A PARALLEL WATER FLOW ALGORITHM WITH LOCAL SEARCH FOR SOLVING THE QUADRATIC ASSIGNMENT PROBLEM

Kien Ming Ng
Department of Industrial Systems Engineering and Management
National University of Singapore
Singapore 119260

Trung Hieu Tran
Laboratory for Urban Complexity and Sustainability
University of Nottingham
Nottingham NG7 2RB, UK

(Communicated by Yuanguo Zhu)

Abstract. In this paper, we adapt a nature-inspired optimization approach, the water flow algorithm, for solving the quadratic assignment problem. The algorithm imitates the hydrological cycle in meteorology and the erosion phenomenon in nature. In this algorithm, a systematic precipitation generating scheme is included to increase the spread of the raindrop positions on the ground to increase the solution exploration capability of the algorithm. Efficient local search methods are also used to enhance the solution exploitation capability of the algorithm. In addition, a parallel computing strategy is integrated into the algorithm to speed up the computation time. The performance of the algorithm is tested with the benchmark instances of the quadratic assignment problem taken from the QAPLIB. The computational results and comparisons show that our algorithm is able to obtain good quality or optimal solutions to the benchmark instances within reasonable computation time.

1. Introduction. The quadratic assignment problem (QAP) is one of the well-known NP-hard combinatorial optimization problems [31] due to its important applications in practice, such as parallel and distributed computing [5], statistical data analysis [19], testing of electronic devices [13], plant layout design [30], data visualization [1], printed circuit board assembly process [12], and website structure improvement [32]. The problem is often described as follows: Given a set of facilities and a set of locations with the same size $n$, assign the facilities to the locations such that the total cost of assignment is minimized. The total cost $w$ is calculated using the distance between locations and the flow between facilities. The QAP can then be expressed as the problem of finding a permutation $\pi$ of $n$ facilities as follows:

$$\min_{\pi \in \Pi_n} w(\pi) = \sum_{i=1}^{n} \sum_{j=1}^{n} f_{ij}d_{\pi[i]\pi[j]},$$

---

2010 Mathematics Subject Classification. Primary: 90C27, 90B80; Secondary: 68W10.

Key words and phrases. Combinatorial optimization, quadratic assignment problem, nature-inspired optimization, water flow algorithm, parallel computing.

* Corresponding author: Trung Hieu Tran.
where $\Pi_n$ denotes the set of possible permutations of $N = \{1, 2, ..., n\}$, while $\pi[i]$ and $\pi[j]$ denote the location of facilities $i$ and $j$ in the permutation $\pi$ respectively. Here, $f_{ij}$ is the flow between facilities $i$ and $j$, and $d_{\pi[i]\pi[j]}$ is the distance between locations $\pi[i]$ and $\pi[j]$.

While there are some well-solved special cases of the QAP (see, for example, [10]), the QAP is generally difficult to solve [38], and so researchers have explored various possible solution methods. These solution methods can be broadly classified into two approaches: exact methods and heuristic methods. Exact methods aim to find the optimal solution to the QAP but they often run into computational difficulties with large QAP instances and are usually unable to solve instances of size $n > 30$ [8, 9, 17, 4, 15]. Heuristic methods on the other hand can be further divided into constructive heuristics and improvement heuristics. Constructive heuristics often construct a feasible solution for the QAP by assigning each facility to a location according to some principles. Such constructive methods for solving the QAP can be seen in [6, 3, 16]. However, they are only efficient for small and medium QAP instances and they may not produce good quality solutions for large QAP instances. Unlike constructive heuristics, improvement heuristics attempt to improve an existing solution through some iterative procedures. Some of the well-known algorithms belonging to this category are the metaheuristic algorithms, such as genetic algorithms [2, 23], ant colony optimization algorithms [24, 28, 41], greedy randomized adaptive search procedure [22], memetic algorithm [25], iterated fast local search algorithm [29], simulated annealing algorithm [37], DNA algorithm [43] and bat algorithm [33]. A summary of the findings and drawbacks of several state-of-the-art metaheuristic algorithms for the QAP based on their reported computational results on the large number of benchmark instances from the QAPLIB is presented in Table 1 (since the bat algorithm was tested on some small instances, we did not report it in the table). Although the algorithms can obtain a relatively high proportion in the number of the best known solutions for the tested instances (e.g., a minimum proportion of 55% with iterated fast local search and a maximum proportion of 75% with ant system) and a small average percentage difference from the optimal or best known objective value (e.g., a minimum of 0.17% with greedy genetic algorithm and a maximum of 0.99% with simulated annealing), the best known solutions found are almost focused on small and medium instances. When the maximum percentage difference is taken into consideration, there are several instances (e.g., chrxxx and lipaxxx instances) in which the algorithms could not obtain good quality solutions possibly due to drawbacks of the algorithms described in Table 1. Moreover, there are limited algorithms, such as iterated fast local search and hybrid genetic algorithm that could solve large QAP instances of size $n > 128$. However, these two algorithms obtained poor quality solutions for the lipaxxx instances since the 2-opt local search used in the exploitation phase could place a limit on the solution intensification capability when solving the instances. Thus, it is desirable to develop an algorithm that can overcome the shortcomings of the state-of-the-art algorithms so as to solve the QAP effectively even for large instances.

Among the metaheuristic algorithms, nature-inspired algorithms have recently gained attention by many researchers. Besides popular algorithms such as the genetic algorithm introduced by Holland [18], ant colony optimization proposed by Dorigo [11], particle swarm optimization developed by Kennedy and Eberhart [20] and honey bee algorithm designed by Nakrani and Tovey [26], novel optimization
Table 1. A summary of state-of-the-art metaheuristic algorithms for solving the QAP.

| Metaheuristic algorithms | No. of best known solutions found / No. of tested instances | Average difference | Maximum difference | Maximum size of instance | Drawbacks |
|--------------------------|-------------------------------------------------------------|--------------------|--------------------|-------------------------|-----------|
| Simulated annealing [37] | 26 / 40                                                     | 0.99%              | 11.51%             | 30                      | Solving QAP relaxation to construct initial solution for simulated annealing depends on the capability of solvers used [37]. Thus, the algorithm may not solve instances of size $n > 30$ effectively or extensive computation time may be required. |
| Ant system [24]          | 33 / 44                                                     | 0.28%              | 2.79%              | 40                      | Computation time of local search is rather onerous [24], and thus does not solve instances of size $n > 40$ efficiently. |
| Population based hybrid ant system [28] | 80 / 110                                                   | 0.41%              | 14.25%             | 100                     | Population size increases significantly according to instance size [28], leading to difficulty for solving large instances. |
| Greedy genetic algorithm [2] | 58 / 87                                                     | 0.17%              | 5.13%              | 100                     | Although greedy approach improves the quality of individuals, this may affect the overall performance of the genetic algorithm [2]. In addition, using 2-exchange local search could limit the capability for searching better solutions. |
| Greedy randomized adaptive search procedure [22] | 27 / 44                                                     | 0.69%              | 4.64%              | 128                     | The algorithm can be applied only to symmetric QAP instances [2]. |
| Iterated fast local search [29] | 72 / 130                                                   | 0.87%              | 20.33%             | 256                     | 2-opt local search of the algorithm could not solve $lipaxxx$ instances of the QAPLIB effectively [29] due to limit in search space. |
| Hybrid genetic algorithm [23] | 84 / 130                                                   | 0.51%              | 16.56%             | 256                     | 2-gene exchange local search could not solve $lipaxxx$ instances of the QAPLIB effectively [23] due to limit in search space. |
approaches based on the natural phenomenon of water flow have also been introduced. One of the earliest applications of the water flow based optimization model was proposed in image processing [21]. The authors simulated the image processing problem by imitating the property of water always moving to lower regions and filling valleys. A threshold process was then used to extract the amount of filled water that represents the characters required to remove from a document image. Oh et al. [27] proposed splitting the input image into interest and desert regions, with rainfall only occurring within the interest regions. Based on this idea, some heuristic algorithms were developed to save computation time for solving the problem. Yang and Wang [44] developed a water flow-like algorithm for solving the bin packing problem. In their algorithm, solution agents were simulated as water flows that can traverse the terrain mapped from the objective function to find improved solutions. The algorithm includes four operations: splitting and moving, merging, evaporation, and precipitation. The numerical results showed that this algorithm outperforms some methods in solving the bin packing problem. Wu et al. [42] adapted the water flow-like algorithm for solving the cell formulation problem. The authors designed the efficient machine shifting and insertion-move strategies in the water flow splitting and moving operations for searching the best neighborhood solution. In addition, some improved operations were integrated into this algorithm, such as the velocity-based evaporation, the moist precipitation included, and the prior estimation of cell size. The computational results demonstrated the superiority of the adapted algorithm in both solution effectiveness and efficiency over other compared approaches. Shah-Hosseini [34] introduced an optimization algorithm based on the dynamics of river systems, as well as the reactions between water drops and the changes of environment in a flowing river, for solving the traveling salesman problem. The experiment results showed that the algorithm may find good solutions. The algorithm was also applied to other optimization problems, such as the multiple knapsack problem [35] and the $n$-queen puzzle [36]. However, the performance of this algorithm is still limited due to the large number of parameters needed by the algorithm, and the incomplete utilization of the properties of water flow. Tran and Ng [39] developed an algorithm based on the properties as well as behaviors of water flow, such as the simulation of raindrop distribution, the property of water flow always moving to lower positions, and especially the erosion process of water flow on the terrain, for the flexible flow shop scheduling with intermediate buffers. The experiment results demonstrate the efficiency of this algorithm for the scheduling problem, as well as the potential of this algorithm for solving other combinatorial optimization problems, including multi-objective optimization problems [40]. Such nature-inspired optimization approaches continued to be developed with the water cycle algorithm [14] and the water wave optimization [45]. Eskandar et al. [14] designed the algorithm based on the water cycle process and how rivers and streams flow to the sea, while Zheng [45] constructed the algorithm based on the shallow water wave theory and the phenomena of water waves, such as propagation, refraction and breaking. The water cycle algorithm was tested on several constrained benchmark optimization and engineering design problems, but its applicability for large-scale optimization problems has not been examined. The water wave optimization was applied successfully for solving the real-world high-speed train scheduling problem in China. Both algorithms demonstrated their competitive performance with state-of-the-art nature-inspired algorithms in the literature.
Following the successful applications of such nature-inspired algorithms for solving other optimization problems, we adapt in this paper the water flow algorithm (WFA) for solving the QAP. The properties and behaviors of water flow in nature are reinforced by many efficient processes to enhance solution diversification and intensification capabilities of the WFA. In particular, the algorithm uses a systematic drop of water (DOW) generator scheme to distribute the positions of DOWs on the ground, where such distributed DOWs represent the permutations in the QAP. Also, this algorithm develops some efficient neighborhood structures to focus on strong exploitation of promising regions. A parallel computing strategy is applied in the distribution of DOWs and the exploitation of promising regions to improve the computation time of the algorithm. Benchmark problem instances drawn from the QAPLIB [7] are used to evaluate the performance of the algorithm. The computational results and comparisons show that the WFA is able to obtain good quality or optimal solutions to these instances and is a promising metaheuristic algorithm for solving the QAP instances.

The remaining content of this paper is organized as follows: In Section 2, the adapted WFA with parallel computing for solving the QAP is described in detail. In Section 3, computational results and comparisons of applying the WFA are shown. Next, the analysis of the results is discussed in Section 4. Finally, some conclusions and possible future work are presented in Section 5.

2. WFA with parallel computing for solving the QAP. WFA is a nature-inspired algorithm, introduced by Tran and Ng [39] for solving certain types of combinatorial optimization problems such as the flexible flow shop scheduling problems with intermediate buffer, and then developed by the same authors [40] for the multi-objective scheduling optimization problem. This algorithm is inspired by the hydrological cycle in meteorology and the erosion phenomenon in nature. The properties and behaviors of water flow in nature are simulated to construct the main operators of the WFA, such as DOWs always moving to lower positions, spreading of DOWs onto many places on the ground, and DOWs eroding obstacles. The simulation of the natural phenomena in this algorithm aims to achieve a balance between its solution exploration and exploitation capabilities.

2.1. Description of standard WFA. We begin with a description of the relationship between the natural phenomena of water flow and the main phases of the WFA. The exploration phase of the WFA mimics the circulation of water (i.e., hydrological cycle) to generate a population of new positions (represented by longitude, latitude, and altitude) for DOWs after each cloud is generated (known as precipitation). The exploitation phase of the WFA imitates the erosion phenomenon to intensify the promising local optimal positions of DOWs and to overcome the obstacles to reach lower positions. The rate of intensity of erosion depends on the amount of precipitation, the falling force of precipitation and the soil hardness. In the WFA, the erosion capability is thus determined by a function of the number of DOWs at the eroding position and the altitude of that position [39].

The standard procedure of the WFA for solving an optimization problem can be described as follows. Firstly, a cloud (an iteration in the WFA) randomly generates a set of DOWs onto some positions on the ground (the solution space of the problem). Because of the gravity force of Earth (a local search algorithm), the DOWs flow down to their local optimal positions. Then, a set of promising positions to perform erosion process is obtained. If the erosion condition is satisfied, the erosion process
is performed at these positions. The erosion process helps the DOWs overcome the local optimal positions to find better or global positions. The procedure is repeated by regenerating new positions for DOWs out of the eroded regions until the termination conditions are met. These conditions consist of the maximum number of clouds, the amount of limited time or non-improvement on the best position obtained after a certain number of clouds.

We next describe in detail the procedure for solving the QAP by the WFA with parallel computing. The main phases in the WFA include the exploration phase, the exploitation phase, and the improvement phase, as well as a systematic precipitation generating scheme. The solution representation and memory lists, which are the basic components of the WFA for solving the QAP, are first described below.

2.2. Encoding scheme and memory lists. The WFA involves encoding a feasible solution of an optimization problem and its objective value into a DOW, which is a component of a cloud representing a pool of solutions. In particular, we consider the permutation \( \pi \) of \( n \) facilities in the QAP as the longitude and latitude in the position of DOW on the ground, while the total cost of flow between the facilities is encoded as the altitude. Given a permutation of \( n \) facilities \( \pi = (\sigma_1, ..., \sigma_n) \), we define:

\[
\text{longitude}(\pi) = (\sigma_1, ..., \sigma_{\lfloor \frac{n}{2} \rfloor}),
\]

\[
\text{latitude}(\pi) = (\sigma_{\lfloor \frac{n}{2} \rfloor + 1}, ..., \sigma_n),
\]

where \( \lfloor \varepsilon \rfloor \) is the largest integer less than or equal to \( \varepsilon \). An illustrative example of a DOW and its positional vector components for the QAP with \( n = 8 \) facilities is shown in Figure 1.

With this encoding scheme, the neighborhood structure used in WFA for the QAP is mainly based on exchanging elements in a permutation. An example is the 2-opt neighborhood structure used in the traveling salesman problem as well as in the QAP [25]. Thus if \( \pi' \) is the permutation obtained by exchanging positions of two facilities \( i \) and \( j \) in the permutation \( \pi \), we can determine \( \pi' \) by:

\[
\pi'[k] = \pi[k], \forall k \in N \setminus \{i, j\}
\]

\[
\pi'[i] = \pi[j],
\]

\[
\pi'[j] = \pi[i],
\]
In addition, we also consider the neighborhood structure of an extended 2-opt algorithm called the 2-opt mirror. In this 2-opt mirror algorithm, other than the usual 2-opt neighboring permutations, we also consider the reflected permutation and its corresponding 2-opt neighboring permutations. The reflected permutation \( \pi_r \) of a permutation \( \pi \) can be determined as follows:

\[
\pi_r[i] = \pi[n - i + 1], \quad \forall i = 1, \ldots, n.
\]

The neighborhood structures are used for both exploration and exploitation phases.

To support the search for global optimal positions, we use three sets of memory lists, namely the best positions list (\( P_0 \)-list), the un-eroded positions list (UE-list), and the eroded positions list (E-list). Here, the \( P_0 \)-list is used to save the positions with the best-known objective value. The UE-list is used to record the local optimal positions which have not been eroded due to the erosion condition not being satisfied. The E-list is used to save the eroded local optimal positions. Hence, the E-list plays an important role in preventing the next clouds from regenerating DowS to the eroded positions. This would in turn help to improve the computation time of the algorithm. Both the UE-list and the E-list are updated after performing the erosion process. The UE-list together with the \( P_0 \)-list is also updated each time a Dow finds a new local optimal position.

Let \( \pi_{c, \text{Best}} \) be the best optimal position found so far by the WFA at cloud \( c \). Assume that we have found a local optimal position \( \pi_{c+1}^{*} \) at cloud \( c + 1 \). Updating of the \( P_0 \)-list at the cloud can then be described as follows:

\[
P_0\text{-list} = \begin{cases} 
\pi_{c+1}^{*} & \text{if } w(\pi_{c+1}^{*}) < w(\pi_{c, \text{Best}}) \\
\pi_{c, \text{Best}} & \text{otherwise.}
\end{cases}
\]

Updating of the UE-list includes two phases in which the phase of removing the local optimal positions eroded and the phase of adding the ones just found are done in succession, while updating of the E-list is only to add the local optimal positions eroded. They are shown as follows:

\[
\text{UE-list}_{c+1} = (\text{UE-list}_c \cup \Pi^2_c) \setminus \Pi^1_c, \quad (7)
\]

\[
\text{E-list}_{c+1} = \text{E-list}_c \cup \Pi^1_c, \quad (8)
\]

where \( \Pi^1_c \) and \( \Pi^2_c \) denote the set of local optimal positions eroded and the set of local optimal positions just found from initial positions generated at each iteration respectively.

2.3. Exploration phase. Two schemes for generating the initial population of Dow positions in the exploration phase are proposed as follows:

The first scheme is the random position generator scheme. In this scheme, a population \( \Omega \) of positions of DowS is generated randomly for each cloud and the number of such positions is the maximum population size allowed (\( \text{MaxPop} \)). Since the WFA mimics the property of water flow always moving from higher positions to lower positions due to Earth’s gravity, a steepest ascent hill sliding algorithm is thus applied to search for local optimal positions from these initial positions. In particular, from an initial solution, the hill sliding algorithm searches for the best improved solution within the initial solution’s neighbors in terms of objective value. Then, this process continues to be performed iteratively for the improved solution obtained until no other improved solution is found.

Due to the random nature of this scheme, the efficiency of WFA for solving the QAP may fluctuate in instances with large size. To resolve this drawback of the first
scheme as well as to improve the solution exploration capability of WFA for QAP instances with large size, a second scheme is proposed. This scheme is a systematic DOW generator scheme that aims to distribute DOWs evenly into divided regions of the solution space. We first divide the solution space into \( n \) regions, where \( n \) is the instance size. Then at each cloud, the WFA generates \( n \) DOWs and each DOW is assigned to only one distinct region. This means that a cloud would consist of \( n \) different positions, which is achieved by fixing the first position of a facility in a permutation from 1 to \( n \) when generating the permutations of DOW. The rest of the positions in a permutation are assigned randomly. This can be represented with the following notation:

\[
\Omega_c = \left\{ [\pi^1_c, ..., \pi^i_c, ..., \pi^n_c]^T \mid \pi^i_c[1] = i, \quad \forall i = 1, ..., n \right\}, \quad \forall c = 1, ..., MaxCloud. \tag{9}
\]

A steepest ascent hill sliding algorithm is also used in the second scheme to search for local optimal positions from these initial positions.

At each cloud, the number of DOWs at the initial positions is updated as follows:

\[
Q^{\pi^{i+1}_c} = \begin{cases} Q^\pi_c + 1 & \text{if } \pi^{i+1}_c = \pi \in \Pi^3_c, \\ 1 & \text{otherwise}, \end{cases} \quad \forall i = 1, ..., MaxPop, \tag{10}
\]

where \( Q^\pi_c \) is the number of DOWs in the position \( \pi \) at cloud \( c \), and \( \pi^{i+1}_c \) denotes the position of \( i \)th DOW generated at cloud \( c + 1 \), while the set of initial positions generated or local optimal positions found through clouds 1 to \( c \) is denoted by \( \Pi^3_c \).

After the steepest ascent hill sliding algorithm is applied to find the local optimal position \( \pi^{*+1}_c \) from the initial position \( \pi^{i+1}_c \), we also update the number of DOWs at the optimal position according to the following equation.

\[
Q^{\pi^{*+1}_c} = \begin{cases} Q^{\pi^{*+1}_c} + Q^{\pi^{i+1}_c} & \text{if } \pi^{*+1}_c \in \text{UE-list}, \\ Q^{\pi^{i+1}_c} & \text{otherwise}. \end{cases} \tag{11}
\]

In general, the exploration phase in the WFA for the QAP results in a set of local optimal permutations. These permutations and the number of DOWs at the permutations are updated in the UE-list to be considered for performing the erosion process in the exploitation phase.

2.4. Exploitation phase. The exploitation phase involves applying the erosion process to overcome the local optimal positions found in the exploration phase. Before describing the erosion process, the erosion condition and limit are first described below.

2.4.1. Erosion condition and limit. The erosion process is triggered by the amount of precipitation. Thus if the number of DOWs at a local optimal permutation increases to the threshold \( \text{MinEro} \), the erosion process is performed at this local optimal permutation.

Next, we consider the limit \( L \) of the erosion process. For the QAP, the erosion limit is based on a main factor, which is the number of DOWs at the eroding local optimal permutation. In particular, the relationship between the erosion limit and this factor is a nonlinear function. However, to simplify the computations in the WFA, we have assumed the erosion limit to be a constant value \( \text{MaxUIE} \), so that
**Procedure** Erosion Process;

Begin

Do loop

Choose un-eroded direction with the smallest $\Delta d_h$ to erode;

Do loop

Apply the steepest ascent hill sliding algorithm for the erosion direction $h$ chosen;

Until (a new local optimal permutation is found or no improvement after $\text{MaxUIE}$ steps)

If (new local optimal permutation found is better than eroding local optimal permutation) Then

Update it into UE-list to continue performing erosion process and update E-list;

End if

Until (erosion termination condition is satisfied)

End.

Figure 2. The erosion process in exploitation phase of the WFA.

the relationship can be described as follows:

$$L(Q_{c*}^{\pi*}) = \begin{cases} \text{MaxUIE} & \text{if } Q_{c*}^{\pi*} \geq \text{MinEro}, \\ 0 & \text{otherwise.} \end{cases} \quad (12)$$

If the erosion process cannot find any improved permutation after $\text{MaxUIE}$ search steps, the erosion process stops and other permutations in the UE-list are considered for performing the next erosion process.

2.4.2. Erosion process. The erosion process is the main operator in the exploitation phase of WFA for solving the QAP. Its task is to help the DOWs overcome local optimal permutations and obtain better local optimal or global optimal permutations. The operational mechanism of this operator is inspired by the erosion phenomenon caused by water flow in nature. In the QAP, the erosion process is built on a topological parameter $\Delta d_h$ representing the geographical shape of the surface. It is defined as the difference of total cost between the local optimal position and its $h$th neighboring position:

$$\Delta d_h = w(\pi_h^*) - w(\pi^*) = \sum_{i=1}^{n} \sum_{j=1}^{n} f_{ij} \left( d_{\pi_{h}^{\pi*}[i] \pi_{h}^{\pi*}[j]} - d_{\pi^*[i] \pi^*[j]} \right), \forall h = 1, ..., \frac{n(n-1)}{2}, \quad (13)$$

where $\pi^*$ and $\pi_h^*$ denote the permutations of the local optimal position and its $h$th neighboring position respectively. In the case of the 2-opt mirror neighborhood, the number of directions is $n(n - 1) + 1$.

The aim of computing $\Delta d_h$ is to help the erosion process choose the most suitable direction to perform erosion. In particular, the smallest $\Delta d_h$ is chosen to be the first erosion direction. If the erosion process for the erosion direction is not able to find a better permutation after $\text{MaxUIE}$ steps, searching for that direction stops and we say that the direction is blocked. This is then followed by a backtracking procedure in which the search from the former local optimal permutation is restarted using another erosion direction with the next smallest $\Delta d_h$. If all the directions are blocked, we say that the local optimal permutation is fully blocked and we move it into the E-list so that it will not be considered for erosion process in the next clouds. On the other hand, if the erosion process is able to find a better permutation than the eroding local optimal permutation, that erosion direction is chosen to erode the local optimal permutation permanently. We then update the new local optimal permutation into the UE-list to continue with performing the erosion process.

The entire erosion process is summarized by Figure 2.
2.5. **Improvement phase.** The exploration and exploitation phases of the WFA for the QAP terminate when the maximum number of clouds (*MaxCloud*) has been generated. To improve the solutions obtained by the WFA, we can apply the 3-opt algorithm at a final improvement phase. This 3-opt algorithm would still use the steepest ascent hill sliding method mentioned in the previous section. A 3-opt list is also used to save the solutions obtained by this improvement algorithm.

In summary, we have the following variants of WFA:

1. **Random 2-opt WFA:** WFA with the random position generator scheme and the 2-opt neighborhood structure.
2. **Systematic generator 2-opt WFA:** WFA with the systematic *DOW* generator scheme and the 2-opt neighborhood structure.
3. **Random 2-opt mirror WFA:** WFA with the random position generator scheme and the 2-opt mirror neighborhood structure.

2.6. **Parallel computing strategy.** In the exploration phase of the WFA, after generating a number of *DOW*s onto the regions, we perform the search process for the local optimal positions of the *DOW*s. To speed up the computation time, we divide equally the number of *DOW*s and assign them into multiple cores of the CPU to implement simultaneously the search process. A similar parallel computing strategy is applied in the exploitation phase of the WFA for the permutations in the UE-list that satisfy the erosion condition. Then, the erosion process is done at the permutations at the same time. Finally, the strategy is used in 3-opt algorithm at the improvement phase of the WFA. At each iteration, the number of constructed neighboring solutions is equally divided and assigned into multiple cores to evaluate their objective values simultaneously. The best neighboring solution is chosen to continue the 3-opt algorithm. A flow chart of the WFA with parallel computing for solving the QAP is shown in Figure 3.

In general, the parallel computing strategy is very useful for such multiple-agent-based algorithm type. When applying the strategy for the WFA to solve the QAP, the efficiency of its computation time is significantly improved, especially for the large-size instances. On average, we can solve 4.3 times (with the usage of 8 cores) faster than the WFA without parallel computing.

3. **Computational results.** To test the performance of the adapted WFA, we have used the random 2-opt WFA for solving all the benchmark instances from the QAPLIB [7]. The systematic generator 2-opt WFA and the random 2-opt mirror WFA are used to solve the QAP benchmark instances when the random 2-opt WFA has not obtained their best known objective value.

3.1. **Benchmark problem sets.** The 134 benchmark instances drawn from the QAPLIB [7] are well-known benchmark problem sets in QAP with size ranging from 12 to 256. The best known upper bounds of these problem sets obtained from the literature were used to compare with the best results obtained by the WFA. In addition, we also compared these results with those obtained by greedy randomized adaptive search procedure (GRASP) of Li et al. [22], by ant system (ANT) of Maniezzo and Colorni [24], by greedy genetic algorithm (GGA) of Ahuja et al. [2], by hybrid genetic algorithm with partial local search (PGA) of Lim et al. [23], by iterated fast local search algorithm (IFLS) of Ramkumar et al. [29], by two-level modified simulated annealing based approach (MSA) of Singh and Sharma [37], and by population-based hybrid ant system (PHAS) of Ramkumar et al. [28], all of
which are some of the most efficient metaheuristic algorithms for solving the QAP instances.
3.2. Platform and parameters. The WFA with parallel computing strategy was implemented in Visual C++, and all the computational experiments were run on a desktop PC with an Intel Core i7-3770 processor (8 cores, 2.40 GHz per chip) and 24 GB of RAM.

The choice of parameters for WFA was determined by the design-of-experiment (DOE) method. In particular, we carried out several simulations that test the WFA on all types of QAP with various values for the controlled parameters, i.e., the exploration parameters $\text{MaxCloud}$ and $\text{MaxPop}$, and the exploitation parameters $\text{MaxUIE}$ and $\text{MinEro}$. The aim is to determine the best parameters of WFA for QAP that would achieve a balance between solution exploration and exploitation capabilities in finding the best solutions within reasonable computation time. Thus, smaller values may be used for the parameters for larger problem instances. The values that were used are as follows: $\text{MaxCloud} = 2, 5, 10, 15, 20$; $\text{MaxPop} = 8, 16, 24$; $\text{MaxUIE} = 5, 10, 15, 20$; and $\text{MinEro} = 2, 3$.

From the preliminary simulation results, the best parameter sets are shown in Table 2. These parameter sets are then used for the random 2-opt WFA and the random 2-opt mirror WFA. For the systematic generator 2-opt WFA, we have used the parameter sets $(\text{MaxCloud}, \text{MaxPop}, \text{MaxUIE}, \text{MinEro}) = (50, n, 20, 2), (20, n, 10, 2),$ and $(5, n, 5, 2)$ for instances with size at most 50, more than 50 but at most 100, and more than 100 respectively, in order to allow for a reasonable amount of the erosion process to occur.

With these parameter sets, 5 independent replicates were used for each instance and the best results obtained from these replicates were used to evaluate the performance of WFA, as well as to compare with other metaheuristic algorithms.

3.3. Computational results. For comparison of objective values, we have used the following relative percentage difference in objective value:

$$\Delta_{\text{Best}} = \left( \frac{\text{Heuristic}_{\text{sol}} - \text{Opt}_{\text{sol}}}{\text{Opt}_{\text{sol}}} \right) \times 100,$$

where $\text{Heuristic}_{\text{sol}}$ and $\text{Opt}_{\text{sol}}$ denote the best objective function value obtained by the WFA and the best known value in the literature respectively.

For evaluation and comparison of the overall performance of algorithms, we use criteria such as the average relative percentage difference for all instances solved and the number of the best known solutions obtained in all instances solved. Thus, the algorithm with a smaller average relative percentage difference and a larger number of the best known solutions obtained would be considered a more effective optimization method.

The computational results and comparisons with other metaheuristic algorithms, such as GRASP, ANT, GGA, PGA, IFLS, MSA, and PHAS, are shown in Tables 3 to 7, while the improvement results obtained by the variants of WFA for the QAP benchmark instances that have not been solved optimally by the 2-opt WFA are displayed in Table 8. In Tables 3 to 7, the column with the best results of WFA shows the best solutions obtained by the random 2-opt WFA and the variants of WFA. Since the details of the computation time of applying PHAS to the QAP instances were not shown in [28], we did not include this information in Tables 3 to 7. In addition, the symbol "--" in the entries of these tables is used to indicate that the compared algorithms have not solved the respective benchmark instances. For Table 8, the entries displayed in italic font highlight the best results obtained by the respective variant of the WFA.
Table 2. The parameter sets of the WFA for the QAP benchmark instances.

| Benchmarks       | n          | Parameter values | MaxCloud | MaxPop | MaxUIE | MinEro |
|------------------|------------|------------------|----------|--------|--------|--------|
| Burkard          | 26         |                  |          |        |        |        |
| Christofides     | 12 – 20    |                  |          |        |        |        |
|                  | 22         |                  |          |        |        |        |
|                  | 25         |                  |          |        |        |        |
| Elshafei         | 19         |                  |          |        |        |        |
| Eschermann       | 16, 64     |                  |          |        |        |        |
|                  | 32 (a, b)  |                  |          |        |        |        |
|                  | 32 (c, e, g) |                |          |        |        |        |
|                  | 32 (d, h)  |                  |          |        |        |        |
|                  | 128        |                  |          |        |        |        |
| Hadley           | 12 – 20    |                  |          |        |        |        |
| Krarup           | 30, 32     |                  |          |        |        |        |
| Li & Pardalos    | 20, 30     |                  |          |        |        |        |
|                  | 40, 50, 60 |                  |          |        |        |        |
|                  | 70         |                  |          |        |        |        |
|                  | 80, 90     |                  |          |        |        |        |
| Nugent           | 12 – 28    |                  |          |        |        |        |
|                  | 30         |                  |          |        |        |        |
| Roucairol        | 12, 15     |                  |          |        |        |        |
|                  | 20         |                  |          |        |        |        |
| Scriabin         | 12, 15, 20 |                  |          |        |        |        |
| Skorin-Kapov     | 42 – 64    |                  |          |        |        |        |
|                  | 72, 81, 90 |                  |          |        |        |        |
|                  | 100        |                  |          |        |        |        |
| Steinberg        | 36         |                  |          |        |        |        |
| Taillard (Taixxa)| 12         |                  |          |        |        |        |
|                  | 15, 17     |                  |          |        |        |        |
|                  | 20 – 35    |                  |          |        |        |        |
|                  | 40, 50     |                  |          |        |        |        |
|                  | 60, 80, 100|                  |          |        |        |        |
| Taillard (Taixxb)| 12 – 20    |                  |          |        |        |        |
|                  | 25         |                  |          |        |        |        |
|                  | 30, 35, 40 |                  |          |        |        |        |
|                  | 50, 60, 80 |                  |          |        |        |        |
|                  | 100        |                  |          |        |        |        |
|                  | 150        |                  |          |        |        |        |
| Taillard (Taixxc)| 64         |                  |          |        |        |        |
|                  | 256        |                  |          |        |        |        |
| Thonemann        | 30         |                  |          |        |        |        |
|                  | 40         |                  |          |        |        |        |
|                  | 150        |                  |          |        |        |        |
| Wilhelm          | 50         |                  |          |        |        |        |
|                  | 100        |                  |          |        |        |        |
Table 3. Comparison results of the WFA with other algorithms for Burkard’s and Christofides’ instances.

| Instances | Best known value | Random 2-opt | WFA | GRASP | ANT | GGA | PGA | IFLS | MSA | PHAN |
|-----------|------------------|--------------|------|-------|-----|-----|-----|-----|-----|-----|
|           | Best solution | Time (s) | Δ | Time (s) | Δ | Time (s) | Δ | Time (s) | Δ | Time (s) | Δ |
| Bur26a    | 5426670 | 30.0 | 0 | 30.0 | 0 | 11.38 | 0 | 21.07 | 0 | 235 | 0 |
| Bur26b    | 3817852 | 23.0 | 0 | 23.0 | 0 | 59.45 | 0 | 35.03 | 0 | 225 | 0 |
| Bur26c    | 5426795 | 16.0 | 0 | 16.0 | 0 | 5.16 | 0 | 19.09 | 0 | 227 | 0 |
| Bur26d    | 3821225 | 29.0 | 0 | 29.0 | 0 | 15.12 | 0 | 19.40 | 0 | 213 | 0 |
| Bur26e    | 5386879 | 32.0 | 0 | 32.0 | 0 | 17.63 | 0 | 20.53 | 0 | 218 | 0 |
| Bur26f    | 3752044 | 41.0 | 0 | 41.0 | 0 | 5.05 | 0 | 11.23 | 0 | 104 | 0 |
| Bur26g    | 10117172 | 26.0 | 0 | 26.0 | 0 | 22.58 | 0 | 18.67 | 0 | 194 | 0 |
| Chr12a    | 9552 | 0.6 | 0 | 0.6 | 0 | 19.56 | 0 | 3.54 | 0 | 1.09 | 0 |
| Chr12b    | 9742 | 0.5 | 0 | 0.5 | 0 | 18.4 | 0 | 0.42 | 0 | 1.11 | 0 |
| Chr12c    | 11156 | 0.6 | 0 | 0.6 | 0 | 20.2 | 0 | 1.29 | 0 | 1.02 | 0 |
| Chr15a    | 9806 | 3.2 | 0 | 3.2 | 0 | 40.6 | 0 | 1.50 | 0 | 2.97 | 0 |
| Chr15b    | 7990 | 4.1 | 0 | 4.1 | 0 | 41.8 | 0 | 1.31 | 0 | 3.08 | 2.7 |
| Chr15c    | 5904 | 4.0 | 0 | 4.0 | 0 | 44 | 0 | 1.30 | 0 | 2.64 | 11.5 |
| Chr18a    | 11098 | 9.3 | 0 | 9.3 | 0 | 79 | 0 | 2.11 | 5.14 | 3.61 | 10.95 |
| Chr18b    | 1554 | 10.2 | 0 | 10.2 | 0 | 78.8 | 0 | 2.62 | 0 | 5.30 | 0 |
| Chr18c    | 2192 | 32.0 | 0 | 32.0 | 1.82 | 509 | 0 | 331 | 0 | 94.6 | 0.18 |
| Chr20a    | 2288 | 27.0 | 0 | 27.0 | 5.92 | 195 | 2.79 | 375 | 5.13 | 96.4 | 3.12 |
| Chr20b    | 14142 | 15.0 | 0 | 15.0 | 0.00 | 9.23 | 0 | 29.49 | 0 | 97.8 | 4.51 |
| Chr22a    | 6156 | 60.0 | 0 | 60.0 | 2.31 | 201 | 0 | 315 | 0.75 | 146 | 0.88 |
| Chr22b    | 6154 | 58.0 | 0 | 58.0 | 2.58 | 213 | 0 | 152 | 1.46 | 5.26 | 1.68 |
| Chr25a    | 3796 | 95.0 | 0 | 95.0 | 2.32 | 115 | 0 | 326 | 0 | 194 | 2.27 |

KUEN MING NG AND TRUNG HIEU TRAN
Table 4. Comparison results of the WFA with other algorithms for Elshafei’s, Eschermann’s, Hadley’s, and Krarup’s instances.

| Instances | Best known value | Random 2-opt WFA | Best results of WFA | GRASP | ANT | GGA | PGA | IFLS | MSA | PHAS |
|-----------|-----------------|------------------|---------------------|------|-----|-----|-----|------|-----|------|
| Els19     | 17212548        | 17212548         | 15                  | 0    | 15  | 0   | 0   | 80.6 | 0   | 44.46 |
| Esc16a    | 68              | 68               | 0.1                 | 0    | 0.1 | 0   | 0   | 0    | 0   | 0    |
| Esc16b    | 16              | 16               | 0.1                 | 0    | 0.1 | 0   | 0   | 0    | 0   | 0    |
| Esc16c    | 16              | 16               | 0.1                 | 0    | 0.1 | 0   | 0   | 0    | 0   | 0    |
| Esc16d    | 28              | 28               | 0.1                 | 0    | 0.1 | 0   | 0   | 0    | 0   | 0    |
| Esc16f    | 0               | 0                | 0.1                 | 0    | 0.1 | 0   | 0   | 0    | 0   | 0    |
| Esc16g    | 50              | 50               | 0.1                 | 0    | 0.1 | 0   | 0   | 0    | 0   | 0    |
| Esc16h    | 14              | 14               | 0.1                 | 0    | 0.1 | 0   | 0   | 0    | 0   | 0    |
| Esc16j    | 2               | 2                | 0.1                 | 0    | 0.1 | 0   | 0   | 0    | 0   | 0    |
| Esc32a    | 130             | 190              | 1.54                | 7.03 | 0   | 226 | 0   | 382  | 1.52 | 97.04 |
| Esc32b    | 168             | 190              | 1.5                 | 7.03 | 0   | 226 | 0   | 382  | 1.52 | 97.04 |
| Esc32c    | 642             | 642              | 1.2                 | 0.45 | 0   | 0   | 0   | 389  | 2.01 | 54.41 |
| Esc32d    | 200             | 11               | 11                  | 1.92 | 0   | 2.33 | 0   | 353  | 2.76 | 74.31 |
| Esc32e    | 2               | 2                | 0.8                 | 0.8  | 0   | 0   | 0   | 370  | 0.66 | 46.09 |
| Esc32e    | 6               | 6                | 0.9                 | 0.9  | 0   | 0   | 0   | 371  | 1.27 | 28.41 |
| Esc32f    | 48              | 31               | 31                  | 3.41 | 0   | 2.64 | 0   | 349  | 6.54 | 85.75 |
| Esc64a    | 116             | 96               | 96                  | 96   | 0   | 263 | 0   | 194  | 152  | 1852 |
| Esc64b    | 64              | 1395             | 1395                | 1395 | 0   | 1631| 0   | 0    | 0    | 0    |
| Had12     | 1652            | 1652             | 0.7                 | 0    | 0.7 | 0   | 0   | 4.27 | 0   | 41.0 |
| Had14     | 3753            | 15               | 1.5                 | 0    | 1.5 | 0   | 0   | 10.25| 0   | 1.97 |
| Had16     | 3720            | 2.2              | 2.2                 | 0    | 2.2 | 0   | 0   | 5.38 | 0.05 | 3.64 |
| Had18     | 5358            | 4.9              | 4.9                 | 4.9  | 0   | 18.54| 0   | 6.52 | 0   | 118  |
| Had20     | 6922            | 11.2             | 11.2                | 11.2 | 0   | 15.26| 0   | 10.56| 0   | 148  |
| Kra30a    | 89800           | 89800            | 430                 | 0    | 430 | 292 | 0   | 301  | 0.89 | 71   |
| Kra30b    | 91200           | 91200            | 441                 | 0    | 441 | 268 | 0   | 331  | 0.13 | 102  |
| Kra32     | 88700           | 88700            | 432                 | 0    | 432 | 140 | 0   | 331  | 0.13 | 102  |
Table 5. Comparison results of the WFA with other algorithms for Li & Pardalos' and Skorin-Kapov's instances.

| Instances   | Best known value | Best solution | Random 2-opt WFA | Best results of WFA | GRASP | ANT | GA | PGA | IFLS | MSA | PHAS |
|-------------|-----------------|---------------|------------------|---------------------|-------|-----|----|-----|-----|-----|------|
| Lipa20a     | 3683            | 3683          | 14               | 0                   | 14    | 0.99| 0  | 107 | 74.8| 0.39| 16.11|
| Lipa20b     | 27076           | 27076         | 16               | 0                   | 16    | 0.66| 0  | 0   | 74.4| 0.39| 16.76|
| Lipa30a     | 13178           | 13178         | 75               | 0                   | 75    | 46.43| 0  | 54.85| 345  | 5.66| 0    | 120  |
| Lipa30b     | 151426          | 151426        | 73               | 0                   | 73    | 7.31| 0  | 0   | 337  | 2.78| 0    | 122  |
| Lipa40a     | 31538           | 31538         | 655              | 0                   | 655   | 1.13| 0  | 1.02| 281  | 0.96| 1022 | 0    | 19.46| 0   | 490  |
| Lipa40b     | 476581          | 476581        | 430              | 0                   | 430   | 6.21| 0  | 0   | 1026 | 9.52| 0    | 486  |
| Lipa50a     | 120044          | 120044        | 777              | 0                   | 777   | 0   | 0  | 0   | 1509 | 39.96| 0    | 1462 |
| Lipa50b     | 107218          | 107218        | 1337             | 0.79                | 2754  | 0   | 0  | 7.31| 3057 | 0.64| 137  | 0    | 3668 |
| Lipa60a     | 2520135         | 2520135       | 893              | 0                   | 893   | 0   | 0  | 0   | 3047 | 86.13| 0    | 3724 |
| Lipa60b     | 349755          | 349755        | 1714             | 0.71                | 1744  | 0   | 0  | 0.71| 6148 | 0.62| 233  | 0    | 8067 |
| Lipa70a     | 4603200         | 4603200       | 1771             | 0                   | 1771  | 0   | 0  | 0   | 2123 | 15.99| 0    | 7762 |
| Lipa70b     | 7760862         | 7760862       | 2511             | 0                   | 2511  | 0   | 0  | 0   | 1799 | 15.56| 0    | 332  |
| Lipa80a     | 340630          | 340630        | 2562             | 0.57                | 5678  | 0   | 0  | 0.58| 12358| 0.54| 592  | 0    | 27999|
| Lipa80b     | 12490441        | 12490441      | 5034             | 0                   | 5034  | 0   | 0  | 0   | 12319| 0    | 503  | 0    | 27788|
| Sko42       | 125612          | 125612        | 1042             | 0.03                | 1347  | 0.25| 0  | 0.25| 1006 | 0.35| 363  | 0    | 614  |
| Sko49       | 23386           | 23386         | 1098             | 0.32                | 1772  | 0.21| 122 | 0.21| 1252 | 0.19| 714  | 0.45| 1318 |
| Sko56       | 33458           | 33458         | 1064             | 0.60                | 1742  | 0.02| 2976| 0.02| 2976 | 0.06| 907  | 0.47| 2613 |
| Sko64       | 44998           | 44998         | 1178             | 0.32                | 2770  | 0.22| 3788| 0.22| 3788 | 0.09| 1399 | 0.25| 4936 |
| Sko72       | 66256           | 66256         | 2072             | 0.47                | 4096  | 0.29| 5078| 0.29| 5078 | 0.21| 1967 | 0.73| 8863 |
| Sko81       | 90998           | 91452         | 1520             | 0.50                | 1655  | 0.20| 10964| 0.20| 10964| 0.12| 2680 | 0.43| 16960|
| Sko90       | 115534          | 115922        | 1625             | 0.91                | 4053  | 0.27| 12698| 0.27| 12698| 0.43| 3822 | 0.45| 28787|
| Sko100a     | 152002          | 153426        | 1905             | 0.94                | 1907  | 0.21| 16608| 0.21| 16608| 0.22| 1486 | 1.30| 309  |
| Sko100b     | 153880          | 155288        | 1494             | 0.85                | 1577  | 0.34| 14729| 0.34| 14729| 0.30| 1405 | 2.34| 274  |
| Sko100c     | 147862          | 148628        | 1525             | 1.15                | 1786  | 0.20| 20311| 0.20| 20311| 0.06| 873  | 1.50| 284  |
| Sko100d     | 149576          | 151196        | 1666             | 0.97                | 3978  | 0.17| 20302| 0.17| 20302| 0.27| 863  | 1.03| 293  |
| Sko100e     | 149150          | 151056        | 2423             | 0.90                | 5415  | 0.24| 21127| 0.24| 21127| 0.33| 745  | 1.55| 301  |
| Sko100f     | 149036          | 150510        | 2038             | 0.91                | 2125  | 0.29| 21479| 0.29| 21479| 0.41| 781  | 0.73| 285  |
### Table 6. Comparison results of the WFA with other algorithms for Nugent’s, Roucairol’s, Scriabin’s, Steinberg’s, Thonemann’s, and Wilhelm’s instances.

| Instances | Best known value | Best solution GRASP | Best results of WFA | GRASP | ANT | GGA | PGA | IFLS | MSA | PHAS |
|-----------|-----------------|---------------------|---------------------|-------|-----|-----|-----|------|-----|------|
| Nug15     | 578             | 578                 | 0.5                 | 0     | 0   | 0   | 0   | 0    | 0   | 0    |
| Nug14     | 1014            | 1014                | 0.8                 | 0     | 0   | 0   | 0   | 0    | 0   | 0    |
| Nug16a    | 1550            | 1550                | 5.8                 | 0     | 5.8 | 0   | 0   | 0.4  | 0   | 0.4  |
| Nug16b    | 1610            | 1610                | 4.1                 | 0     | 4.1 | 0   | 0   | 0.1  | 0   | 0.1  |
| Nug16c    | 1240            | 1240                | 4.7                 | 0     | 4.7 | 0   | 0   | 0.4  | 0   | 0.4  |
| Nug17     | 1712            | 1712                | 5.7                 | 0     | 5.7 | 0   | 0   | 0.4  | 0   | 0.4  |
| Nug18     | 1950            | 1950                | 7.9                 | 0     | 7.9 | 0   | 0   | 0.7  | 0   | 0.7  |
| Nug20     | 2570            | 2570                | 15.7                | 0     | 15.7| 0   | 0   | 0.7  | 0   | 0.7  |
| Rou12     | 235528          | 235528              | 0.5                 | 0     | 0.5 | 0   | 0   | 0.5  | 0   | 0.5  |
| Rou15     | 354210          | 354210              | 1.1                 | 0     | 1.1 | 0   | 0   | 0.6  | 0   | 0.6  |
| Rou20     | 725522          | 725522              | 12.2                | 0     | 12.2| 0   | 0   | 0.9  | 0   | 0.9  |
| Scr12     | 31110           | 31110               | 0.7                 | 0     | 0.7 | 0   | 0   | 0.44 | 0   | 0.44 |
| Scr15     | 51140           | 51140               | 1.0                 | 0     | 1.0 | 0   | 0   | 0.42 | 0   | 0.42 |
| Scr20     | 110090          | 110090              | 5.4                 | 0     | 5.4 | 0   | 0   | 0.89 | 0   | 0.89 |
| Ste36a    | 9526            | 9526                | 623.5               | 0     | 623.5| 1.81| 276 | 0.76 | 295 | 0.27 |
| Ste36b    | 15852           | 15852               | 763.2               | 0     | 763.2| 1.38| 276 | 0.76 | 295 | 0.27 |
| Ste36c    | 8239110         | 8239110             | 800.4               | 0     | 800.4| 0.92| 142 | 0.33 | 321 | 0.06 |
| Tho30     | 149006          | 149006              | 306.2               | 0     | 306.2| 0   | 216 | 0    | 388 | 0.66 |
| Tho40     | 240516          | 240516              | 823.1               | 0.04  | 823.1| 1.17| 184 | 0.66 | 312 | 0.32 |
| Tho150    | 8133398         | 8238058             | 4500.3              | 1.29  | 4500.3|      |     |      | 0.41| 729 |
| Wil50     | 48816           | 48816               | 829.8               | 0.06  | 822.5|      |     |      | 2115| 695 |
| Wil100    | 278308          | 278308              | 8330.7              | 0.34  | 527.6|      |     |      | 20544| 1252 |

Time column in second (s), ∆Time denotes the time advantage of the WFA over the other algorithm.

- **GRASP**: A Parallel Water Flow Algorithm for QAP
Table 7. Comparison results of the WFA with other algorithms for Taillard's instances.

| Instances | Best known value | Random 2-opt | WFA | GRASP | ANT | GSA | PCGA | IFLS | MSA | PHAS |
|-----------|-----------------|--------------|-----|-------|-----|-----|------|------|-----|------|
|           | Best solution   | Time (s)     | ∆   | Time (s) | ∆   | Time (s) | ∆   | Time (s) | ∆   | Time (s) | ∆   |
| Tai12a    | 224416          | 0.3          | 0   | 0.3    | –   | –     | –   | –     | –   | –     | –   |
| Tai12b    | 304624          | 0.2          | 0   | 0.2    | –   | –     | –   | –     | –   | –     | –   |
| Tai15a    | 388214          | 2.0          | 0   | 2.0    | –   | –     | –   | –     | –   | –     | –   |
| Tai15b    | 5176268         | 0.8          | 0   | 0.8    | –   | –     | –   | –     | –   | –     | –   |
| Tai17a    | 4918122         | 4.0          | 0   | 4.0    | –   | –     | –   | –     | –   | –     | –   |
| Tai20a    | 703482          | 175.5        | 0   | 175.5  | 0   | 0     | 160 | –     | –   | –     | –   |
| Tai20b    | 122455319       | 122455319    | 4.8 | 4.8    | –   | –     | –   | –     | –   | –     | –   |
| Tai25a    | 1162756         | 236.4        | 0   | 154.3  | 1.43| 355   | 0.55| 206   | –   | –     | –   |
| Tai25b    | 344355646       | 70.6         | 0   | 70.6   | –   | –     | –   | –     | –   | –     | –   |
| Tai30a    | 1851816         | 437.2        | 0.57| 546.7  | 1.58| 265   | 1.46| 332   | –   | –     | –   |
| Tai30b    | 677171133       | 633.8        | 0.8 | 633.8  | –   | –     | –   | –     | –   | –     | –   |
| Tai35a    | 242002          | 550.9        | 0.59| 1288.4 | 1.90| 531   | 1.64| 232   | –   | –     | –   |
| Tai35b    | 283314545       | 1032.5       | 0   | 1032.5 | –   | –     | –   | –     | –   | –     | –   |
| Tai40a    | 3139370         | 866.1        | 0.67| 866.1  | 2.20| 325   | 2.05| 253   | –   | –     | –   |
| Tai40b    | 67250948        | 1217         | 0   | 1217   | –   | –     | –   | –     | –   | –     | –   |
| Tai45a    | 4938796         | 1113         | 1.46| 2232   | –   | –     | –   | –     | –   | –     | –   |
| Tai45b    | 45832217        | 1592         | 0.02| 1592   | –   | –     | –   | –     | –   | –     | –   |
| Tai50a    | 7205962         | 2165         | 1.55| 3142   | –   | –     | –   | –     | –   | –     | –   |
| Tai50b    | 608215948       | 1866         | 0.65| 1866   | –   | –     | –   | –     | –   | –     | –   |
| Tai60a    | 7205962         | 2165         | 1.55| 3142   | –   | –     | –   | –     | –   | –     | –   |
| Tai60b    | 4938796         | 1113         | 1.46| 2232   | –   | –     | –   | –     | –   | –     | –   |
| Tai60c    | 1851816         | 437.2        | 0.57| 546.7  | 1.58| 265   | 1.46| 332   | –   | –     | –   |
| Tai60d    | 283314545       | 1032.5       | 0   | 1032.5 | –   | –     | –   | –     | –   | –     | –   |
| Tai80a    | 3139370         | 866.1        | 0.67| 866.1  | 2.20| 325   | 2.05| 253   | –   | –     | –   |
| Tai80b    | 67250948        | 1217         | 0   | 1217   | –   | –     | –   | –     | –   | –     | –   |
| Tai100a   | 21052466        | 2450         | 1.76| 820    | –   | –     | –   | –     | –   | –     | –   |
| Tai100b   | 1851816         | 437.2        | 0.57| 546.7  | 1.58| 265   | 1.46| 332   | –   | –     | –   |
| Tai150a   | 4938796         | 1113         | 1.46| 2232   | –   | –     | –   | –     | –   | –     | –   |
| Tai150b   | 7205962         | 2165         | 1.55| 3142   | –   | –     | –   | –     | –   | –     | –   |
| Tai256c   | 44759294        | 7533         | 0.27| 12126  | –   | –     | –   | –     | –   | –     | –   |
Table 8. Improved results of the WFA variants for the QAP instances that have not been optimally solved by 2-opt WFA.

| Instances | Best known value | Random 2-opt WFA | Systematic generator 2-opt WFA | Random 2-opt mirror WFA | WFA-3-opt |
|-----------|------------------|------------------|-------------------------------|------------------------|-----------|
|           | Best solution    | Time (s)         | Best solution                 | Time (s)               | Best solution | Time (s) |
| Lipa50a   | 62093            | 62619            | 62703                         | 1158                   | 62666       | 1507     | 62593   | 971     |
| Lipa60a   | 107218           | 108103           | 108172                        | 2508                   | 108070      | 2754     | 108070  | 2529    |
| Lipa80a   | 253195           | 254853           | 254840                        | 3965                   | 254800      | 3467     | 254800  | 3930    |
| Lipa90a   | 360330           | 362854           | 362906                        | 4423                   | 362673      | 5678     | 362673  | 5680    |
| Sko42     | 15812            | 15836            | 15816                         | 1547                   | 15830       | 1071     | 15816   | 1618    |
| Sko49     | 23386            | 23510            | 23460                         | 1772                   | 23474       | 1868     | 23460   | 1897    |
| Sko56     | 34458            | 34568            | 34528                         | 1742                   | 34558       | 2454     | 34528   | 2134    |
| Sko64     | 48498            | 48796            | 48648                         | 2770                   | 48758       | 2954     | 48648   | 3429    |
| Sko72     | 66256            | 66660            | 66570                         | 4096                   | 66820       | 3425     | 66570   | 3702    |
| Sko90     | 115334           | 116922           | 116968                        | 4385                   | 116632      | 4491     | 116590  | 4053    |
| Sko100b   | 153890           | 155288           | 155728                        | 4388                   | 155398      | 4942     | 155204  | 1577    |
| Sko100c   | 147862           | 149628           | 149862                        | 3072                   | 149900      | 4830     | 149964  | 1786    |
| Sko100d   | 149576           | 151196           | 151186                        | 4230                   | 151022      | 3978     | 151022  | 3897    |
| Sko100e   | 149150           | 151056           | 151140                        | 4162                   | 150588      | 5515     | 150498  | 5415    |
| Sko100f   | 149036           | 150510           | 151032                        | 4060                   | 150794      | 4803     | 150390  | 2125    |
| Tai25a    | 1167256          | 1169030          | 1169030                       | 570                    | 1167256     | 1514     | 1167256 | 1817    |
| Tai30a    | 1818146          | 1825900          | 1828576                       | 547                    | 1830918     | 2571     | 1828576 | 640     |
| Tai35a    | 2422002          | 2436540          | 2436458                       | 1288                   | 2443826     | 2147     | 2436458 | 1508    |
| Tai50a    | 4938796          | 5031472          | 5010798                       | 2323                   | 5026322     | 3595     | 5010798 | 2520    |
| Tai60a    | 7205962          | 7342990          | 7317694                       | 3142                   | 7353798     | 3910     | 7317694 | 3074    |
| Tai80a    | 13511780         | 13821180         | 13700286                      | 2647                   | 1376420     | 5707     | 1376420 | 4123    |
| Tai100a   | 21052466         | 21538854         | 21577638                      | 3897                   | 21422344    | 8120     | 21422344 | 6220 |
| Tai256c   | 44750294         | 44896638         | 44879888                      | 12126                  | 44881948    | 13280    | 44879888 | 15916   |
| Wil50     | 48816            | 48916            | 48856                         | 3365                   | 48846       | 2522     | 48846   | 2892    |
| Wil100    | 273038           | 274446           | 274244                        | 4278                   | 274608      | 4234     | 273980  | 5273    |
Table 9. The average-value-based comparison results of the WFA and the PGA for all the QAP instances.

| Instances     | WFA Δ | Time (s) | PGA Δ | Time (s) |
|---------------|-------|----------|-------|----------|
| Burkard       | 0.000 | 28.14    | 0.006 | 22.97    |
| Christofides  | 0.000 | 23.65    | 3.834 | 2.54     |
| Elshafei      | 0.000 | 15.20    | 2.150 | 44.46    |
| Eschermann    | 0.042 | 91.23    | 0.586 | 104.06   |
| Hadley        | 0.000 | 4.16     | 0.002 | 10.80    |
| Krarup        | 0.000 | 441.83   | 1.190 | 96.97    |
| Li & Pardalos | 1.336 | 1539.66  | 5.104 | 161.72   |
| Skorin-Kapov  | 0.865 | 2573.38  | 0.590 | 1386.65  |
| Nugent        | 0.000 | 29.82    | 0.259 | 26.94    |
| Roucairol     | 0.000 | 4.78     | 0.523 | 0.762    |
| Scriabin      | 0.000 | 2.40     | 0.627 | 0.808    |
| Steinberg     | 0.000 | 745.02   | 2.090 | 160.15   |
| Thonemann     | 0.559 | 1926.51  | 0.650 | 401.51   |
| Wilhelm       | 0.222 | 3972.06  | 0.225 | 973.44   |
| Taillard      | 0.890 | 2514.07  | 0.920 | 320.17   |
| **Average**   | 0.261 | 927.46   | 1.250 | 247.60   |

From Table 8, we can see that the systematic generator 2-opt WFA, the random 2-opt mirror WFA and the WFA-3-opt have obtained better results than the random 2-opt WFA for some of the benchmark instances when the random 2-opt WFA was unable to obtain the best known solution. In particular, these variants of WFA improved the solution quality for 25 instances.

From the best results obtained by WFA in Tables 3 to 7, it can be seen that out of the 134 instances from the QAPLIB [7], the best known solutions for 99 instances have been obtained by the WFA within reasonable computation time. The WFA is also able to obtain solutions with a relative percentage difference of less than 2% for all the remaining instances. The average relative percentage difference of WFA for all the 134 instances is found to be 0.20%.

When compared with other metaheuristic algorithms, namely GRASP, ANT, GGA, PGA, IFLS, MSA, and PHAS, it can be seen from Tables 3 to 7 that WFA outperforms GRASP, ANT, and MSA in all the instances, and also outperforms GGA, PGA, IFLS, and PHAS in many of the instances. When comparing the overall performance using the average relative percentage difference, as well as the number of the best known solutions obtained, the WFA dominates these metaheuristic algorithms as shown in Figure 4, especially when compared with GRASP, ANT, IFLS, and MSA. This shows that the WFA is able to obtain good results when compared to other efficient metaheuristic algorithms.

In addition, we present the comparison results, based on the average value of independent runs, between our algorithm and PGA in Table 9. The results show that the WFA can obtain better overall average solution quality than PGA, although our average computation time is larger than that of PGA.
Figure 4. Comparison of the WFA with other algorithms on (a). Average percentage difference; and (b). The number of the best known solutions obtained.
4. Discussions. The computational results indicate that the WFA can obtain a large number of the best known solutions for the benchmark instances (99 out of 134 instances), a small average percentage difference (0.20% for all 134 instances), and a small maximum percentage difference (less than 2% for all 134 instances), as well as solve the largest instances ($n = 256$) from the QAPLIB [7]. The small maximum percentage difference also affirms the efficiency of the WFA for solving $chrxxx$ and $lipxxx$ instances, in which the state-of-the-art metaheuristic algorithms (e.g., GRASP, ANT, GGA, PGA, IFLS, MSA, and PHAS) have difficulty finding good quality solutions as described in Table 1. The adapted algorithm is thus able to avoid the shortcomings of these metaheuristic algorithms. Using 2-opt mirror and 3-opt neighborhood structures also helps the WFA to exploit the solution space of the QAP more efficiently than the GGA, PGA, and IFLS. With the systematic $DOW$ generator scheme, the WFA spreads its solutions widely onto the solution space that makes a better balance of the solution diversification and intensification capabilities than the GGA, PGA, and IFLS as well. In addition, this helps the WFA obtain a collection of diverse initial solutions in order to avoid the dependence on a constructive heuristic for an initial solution as in the case of GRASP. Moreover, the WFA is able to solve symmetric and non-symmetric QAP instances effectively since the exploration and exploitation phases are constructed without relying on the structure of the problem. Compared with ANT, since the WFA only performs erosion process on promising regions, the exploitation phase saves computation time to find a good quality solution. Also, unlike PHAS, this helps the WFA to avoid having to use the entire solution population. On the other hand, since the WFA does not use a solver to solve the QAP relaxation to construct an initial solution, it does not have to depend on the quality of the solution of the QAP relaxation as in the case of MSA. Thus, with a good balance of solution exploration and exploitation capabilities based on the systematic $DOW$ generator scheme and the efficient local search schemes, the WFA is able to outperform the compared metaheuristic algorithms in the computational results.

However, as in any metaheuristic algorithm, the WFA has some controlled parameters (e.g., the exploration parameters $MaxCloud$ and $MaxPop$, and the exploitation parameters $MaxUIE$ and $MinEro$) that can affect the performance of the algorithm. If the values for the parameters are not chosen properly, the algorithm may spend more computation time to search for better solutions, or converge prematurely to a local optimal solution. To address this issue, an adaptive parameter tuning scheme could be integrated into the WFA in order to adjust the values of the parameters through each cloud or some clouds based on the information of the previous clouds. Then, the WFA may become more efficient for solving the QAP.

5. Conclusions and future work. In this paper, a water flow algorithm with parallel computing is developed for solving the QAP. This algorithm includes the solution exploration and exploitation phases that mimic the water flow in hydrological cycle and erosion process in nature respectively. To solve the QAP by the adapted algorithm with enhanced solution diversification and intensification capabilities, a systematic $DOW$ generator scheme to distribute the positions of $DOW$ is applied, while neighborhood structures, such as the 2-opt mirror and 3-opt, are used to focus on strong searching of promising regions. In addition, a parallel computing strategy is applied to improve the efficiency of computation time for the WFA. The benchmark problem sets from the QAPLIB [7] are used to evaluate the performance
of the WFA. The computational results show that the WFA is able to generate optimal solutions for many benchmark problems of QAP, and near-optimal solutions for the remaining problems. The algorithm is also compared with other metaheuristic algorithms from the literature. The results of the comparison show that the WFA compares favorably with other metaheuristic algorithms used to solve the QAP.

As discussed in the previous section, an area of future research is to improve the choice of the WFA parameters to solve the QAP more efficiently with the possibility of obtaining better quality solutions quickly through integrating an adaptive parameter tuning scheme. This scheme could involve adjusting the parameters of the WFA after some clouds without improvement in best solution obtained so as to increase the solution exploitation capability. One such adjustment is to decrease \( \text{MinEro} \) and increase \( \text{MaxUIE} \) appropriately. An alternative adjustment is to increase \( \text{MaxPop} \) and generate \( \text{DOW}s \) onto the regions in which they exploit weakly through the previous clouds. It is also possible to explore the incorporation of more elements or factors of the hydrological cycle into the WFA, such as evaporation and percolation. Another area of future research is to apply the WFA to solve other types of optimization problems that are similar to the QAP by adapting some of the components of the algorithm appropriately.

Acknowledgments. The authors would like to thank the Editor and the two anonymous reviewers for their valuable and constructive comments on the paper. This work was supported in part by the Academic Research Council, Ministry of Education, Singapore under the following grants: R-266-000-033-112 and R-266-000-033-133.

REFERENCES

[1] R. Abbiw-Jackson, B. Golden, S. Raghavan and E. Wasil, A divide-and-conquer local search heuristic for data visualization, Computers & Operations Research, 33 (2006), 3070–3087.

[2] R. K. Ahuja, J. B. Orlin and A. Tiwari, A greedy genetic algorithm for the quadratic assignment problem, Computers & Operations Research, 27 (2000), 917–934.

[3] E. M. Arkin, R. Hassin and M. Sviridenko, Approximating the maximum quadratic assignment problem, Information Processing Letters, 77 (2001), 13–16.

[4] A. Blanchard, S. Elloumi, A. Faye and N. Wicker, A cutting algorithm for the quadratic assignment problem, INFOR, 41 (2003), 35–49.

[5] S. H. Bokhari, Assignment Problems in Parallel and Distributed Computing, Kluwer Academic Publishers, Boston, MA, 1987.

[6] E. S. Buffa, G. C. Armour and T. E. Vollmann, Allocating facilities with CRAFT, Harvard Business Review, 42 (1964), 136–158.

[7] R. E. Burkard, S. E. Karisch and F. Rendl, QAPLIB - A quadratic assignment problem library, Journal of Global Optimization, 10 (1997), 391–403.

[8] R. E. Burkard, E. Cela, G. Rote and G. J. Woeginger, The quadratic assignment problem with a monotone Anti-Monge and a symmetric Toeplitz matrix: Easy and hard cases, in: Proceedings of the 5th International Conference on Integer Programming and Combinatorial Optimization, Vancouver, British Columbia, Canada, (1996), 204–218.

[9] E. Cela, The Quadratic Assignment Problem: Theory and Algorithms, Combinatorial Optimization (eds. D. Z. Du and P. Pardalos), Kluwer Academic Publishers, London, 1998.

[10] V. M. Demidenko, G. Finke and V. S. Gordon, Well solvable cases of the quadratic assignment problem with monotone and bimonotone matrices, Journal of Mathematical Modeling and Algorithms, 5 (2006), 167–187.

[11] M. Dorigo, Optimization, Learning and Natural Algorithms, Ph.D thesis, Politecnico di Milano, Italy, 1992.

[12] E. Duman and I. Or, The quadratic assignment problem in the context of the printed circuit board assembly process, Computers & Operations Research, 34 (2007), 163–179.
[13] B. Eschermann and H. J. Wunderlich, Optimized synthesis of self-testable finite state machines, in: *Proceedings of the 20th International Symposium Fault-Tolerant Computing (FTCS 20)*, Newcastle Upon Tyne, UK, (1990), 390–397.

[14] H. Eskandar, A. Sadollah, A. Bahreininejad and M. Hamdi, Water cycle algorithm–A novel metaheuristic optimization method for solving constrained engineering optimization problems, *Computers & Structures, 110/111* (2012), 151–166.

[15] R. N. Gasimov and O. Ustun, Solving the quadratic assignment problem using F-MSG algorithm, *Journal of Industrial and Management Optimization, 3* (2007), 173–191.

[16] G. Gutin and A. Yeo, Polynomial approximation algorithms for TSP and QAP with a factorial domination number, *Discrete Applied Mathematics, 119* (2002), 107–116.

[17] J. H. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, 1975.

[18] J. Kennedy and R. C. Eberhart, Particle swarm optimization, in: *Proceedings of IEEE International Conference on Neural Networks*, Piscataway, NJ, (1995), 1942–1948.

[19] L. J. Hubert, *Assignment Methods in Combinatorial Data Analysis*, Marcel Dekker Inc., New York, 1987.

[20] J. Kennedy and R. C. Eberhart, Particle swarm optimization, in: *Proceedings of IEEE International Conference on Neural Networks*, Piscataway, NJ, (1995), 1942–1948.

[21] P. M. Hahn, W. L. Hightower, T. A. Johnson, M. Guignard-Spielberg and C. Roucairol, Tree elaboration strategies in branch and bound algorithms for solving the quadratic assignment problem, *Yugoslav Journal of Operational Research, 11* (2001), 41–60.

[22] H. Q. Saremi, B. Abedin and A. M. Kermani, Website structure improvement: Quadratic assignment problem approach and ant colony metaheuristic technique, *Applied Mathematics and Computation, 195* (2008), 285–298.

[23] H. Shah-Hosseini, Problem solving by intelligent water drops, in: *Proceedings of IEEE Congress on Evolutionary Computation (CEC’ 07)*, Singapore, (2007), 3226–3231.

[24] A. Shukla, A modified bat algorithm for the quadratic assignment problem, in: *Proceedings of IEEE Congress on Evolutionary Computation (CEC’ 07)*, Singapore, (2007), 3226–3231.
[36] H. Shah-Hosseini, The intelligent water drops algorithm: A nature-inspired swarm-based optimization algorithm, *International Journal of Bio-Inspired Computation*, 1 (2009), 71–79.

[37] S. P. Singh and R. R. K. Sharma, Two-level modified simulated annealing based approach for solving facility layout problem, *International Journal of Production Research*, 46 (2008), 3563–3582.

[38] E. Taillard, Comparison of iterative searches for the quadratic assignment problem, *Location Science*, 3 (1995), 87–105.

[39] T. H. Tran and K. M. Ng, A water flow algorithm for flexible flow shop scheduling with intermediate buffers, *Journal of Scheduling*, 14 (2011), 483–500.

[40] T. H. Tran and K. M. Ng, A hybrid water flow algorithm for multi-objective flexible flow shop scheduling problems, *Engineering Optimization*, 45 (2013), 483–502.

[41] K. Y. Wong and P. C. See, A hybrid ant colony optimization algorithm for solving facility layout problems formulated as quadratic assignment problems, *Engineering Computations: International Journal for Computer-Aided Engineering and Software*, 27 (2010), 117–128.

[42] T. H. Wu, S. H. Chung and C. C. Chang, A water flow-like algorithm for manufacturing cell formation problems, *European Journal of Operational Research*, 205 (2010), 346–360.

[43] X. Yang, Q. Lu, C. Li and X. Liao, Biological computation of the solution to the quadratic assignment problem, *Applied Mathematics and Computation*, 200 (2008), 369–377.

[44] F. C. Yang and Y. P. Wang, Water flow-like algorithm for object grouping problems, *Journal of the Chinese Institute of Industrial Engineers*, 24 (2007), 475–488.

[45] Y. J. Zheng, Water wave optimization: A new nature-inspired metaheuristic, *Computers & Operations Research*, 55 (2015), 1–11.

Received May 2017; 1st revision October 2017; 2nd revision December 2017.

E-mail address: kienming@nus.edu.sg
E-mail address: trunghieu.tran@nottingham.ac.uk; ht060324r@alumni.nus.edu.sg