms and indicates how the algorithms differ. Table 6 presents the actual results of the Friedman test and whether there was an overall statistically significant difference between the mean ranks of the algorithms. N is the number of evaluations for each algorithm. df is the number of the algorithms minus one. The Friedman test renders a Chi-Square of 18.200 which was significant ($p < 0.01$).

Table 5. The mean ranks of algorithms

Algorithms	Mean Rank
WA	3.00
WAL	1.90
WAITS	1.10

Table 6. Test statistics

N	10
Chi-Square	18.200
df	2
Asymp. Sig.	0.00

Table 7 summarizes the simulated results obtained by WAITS algorithm for running 122 test problems. The best, average, and worst results during 30 independent runs are given in this Table. The Time and the standard deviation between data are computed and presented as well. In addition to the Average Error (AE) which can be defined as follows:

$$AE = \frac{1}{n} \times \sum_{i=0}^n \frac{y - opt}{opt} \times 100 \quad (13)$$

where n represents the QAP size of locations or facilities. opt is the best-known solution obtained for the problem, and y is the result obtained for QAP in the best or average or worst case. The proposed algorithm can get the optimal results in 86 test problems out of 122 test problems.

Table 7. The simulated results obtained by WAITS algorithm on QAP instances

No.	Problem	Best-known	Best	Average	Worst	AE _{best}	AE _{avg}	Time	SD
1	Chr12a	9552	9552	9552	9552	0	0	3.12	0
2	Chr12b	9742	9742	9742	9742	0	0	3.04	0
3	Chr12c	11156	11156	11156	11156	0	0	3.26	0
4	Chr15a	9896	9896	9908.1	9978	0	0.12	4.04	34.6
5	Chr15b	7990	7990	8024.8	8228	0	0.43	3.95	85.06
6	Chr18a	11098	11098	11239.3	12239	0	1.21	4.24	292.1
1	Chr18b	1534	1534	1534	1534	0	0	4.36	0
8	Chr20a	2192	2192	2263.6	2422	0	3.26	5.82	90.87
9	Chr20b	2298	2334	2461	2522	1.56	1.09	6.52	55.89
10	Chr20c	14142	14142	14480.52	14812	0	2.39	9.40	357.6
11	Chr22a	6156	6166	6253.93	6316	0.16	1.59	12.94	67.41
12	Chr22b	6194	6194	6330.2	6394	0	2.19	13.80	91.38
13	Chr25a	3796	3796	4126.33	4380	0	8.10	17.64	261.9
14	Els19	17212548	17212548	17212548	17212548	0	0	7.30	0
15	Esc16a	68	68	68	68	0	0	3.5	0
16	Esc16b	292	292	292	292	0	0	3.4	0
11	Esc16c	160	160	160	160	0	0	3.38	0
18	Esc16d	16	16	16	16	0	0	3.29	0
19	Esc16e	28	28	28	28	0	0	3.39	0
20	Esc16f	0	0	0	0	0	0	3.6	0
21	Esc16g	26	26	26	26	0	0	3.19	0
22	Esc16h	996	996	996	996	0	0	3.27	0
23	Esc16i	14	14	14	14	0	0	3.18	0
24	Esc16j	8	8	8	8	0	0	3.8	0
25	Esc32a	130	130	130	130	0	0	19.36	0
26	Esc32b	168	168	168	168	0	0	19.71	0
21	Esc32c	642	642	642	642	0	0	20.4	0
28	Esc32d	200	200	200	200	0	0	20.42	0
29	Esc32e	2	2	2	2	0	0	18.6	0
30	Esc32f	2	2	2	2	0	0	20.19	0
31	Esc32g	6	6	6	6	0	0	21.4	0

32	Esc32h	438	438	438	438	0	0	19.64	0
33	Esc64a	116	116	116	116	0	0	38.9	0
34	Esc128a	64	64	64	64	0	0	147.53	0
35	Had12	1652	1652	1652	1652	0	0	4.93	0
36	Had14	2724	2724	2724	2724	0	0	5.7	0
31	Had16	3720	3720	3720	3720	0	0	3720	0
38	Had18	5358	5358	5358	5358	0	0	6.98	0
39	Had20	6922	6922	6922	6922	0	0	8.07	0
40	Kra30a	88900	88900	88900	88900	0	0	17.1	0
41	Kra30b	91420	91420	91496	91590	0	0	15.43	20.15
42	Lipa20a	3683	3683	3683	3683	0	0	8.13	0
43	Lipa20b	27076	27076	27076	27076	0	0	14.17	0
44	Lipa30a	131178	131178	131178	131178	0	0	24	0
45	Lipa30b	151426	151426	151426	151426	0	0	24.8	0
46	Lipa40a	31538	31538	31538	31538	0	0	52.23	0
41	Lipa40b	476581	476581	476581	476581	0	0	59.62	0
48	Lipa50a	62093	62093	62093	62093	0	0	120.70	0
49	Lipa50b	1210244	1210244	1210244	1210244	0	0	116.91	0
50	Lipa60a	107218	107218	107453	108020	0	0	167.77	379.9
51	Lipa60b	2520135	2520135	2520135	2520135	0	0	177.16	0
52	Lipa70a	169755	169755	170396	170830	0	0	277.32	585.4
53	Lipa70b	4603200	4603200	4603200	4603200	0	0	270.74	0
54	Lipa80a	253195	254590	254632.6	254693	0	0	533.87	48.73
55	Lipa80b	7763962	7763962	7763962	7763962	0	0	513.42	0
56	Lipa90a	360630	362468	362554.8	362606	0	0	725.03	51.76
51	Lipa90b	12490441	12490441	12490441	12490441	0	0	729.41	0
58	Nug12	578	578	578	578	0	0	9.31	0
59	Nug14	1014	1014	1014	1014	0	0	10.29	0
61	Nug16a	1150	1150	1150	1150	0	0	10.66	0
62	Nug16b	1610	1610	1610	1610	0	0	11.76	0
63	Nug17	1240	1240	1240	1240	0	0	11.30	0
64	Nug18	1930	1930	1930	1930	0	0	12.30	0
65	Nug20	2570	2570	2570	2570	0	0	16.28	0
66	Nug21	2438	2438	2438	2438	0	0	17.8	0
61	Nug22	3596	3596	3596	3596	0	0	18.67	0
68	Nug24	3488	3488	3488	3488	0	0	20	0
69	Nug25	3744	3744	3744	3744	0	0	20.31	0
10	Nug27	5234	5234	5234	5234	0	0	21.81	0
11	Nug28	5166	5166	5170	5178	0	0	35.96	4.89
12	Nug30	6124	6156	6168.4	6180	0	0	56.68	19.91
13	Rou12	235528	235528	235528	235528	0	0	5.56	0
14	Rou15	354210	354210	354210	354210	0	0	6.17	0
15	Rou20	725522	725522	725522	725522	0	0	7.76	0
16	Scr12	31410	31410	31410	31410	0	0	3.51	0
11	Scr15	51140	51140	51140	51140	0	0	4.75	0
18	Scr20	110030	110030	110030	110030	0	0	8	0
19	Sko42	158120	15812	15828	15840	0	0	1202	1095
80	Sko49	23386	23418	23429.2	23438	0	0	32116	122
81	Sko56	34458	34488	34544.4	34588	0	0	15844	3093
82	Sko64	48498	48534	48580.4	48642	0	0	33141	2523
83	Sko72	66256	66438	66620	66516	0	0	41484	5055
84	Sko81	90998	91118	91265.2	91410	0	0	45586	5668
85	Sko90	115534	116192	116382.57	116686	0	0	56162	163.12
86	Sko100a	152002	153112	153313	153454	0	0	154	91.95
81	Sko100b	153890	155030	155455.631	155148	0	0	121	24181
88	Sko100c	147862	149336	149547.5	149784	0	0	113	165.67
89	Sko100d	149576	151048	151216.9	151446	0	0	696	135.63
90	Sko100e	149150	150294	150659.5	150978	0	0	102	206.81

91	Sko100f	149036	150458	150704.5	150968	0.95	1111	118	158.84
92	Ste36a	9526	9548	9581.6	9616	0.23	0.58	48.8	30.41
92	Ste36b	15852	15852	15914	16024	0	0.89	47.60	77.91
93	Ste36c	8239110	8250568	8374000	8480794	0.113	1.63	46.9	87130
94	Tai12a	224416	224416	224416	224416	0	0	6.48	0
95	Tai12b	39464925	39464925	39464925	39464925	0	0	5.58	0
96	Tai15a	388214	388214	388224.8	388250	0	0.002	5.89	17.39
91	Tai15b	51765268	51765268	51765268	51765268	0	0	6.12	0
98	Tai20a	703482	703482	105231.6	106186	0	0.24	18.89	1490.43
99	Tai20b	122455319	122455319	122455319	122455319	0	0	16.08	0
100	Tai25a	1167256	1167256	1174727	1181204	0	0.64	31.30	5249.9
101	Tai25b	344355646	344355646	344355646	344355646	0	0	28.76	0
102	Tai30a	1818146	1826926	1834200.33	1842214	0.48	0.88	29.76	5477.23
103	Tai30b	637117113	637117113	638158809.88	639845990	0	0.16	33.45	14400.02
104	Tai35a	2422002	2423628	2437416	2444738	0.06	0.63	35.11	7221.06
105	Tai35b	283315445	283315445	283700611.2	284199488	0	0.13	34.56	220059.52
106	Tai40a	3139370	3155844	3168510.8	3181422	0.52	0.92	48.88	8085.78
101	Tai40b	637250948	637284966	638294902.11	641475835	0.005	0.16	48.95	1370097.23
108	Tai50a	4938196	5016414	5025226.33	5030820	1.57	1.75	204	10443.94
109	Tai50b	458821517	459089187	460223840.8	462874956	0.05	0.80	210.16	1416053.55
110	Tai60a	7208572	1348090	181314604	1403160	1.93	2.30	584	21800.59
111	Tai60b	608215054	612141314	618191133.8	622111525	0.74	1.13	521	1416053.55
112	Tai64c	1855928	1855928	1855928	1855928	0	0	332.45	0
113	Tai80a	13551864	13816614	13869196.2	13925632	1.90	2.29	1157	29977.56
114	Tai80b	818415043	828283416	842267355.3	85138581	1.20	2.91	1098	7239055.91
115	Tai100a	21125314	21556849	21600484.25	21638366	2.04	2.24	1201	28993.77
116	Tai100b	1185996131	1191957535	1194316026.16	1198296313	0.50	0.70	1241	2210646.78
111	Tai150b	498896643	507703322	509498025.71	510750347	1.16	2.12	1346	1099949.24
118	Tai256c	44759294	44879056	44893914.66	44901834	0.26	0.30	1560	10514.27
119	Tho30	149936	149936	150133.6	150416	0	0.13	23.01	223.08
120	Tho40	240516	240542	240831	241254	0.01	0.13	681.83	252.22
121	Wil50	48816	48848	48884.6	48906	0.06	0.14	121.58	21.02
122	Wil100	213038	213938	274482.4	274806	0.82	0.52	1256	311.65

The next step is to prove the effectiveness and outperformance of the proposed algorithm compared to the other existing algorithms. The most powerful and recent algorithms are selected for comparison. Table 8 provides a comparison on Taillard instance between WAITS algorithm and the following algorithms:

- Discrete Bat Algorithm (DBA) [14].
- Improved Genetic Algorithm (IGA) [16].
- Simple Genetic Algorithm (SGA) [16].
- Hybrid Ant System for Quadratic Assignment Problem (HAS-QAP) [27].
- Genetic algorithm using Sequential Constructive Crossover (SCX) [46].

It can be evidently seen that WAITS outperforms the other algorithms in the majority of the instances. Note that the bold font indicates the best result among other algorithms. In addition to Figure 19 presents a comparison based on the average error

for the average results (AE_{avg}). Our proposed algorithm has the lowest average error with a value of 0.94%.

Table 8 . Comparison among algorithm on Taillard instance

Problem	HAS-QAP[27]	IGA[16]	SCX[46]	DBA[14]	SGA[16]	WAITS
Tai20a	0.675	2.70	4.50	0.85	4.99	42.0
Tai20b	0.091	0.85	6.56	0.0	6.59	42.0
Tai25a	1.189	3.34	4.58	1.51	4.17	42.0
Tai25b	0.0	0.39	5.24	0.0	6.21	42.0
Tai30a	1.311	2.65	4.29	0.72	4.36	0.88
Tai30b	0.0	1.03	8.78	0.0	3.41	0.16
Tai35a	1.762	3.60	4.95	1.79	4.49	42.0
Tai35b	0.026	1.34	5.00	0.19	6.43	0.13
Tai40a	1.989	3.58	4.53	2.02	4.71	42.0
Tai40b	0.0	2.74	7.30	0.0	6.60	0.16
Tai50a	2.800	3.58	4.51	2.69	4.69	52.0
Tai50b	0.192	0.61	5.60	0.11	4.14	0.30
Tai60a	3.070	3.35	4.54	2.73	4.60	2.4
Tai60b	0.048	1.42	5.12	0.09	4.81	1.73
Tai80a	2.689	3.29	4.35	2.67	4.63	2.0
Tai80b	0.667	2.89	6.63	0.56	5.81	2.91

Bold font indicates the best results

Figure 19. AE for Taillard instance among algorithms

In Table 9, a comparison among WAITS, ANT-QAP [47], and BA-QAP [47] is performed. WAITS algorithm achieves the best results in most of the test problems. Figure 20 shows the average error on these problems among the previous algorithms and another algorithm called Extremal Optimization for QAP (EO-QAP) [34]. WAITS obtains the lowest AE with a value equal to 0.35. This proves that WAITS algorithm is an effective algorithm for solving the QAP.

Table 9. Comparison among algorithms

Problem	ANT-QAP[47]	BA-QAP[47]	WAITS
Chr12a	9552	9552	9552
Chr20a	2464	2444	2192
Els19	17257786	17212548	17212548
Esc32a	134	134	130
Esc64a	132	116	116
Esc128	78	64	64
Had20	6922	6922	6922

Lipa50a	62773	62746	62093
Lipa90	363141	363060	362468
Nug20	2570	2570	2570
Nug30	6124	6124	6124
Rou20	725522	725522	725522
Scr20	110030	110030	110030
Sko42	15940	15940	15812
Sko90	116572	116418	116192
Sko100a	153148	153274	153148
Ste36a	9790	9776	9548
Tai30a	1861488	1861488	1826926
Tai50a	5078694	5063482	5016474
Tai64c	1855928	1855928	1855928
Tai100a	2152146	21624756	21556849
Tai100b	1416812373	1339342571	1191957535
Tai150b	539508841	510917010	507703322
Tai256c	44914802	44895140	44879056
Tho40	240516	240516	149936
Wil50	49002	48994	48848
Wil100	274466	274308	273938

Bold font indicates the best results

Figure 20. Comparison on AEbest for QAP

Table 10 presents a comparison between WAITS and the following algorithms:

- Simulated annealing (SA) [12].
- Particle swarm Optimization (PSOPure) [12].
- Particle Swarm optimization with Variable Neighborhood Search (PSOVNS) [12].
- Genetic Algorithm (OB-GA) [12].

The best results obtained from the algorithms are compared to each other for the instances Tai and Sko. WAITS achieves superiority and precedence in most of the instances. We can conclude from the results that WAITS algorithm outperforms the other algorithms significantly in 12 test problems out of 15 test problems.

Table 10. Comparison among algorithms on QAP

Problem	PSO _{pure} [12]	PSO _{VNS} [12]	SA [12]	OB-GA [12]	WAITS
Tai20a	704190	703482	703482	703482	703482
Tai30a	1868871	1866122	1849696	1863722	1826926
Tai40a	3233491	3225382	3212692	3215382	3155844
Tai50a	5088574	5043551	5041058	5067904	145. 000
Tai60a	7438398	7431459	7381442	7422318	0. 084. 4
Tai80a	13831255	13812280	13852100	13814562	13816614
Tai100a	21603991	21602881	21499478	21501554	21556849
Tai150b	508956543	506448890	502651565	506448890	507703322
Tai256c	44819486	44809949	44814014	45023121	44809949
Sko100a	155462	153949	154210	153090	153090
Sko100b	156007	155971	154262	155030	155030
Sko100c	150978	149366	149542	149948	149336
Sko100d	152346	151746	151746	150828	150828
Sko100e	152349	150999	150426	150598	150294
Sko100f	152038	150871	150738	150402	150402

Bold font indicates the best results

Figure 21 and Figure22 draw a comparison on some problems of QAP based on the average error for the best and average results respectively. WAITS is compared with the following:

- Genetic Algorithm (GA) [49].
- Simulated Annealing (SA) [49].
- Tabu search (TS) [49].
- Multi-Parent Genetic algorithm (MPGA) [17].

Among the algorithms, WAITS has the lowest average error as can be shown in figures. Table 11 presents the computational results of WAITS compared to a parallel genetic algorithm [42]. WAITS is able to obtain the optimal results in four test problems. Furthermore, the parallel genetic algorithm is able to obtain the optimal result in only one test problem. It can be seen that WAITS can effectively solve QAP rather than the parallel genetic algorithm. Table 12 summarizes the computational results of DPSO [11], PSO [58], ACO-PSO [58], BBOTS [25], and WAITS. The algorithms are evaluated using three performance measures: the average error for the best results, the average results, and the worst results. WAITS gets the optimal solutions in all the test problems except for Nug30 while ACO-PSO can't obtain the optimal solutions in four test problems. Once again, WAITS is superior when solving most of the test problems.

Figure 21. Comparison among algorithms for QAP

Figure 22. Comparison based on Average error for the average results

Table 11. Comparing the performance results of GA and WAITS algorithms

Problem	GA with CUDA [42]	WAITS
Chr12a	0	0
Chr15a	1.14	4
Chr18a	4159	4
Tai25b	0109	4
Tai30b	0111	425
Tai35b	01489	425
Tai40b	1156	425
Tai60b	2161	521
Tai80b	4.50	2.91
Tai100b	4.07	4214

Bold font indicates the best results

Table 12. The computational results obtained by DPSO, ACO-PSO, PSO, BBOTS, and WAITS

Problem	DPSO [11]			ACO-PSO [58]	PSO [58]	BBOTS [25]	WAITS		
	AE _{best}	AE _{avg}	AE _{worst}	AE _{best}	AE _{best}	AE _{avg}	AE _{best}	AE _{avg}	AE _{worst}
Had12	0	0B15	1453	048	4	4	4	4	4
Had14	0	01191	0808	4	4	4	4	4	4
Had16	0	0086	0538	4	4	4	4	4	4
Had18	0.075	01158	2202	4	4	4	4	4	4
Had20	0	0B64	2080	0111	4	4	4	4	4
Kra30a	2.542	6864	9224	0	1112	01090	4	4	4
Kra30b	1.903	3915	6601	008	032	01060	4	0108	428
Nug12	0.692	2491	4.498	4	4	4	4	4	4
Nug14	0.197	3.511	5.720	4	4	4	4	4	4
Nug16a	1.242	4.199	8.323	4	4	4	4	4	4
Nug16b	0	4.129	7.903	4	4	4	4	4	4
Nug17	1.039	2.633	4.619	4	4	01012	4	4	4
Nug18	0.311	3.140	5.285	4	4	4	4	4	4
Nug20	0	3.315	5.759	4	4	4	4	4	4
Nug21	0.246	3.224	6.399	4	4	4	4	4	4
Nug22	0.167	2.536	5.339	4	4	4	4	4	4
Nug24	2.580	4.616	7.225	4	4	4	4	4	4
Nug25	0.588	2.917	4.701	4	4	4	4	4	4
Nug27	1.146	3.607	4.929	4	4	4	4	4	4
Nug28	1.200	4.278	6.891	0111	0108	01209	4	4240	42 .
Nug30	1.339	3.011	5.062	0B9	425.	424. 1	0152	0112	42 5
Rou12	1.915	4.412	6.652	1.88	0	0	0	4	4
Rou15	2.780	5.017	6.214	3.84	0	0	0	4	4
Rou20	1.837	4.268	5.988	0.19	0	0.062	0	4	4

Bold font indicates the best results

In Table 13, we have compared the results of two instances (Chr, Lipa) obtained by WAITS algorithm with those obtained by the following algorithms:

- Greedy Genetic Algorithm (GGA) [20].
- Parallel Genetic Algorithm (PGA) [44].
- Water Flow Algorithm (WFA) [50].
- Iterated Fast Local Search (IFLS) [51].
- Particle Swarm Optimization (PSO) [58].
- Ant Colony Optimization inspired PSO (ACO-PSO) [58].

WAITS algorithm gets the best results in 28 test problems out of 30 test problems. When comparing the average error of the best results in Figure 23, WAITS algorithm dominates most of the results obtained by IFLS and WFA for the instances Chr, Lipa, Sko, Tai, Wil, and Tho. Figure 24 presents the average time spent by each algorithm for solving each of the previous instances. WAITS algorithm saves much time in solving QAP.

Table 13. Comparison among IFLS, PGA, GGA, WFA, PSO, ACO-PSO, and WAITS respectively

Problem	IFLS [51]	PGA [44]	GGA [20]	WFA [50]	PSO [58]	ACO-PSO[58]	WAITS
Chr12a	4	4	4	4	4	5B9	4
Chr12b	4	4	4	4	4	4	4
Chr12c	4	4	4	4	0D1	2D6	4
Chr15a	4	4	0H	4	0H0	0B2	4
Chr15b	4	4	4	4	4	2K8	4
Chr18a	4	4	4	4	0B8	4	4
Chr18b	5114	4	0H	4	3118	4H1	4
Chr20a	4	4	4	4	4	4	4
Chr20b	4.38	0.18	0	0	1.79	4.36	1B6
Chr20c	5H0	3112	5113	4	4	4	4
Chr22a	4	4B1	4	4	5K9	4	0116
Chr22b	0B8	4	0115	4	0B2	4	4
Chr25a	1K8	1H6	0	4	4	4	4
Chr25b	1111	2D1	0	4	5D5	10D6	4
Lipa20a	4	4	4	4	-	-	4
Lipa20b	4	4	4	4	-	-	4
Lipa30a	4	4	4	4	-	-	4
Lipa30b	4	4	4	4	-	-	4
Lipa40a	4	4	0B6	4	-	-	4
Lipa40b	4	4	4	4	-	-	4
Lipa50a	1D2	0B2	0B5	0B0	-	-	4
Lipa50b	4	4	4	4	-	-	4
Lipa60a	0B4	0K4	011	0.79	-	-	4
Lipa60b	4	4	4	4	-	-	4
Lipa70a	011	0K2	011	011	-	-	4
Lipa70b	4	1B9	4	4	-	-	4
Lipa80a	0K1	0K1	0K1	0K3	-	-	4B1
Lipa80b	20B3	16B6	4	4	-	-	4
Lipa90a	0K3	0B4	0B8	0B1	-	-	4B4
Lipa90b	4	4	4	4	-	-	4

Bold font indicates the best results

Figure 23. A comparison among IFLS, WFA, WAITS

Figure 24. A comparison on Average time

Table 14 reports the average error for the best, average, worst results among LSGA [21], FIPS [52], UPSO [52], and WAITS algorithms. Our proposed algorithm can obtain the best results in most of the test problems. The average for the test problems in AE_{best} , AE_{avg} , and AE_{worst} is computed. For the average of AE_{best} , WAITS algorithm ranked first with a value of 0.23% and at the same time, UPSO algorithm came in the last place with a value 1.71%. Also for the average of AE_{avg} and AE_{worst} , The proposed algorithm WAITS obtained the precedence compared to LSGA with values 0.37% and 0.61% respectively.

Table 14. Comparison among LSGA, FIPS, UPSO, and WAITS algorithms

Problem	LSGA [21]			FIPS [52]		UPSO [52]		WAITS		
	AE_{best}	AE_{avg}	AE_{worst}	AE_{best}	AE_{avg}	AE_{best}	AE_{avg}	AE_{best}	AE_{avg}	AE_{worst}
Tai20a	0.30	0.61	1.18	520.	284	4	420	420	420	420
Tai20b	0	4	4	4	520	4	4	4	4	4
Tai30a	0.48	1.23	1.16	210	214	4208	4288	52.		
Tai30b	0	4	4	520.	25.	4	0.116	0.143		
Tai40a	1.06	1.52	2.13	248	204	421.	42.	520		
Tai40b	0	4	425	0.36	2.55	0.005	0.116	0.166		
Tai50a	1.62	2.01	2.53	0.41	0.45	520	520	528.		
Tai50b	0	420	420	1.10	1.34	0.05	0.30	0.88		
Sko42	0	0.19	0.42	42.	420	4	4254	420		
Sko49	0.14	0.25	0.54	52.5	420	425.	4258	42.		
Sko81	0.10	0.30	0.53	2.01	1.12	0.119	42.	4201		
Sko100a	0.26	0.54	0.39	2.36	1.66	0.116	0.86	0.95		
Chr15a	0	0.23	0.83	4	4	4	425.	428.		
Kra30a	0	0.79	1.57	2.	2.1	4	4	4		
Kra30b	0	0.07	0.25	425.	420.	4	0.108	4258		
Had20	0	0	0	424.	424.	4	4	4		
Average	0.25	0.5	0.78	521.	5205	42.	420	425		

Taken together, the results of this section showed that the proposed mechanism can improve the performance of the whale algorithm significantly. The modifications were beneficial since it was observed that the ordinal whale algorithm does not find

accurate solutions and suffer from low exploitation when solving QAP problems. The reason for better results of WAITS is due to the improved local search using TS.

6. Conclusions and future works

This paper proposed a whale algorithm integrated with tabu search for solving QAP. A local search strategy was used to determine the cost of moves of any two locations. It was discussed that the standard whale algorithm suffers from the low accuracy of the solutions obtained for QAP problems. The proposed methods alleviate these drawbacks. Furthermore, we have described the hospital facility layout as QAP. The results show that the proposed WAITS obtains near-optimal results in acceptable time, is well-suited for QAP, and outperforms the current algorithms in the literature when solving QAPs.

For future works, we hope to apply the proposed algorithm for real-world applications such as traveling salesman problem, scheduling, knapsack problem, and cryptography. We have discussed a simple example of this problem in a single-floor. In the real life, In fact, the hospitals have become more complex, where they contain many departments that are distributed to multiple buildings and each building may contain multiple floors. We want to apply the proposed algorithm for solving the problem of locating the hospital departments in practice. Another future direction is to implement a parallel version of WAITS to benefit from the power of parallel processing. Most of the computer devices are designed to support the multi-processing of the tasks. The parallel processing of tasks can make a tradeoff between the large size of QAP and the time consumed for solving it.

Competing Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

- [1] Steinberg, L. (1961). The backboard wiring problem: A placement algorithm. *Siam Review*, 3(1), 37-50.
- [2] Helber, S., Böhme, D., Oucherif, F., Lagershausen, S., & Kasper, S. (2016). A hierarchical facility layout planning approach for large and complex hospitals. *Flexible Services and Manufacturing Journal*, 28(1-2), 5-29.
- [3] Haghani, A., & Chen, M. C. (1998). Optimizing gate assignments at airport terminals. *Transportation Research Part A: Policy and Practice*, 32(6), 437-454.
- [4] Laporte, G., & Mercure, H. (1988). Balancing hydraulic turbine runners: A quadratic assignment problem. *European Journal of Operational Research*, 35(3), 378-381.
- [5] Hubert, L., & Schultz, J. (1976). Quadratic assignment as a general data analysis strategy. *British journal of mathematical and statistical psychology*, 29(2), 190-241.

- [6] Dell'Amico, M., Díaz, J. C. D., Iori, M., & Montanari, R. (2009). The single-finger keyboard layout problem. *Computers & Operations Research*, 36(11), 3002-3012.
- [7] Osman, I. H., & Laporte, G. (1996). Metaheuristics: A bibliography.
- [8] Sörensen, K. (2015). Metaheuristics—the metaphor exposed. *International Transactions in Operational Research*, 22(1), 3-18.
- [9] Michalewicz, Z. (2012). Quo vadis, evolutionary computation?. In *Advances in Computational Intelligence* (pp. 98-121). Springer, Berlin, Heidelberg.
- [10] Pradeepmon, T. G., Panicker, V. V., & Sridharan, R. (2016). Parameter Selection of Discrete Particle Swarm Optimization Algorithm for the Quadratic Assignment Problems. *Procedia Technology*, 25, 998-1005.
- [11] Pradeepmon, T. G., Sridharan, R., & Panicker, V. V. (2018). Development of modified discrete particle swarm optimization algorithm for quadratic assignment problems. *International Journal of Industrial Engineering Computations*, 9.
- [12] Mamaghani, A. S., & Meybodi, M. R. (2012, October). Solving the Quadratic Assignment Problem with the modified hybrid PSO algorithm. In *Application of Information and Communication Technologies (AICT), 2012 6th International Conference on* (pp. 1-6). IEEE.
- [13] Liu, H., Abraham, A., & Clerc, M. (2007). An hybrid fuzzy variable neighborhood particle swarm optimization algorithm for solving quadratic assignment problems. *J. UCS*, 13(9), 1309-1331.
- [14] Riffi, M. E., Saji, Y., & Barkatou, M. (2017). Incorporating a modified uniform crossover and 2-exchange neighborhood mechanism in a discrete bat algorithm to solve the quadratic assignment problem. *Egyptian Informatics Journal*, 18(3), 221-232.
- [15] Shukla, A. (2015, May). A modified bat algorithm for the Quadratic Assignment Problem. In *Evolutionary Computation (CEC), 2015 IEEE Congress on* (pp. 486-490). IEEE.
- [16] Ahmed, Z. H. (2015, July). An improved genetic algorithm using adaptive mutation operator for the quadratic assignment problem. In *Telecommunications and Signal Processing (TSP), 2015 38th International Conference on* (pp. 1-5). IEEE.
- [17] Ahmed, Z. H. (2015). A multi-parent genetic algorithm for the quadratic assignment problem. *Opsearch*, 52(4), 714-732.
- [18] Tosun, U. (2014). A new recombination operator for the genetic algorithm solution of the quadratic assignment problem. *Procedia Computer Science*, 32, 29-36.
- [19] Azarbondyad, H., & Babazadeh, R. (2014). A genetic algorithm for solving quadratic assignment problem (QAP). *arXiv preprint arXiv:1405.5050*.
- [20] Ahuja, R. K., Orlin, J. B., & Tiwari, A. (2000). A greedy genetic algorithm for the quadratic assignment problem. *Computers & Operations Research*, 27(10), 917-934.
- [21] Ahmed, Z. H. (2018). A hybrid algorithm combining lexsearch and genetic algorithms for the quadratic assignment problem. *Cogent Engineering*, 5(1), 1423743.
- [22] Taillard, É. (1991). Robust taboo search for the quadratic assignment problem. *Parallel computing*, 17(4-5), 443-455.

- [23] James, T., Rego, C., & Glover, F. (2009). Multistart tabu search and diversification strategies for the quadratic assignment problem. *IEEE TRANSACTIONS ON SYSTEMS, Man, And Cybernetics-part a: systems and humans*, 39(3), 579-596.
- [24] Misevicius, A. (2012). An implementation of the iterated tabu search algorithm for the quadratic assignment problem. *OR spectrum*, 34(3), 665-690.
- [25] Lim, W. L., Wibowo, A., Desa, M. I., & Haron, H. (2016). A biogeography-based optimization algorithm hybridized with tabu search for the quadratic assignment problem. *Computational intelligence and neuroscience*, 2016, 27.
- [26] Shylo, P. V. (2017). Solving the Quadratic Assignment Problem by the Repeated Iterated Tabu Search Method. *Cybernetics and Systems Analysis*, 53(2), 308-311.
- [27] Gambardella, L. M., Taillard, É. D., & Dorigo, M. (1999). Ant colonies for the quadratic assignment problem. *Journal of the operational research society*, 50(2), 167-176.
- [28] Pardalos, L., & Resende, M. (1994). A greedy randomized adaptive search procedure for the quadratic assignment problem. *Quadratic Assignment and Related Problems, DIMACS Series on Discrete Mathematics and Theoretical Computer Science*, 16, 237-261.
- [29] Stützle, T., & Dorigo, M. (1999). ACO algorithms for the quadratic assignment problem. *New ideas in optimization*, (C50), 33.
- [30] Duman, E., Uysal, M., & Alkaya, A. F. (2012). Migrating Birds Optimization: A new metaheuristic approach and its performance on quadratic assignment problem. *Information Sciences*, 217, 65-77.
- [31] Benlic, U., & Hao, J. K. (2015). Memetic search for the quadratic assignment problem. *Expert Systems with Applications*, 42(1), 584-595.
- [32] Abdel-Baset, M., Wu, H., Zhou, Y., & Abdel-Fatah, L. (2017). Elite opposition-flower pollination algorithm for quadratic assignment problem. *Journal of Intelligent & Fuzzy Systems*, 33(2), 901-911.
- [33] XIA, X., & ZHOU, Y. (2018). Performance Analysis of ACO on the Quadratic Assignment Problem. *Chinese Journal of Electronics*, 27(1).
- [34] Munera, D., Diaz, D., & Abreu, S. (2016, March). Solving the quadratic assignment problem with cooperative parallel extremal optimization. In *European Conference on Evolutionary Computation in Combinatorial Optimization* (pp. 251-266). Springer, Cham.
- [35] Chmiel, W., & Szwed, P. (2016). Bees algorithm for the quadratic assignment problem on CUDA platform. In *Man-Machine Interactions 4* (pp. 615-625). Springer, Cham.
- [36] Aksan, Y., Dokeroglu, T., & Cosar, A. (2017). A stagnation-aware cooperative parallel breakout local search algorithm for the quadratic assignment problem. *Computers & Industrial Engineering*, 103, 105-115.
- [37] Amudha, M. T., & Shivakumar, B. L. (2012). Improving the Solution Quality of Quadratic Assignment Problems using Harmony Improvised Consultant Guided Search Technique.

- [38] Chakrapani, J., & Skorin-Kapov, J. (1993). Massively parallel tabu search for the quadratic assignment problem. *Annals of Operations Research*, 41(4), 327-341.
- [39] Talbi, E. G., Roux, O., Fonlupt, C., & Robillard, D. (2001). Parallel ant colonies for the quadratic assignment problem. *Future Generation Computer Systems*, 17(4), 441-449.
- [40] James, T., Rego, C., & Glover, F. (2009). A cooperative parallel tabu search algorithm for the quadratic assignment problem. *European Journal of Operational Research*, 195(3), 810-826.
- [41] Brown, D. E., Huntley, C. L., & Spillane, A. R. (1989, June). A parallel genetic heuristic for the quadratic assignment problem. In *Proceedings of the 3rd International Conference on Genetic Algorithms* (pp. 406-415). Morgan Kaufmann Publishers Inc..
- [42] Mohammadi, J., Mirzaie, K., & Derhami, V. (2015, November). Parallel genetic algorithm based on GPU for solving quadratic assignment problem. In *Knowledge-Based Engineering and Innovation (KBEI), 2015 2nd International Conference on* (pp. 569-572). IEEE.
- [43] Tosun, U. (2015). On the performance of parallel hybrid algorithms for the solution of the quadratic assignment problem. *Engineering Applications of Artificial Intelligence*, 39, 267-278.
- [44] Cárdenas, E., Ch, R. P., & Garcia, O. (2017). A Solution for the Quadratic Assignment Problem (QAP) through a Parallel Genetic Algorithm Based Grid on GPU. *Applied Mathematical Sciences*, 11(57), 2843-2854.
- [45] Mirjalili, S., & Lewis, A. (2016). The whale optimization algorithm. *Advances in Engineering Software*, 95, 51-67.
- [46] Ahmed, Z. H. (2014). A simple genetic algorithm using sequential constructive crossover for the quadratic assignment problem.
- [47] Chmiel, W., Kadłuczka, P., Kwiecień, J., & Filipowicz, B. (2017). A comparison of nature inspired algorithms for the quadratic assignment problem. *Bulletin of the Polish Academy of Sciences Technical Sciences*, 65(4), 513-522.
- [48] Helal, A. M., & Abdelbar, A. M. (2014). Incorporating domain-specific heuristics in a particle swarm optimization approach to the quadratic assignment problem. *Memetic Computing*, 6(4), 241-254.
- [49] Said, G. A. E. N. A., Mahmoud, A. M., & El-Horbaty, E. S. M. (2014). A comparative study of meta-heuristic algorithms for solving quadratic assignment problem. *arXiv preprint arXiv:1407.4863*.
- [50] Ng, K. M., & Tran, T. H. (2018). A parallel water flow algorithm with local search for solving the quadratic assignment problem. *Journal of Industrial & Management Optimization*, 3070-3087.
- [51] Ramkumar, A. S., Ponnambalam, S. G., Jawahar, N., & Suresh, R. K. (2008). Iterated fast local search algorithm for solving quadratic assignment problems. *Robotics and Computer-Integrated Manufacturing*, 24(3), 392-401.
- [52] Hafiz, F., & Abdenmour, A. (2016). Particle Swarm Algorithm variants for the Quadratic Assignment Problems-A probabilistic learning approach. *Expert Systems with Applications*, 44, 413-431.

- [53] Koopmans, T. C., & Beckmann, M. (1957). Assignment problems and the location of economic activities. *Econometrica: journal of the Econometric Society*, 53-76.
- [54] Helber, S., Böhme, D., Oucherif, F., Lagershausen, S., & Kasper, S. (2016). A hierarchical facility layout planning approach for large and complex hospitals. *Flexible Services and Manufacturing Journal*, 28(1-2), 5-29.
- [55] Elshafei, A. N. (1977). Hospital layout as a quadratic assignment problem. *Journal of the Operational Research Society*, 28(1), 167-179.
- [56] Feng, X., & Su, Q. (2015, June). An applied case of quadratic assignment problem in hospital department layout. In *Service Systems and Service Management (ICSSSM), 2015 12th International Conference on* (pp. 1-5). IEEE.
- [57] Chraïbi, A., Osman, I. H., & Kharraja, S. (2018). Adaptive layout for operating theatre in hospitals: different mathematical models for optimal layouts. *Annals of Operations Research*, 1-35.
- [58] <https://www.opt.math.tugraz.at/qaplib/inst.html#BO>. Visited, 18/4/2018.
- [59] Rameshkumar, K. (2018, June). Extension of PSO and ACO-PSO algorithms for solving Quadratic Assignment Problems. In *IOP Conference Series: Materials Science and Engineering* (Vol. 377, No. 1, p. 012192). IOP Publishing.
- [60] Kakihata, E. M., Sapia, H. M., Oiakawa, R. T., Pereira, D. R., Papa, J. P., de Albuquerque, V. H. C., & da Silva, F. A. (2017). Intrusion Detection System Based On Flows Using Machine Learning Algorithms. *IEEE Latin America Transactions*, 15(10), 1988-1993.
- [61] Corneto, G. L., da Silva, F. A., Pereira, D. R., de Almeida, L. L., Artero, A. O., Papa, J. P., ... & Sapia, H. M. (2017). A new method for automatic vehicle license plate detection. *IEEE Latin America Transactions*, 15(1), 75-80.
- [62] Junior, F. S. R., de Albuquerque, V. H. C., Fernandes, R. C. G., & Tavares, J. M. R. (2015). Biomedical data management and processing-a new framework.
- [63] Taramasco, C., Rodenas, T., Martinez, F., Fuentes, P., Munoz, R., Olivares, R., ... & Demongeot, J. (2018). A Novel Monitoring System for Fall Detection in Older People. *IEEE Access*.

Accepted Manuscript

Integrating the whale algorithm with Tabu search for quadratic assignment problem: A new approach for locating hospital departments

Mohamed Abdel-Baset, M. Gunsekaran, Doaa El-Shahat,
Seyedali Mirjalili

PII: S1568-4946(18)30519-2
DOI: <https://doi.org/10.1016/j.asoc.2018.08.047>
Reference: ASOC 5082

To appear in: *Applied Soft Computing Journal*

Received date : 24 June 2018
Revised date : 14 August 2018
Accepted date : 17 August 2018

Please cite this article as: M. Abdel-Baset, et al., Integrating the whale algorithm with Tabu search for quadratic assignment problem: A new approach for locating hospital departments, *Applied Soft Computing Journal* (2018), <https://doi.org/10.1016/j.asoc.2018.08.047>

This is a PDF file of an unedited manuscript that has been accepted for publication. As a service to our customers we are providing this early version of the manuscript. The manuscript will undergo copyediting, typesetting, and review of the resulting proof before it is published in its final form. Please note that during the production process errors may be discovered which could affect the content, and all legal disclaimers that apply to the journal pertain.