An Ant Colony Optimization Algorithm for Quadratic Assignment Problem

Kun-Chih Wu[†1], Ching-Jung Ting[2] and Lionel Francisco Casanova Gonzalez[3]
Department of Industrial Engineering and Management, Yuan Ze University
135 Yuan Tung Road, Chung-Li, Taiwan 32003
[†]Corresponding author: s968907@mail.yzu.edu.tw[1]
ietingcj@saturn.yzu.edu.tw[2]
s975449@mail.yzu.edu.tw[3]

Abstract - The Quadratic Assignment Problem (QAP) is a well-known and important NP-hard Problem. Many optimization problems are related to QAP, such as facilities layout problem, airport gate assignment problem, and backboard wiring problem. The QAP considers assigning N facilities to N locations, where the cost function is composed by the multiplications of the distance between two locations and flow between two components. The objective is to find the minimum total cost. In this paper, we develop an Ant Colony Optimization (ACO) to solve the problem, in which a specific heuristic information and local search are both well developed. To test the efficiency and effectiveness of our ACO, we run the ACO on the benchmark instances for facilities location problem obtained from the QAPLIB web site. The experiment results show that our ACO can achieve the competing results on the benchmark instances.

Keywords: Quadratic Assignment Problem, Ant Colony Optimization, Heuristics.

1. INTRODUCTION

The quadratic assignment problem (QAP) is an extension from the linear assignment problem (AP), which was first introduced by Koopmans and Beckmann (1957) from the viewpoint of the economic activities. The QAP is considered as assigning N facilities to N locations with given flow of materials between facilities and distances between locations. The objective is to minimize the total multiplication of flows and distances among facilities and locations, respectively. Both AP and QAP are well-known optimization problem. The only difference between these two problems is in their objective functions; the former one can simply be dealt with by polynomial time method but the latter one is an NP-hard problem. Moreover, the QAP is regarded as one of the most difficult NP-hard combinatorial problems (Adams et al., 2007; Loiola et al., 2007). In terms of the objective function, QAP concerns not only relations between a facility and a location, but also interactions among facilities and locations.

To demonstrate the quadratic assignment problem, two types of input parameters are considered: assignment cost c_{ijkl} and allocation cost b_{ij}. Let f_{ik} be the flow between facilities i and k, and let d_{jl} be the distances between locations j and l. The cost c_{ijkl} can be regarded as the multiplications of flows and distances ($c_{ijkl} = f_{ik}d_{jl}$), or it can be any incurred cost while facilities i and k are assigned to locations j and l respectively. The binary variable x_{ij} is the only decision variable involved in the model, and it is defined as:

$$x_{ij} = \begin{cases} 1 & \text{if facilities } i \text{ is assigned to the location } j \\ 0 & \text{otherwise} \end{cases}$$

The quadratic assignment problem can be then generally formulated as a quadratic programming model:

$$Min \quad \sum_{i=1}^{n}\sum_{j=1}^{n} b_{ij}x_{ij} + \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} c_{ijkl}x_{ij}x_{kl} \qquad (1)$$

subject to:

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad 1 \le j \le n \qquad (2)$$

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad 1 \le i \le n \qquad (3)$$

$$x_{ij} \in \{0, 1\} \qquad (4)$$

Since the first paper for QAP (Koopmans and Beckmann, 1957) has been published, many papers for QAP have been proposed due to its practical and theoretical significance. The QAP is a fundamental combinatorial problem related to various practical applications. The vast literature pertaining to QAP has contributed to important developments recently, such as the airport gate assignment problem (Haghani and Chen, 1998), the backboard wiring problem (Steinberg, 1961), the website structure problem (Qahri Saremi et al., 2008), the keyboard layout problem (Dell'Amico et al., 2009), and the most popular applications on the facilities layout problem. Also, several theoretical problems are also related to QAP directly, for example the quadratic 3-dimensional assignment problem, the biquadratic assignment problem, and the multiobjective QAP. Furthermore, some combinatorial problems can be formulated as QAP, for example the traveling salesman problem (Pardalos et al., 1994), the bin-packing problem and the max clique problem. More discussions can be found in the survey provided by Loiola et al. (2007).

Some developments of the methods to solve QAP are based on the exact solution methods. Several exact methods were applied to solve QAP, for example, Brixius and Anstreicher (2001) proposed a branch-and-bound algorithm to solve time-constrained traveling salesman problem. In their branch-and-bound, the quadratic lower bound is applied and different branching strategies are tested to solve the

Proceedings of 12th Asia Pacific Industrial Engineering and Management Systems Conference, Beijing, China, October 14-16, 2011, pp. 138-144.

problem. Their results shown that the maximum problem size of 30 can be solved by the branch-and-bound. Erdoğan and Tansel (2007) applied a branch-and-cut algorithm with a flow-based linearization technique, and their method can solve the problem size up to 25.

Because the QAP is a NP-Hard problem and very difficult to achieve optimality by exact method. In practice, the only feasible way to solve QAP instances with large sizes is to apply heuristic algorithms which can find high quality solutions in reasonable computation times. The solving methods that are most adapted recently to optimization problems are meta-heuristics. The meta-heuristics is a high-level search procedure with guiding other heuristics to search for good solution in feasible domains, and to be able to avoid trapping in local optima. Meta-heuristics have been most generally applied to problems with computational complexity of NP-Hard or NP-Complete. With the growth of meta-heuristics, QAP research has attracted much attention. Several such algorithms have been proposed as shown in Table 1, including simulated annealing (SA), tabu search (TS), Genetic algorithm (GA), greedy randomized adaptive search procedure (GRASP), and scatter search.

The remainder of this paper is organized as follows. In section 2, we describe the proposed ant colony system (ACO) and its implementation steps. Section 3 gives numerical results of tested benchmark instances. Then the conclusions are drawn in section 4.

Table 1 Heuristic Algorithms applied to QAP

Heuristic Algorithms	Scholars	Year
Simulated Annealing (SA)	Arostegui Jr. et al.	2006
	Chwif et al.	1998
Tabu Search (TS)	James et al.	2009
	Mckendall Jr.	2008
	Mckendall Jr. and Jaramillo	2006
Genetic Algorithm (GA)	Misevicius	2004
	Ahuja et al.	2000
	Tate and Smith	1995
Greedy randomized adaptive search procedure (GRASP)	Oliveira et al.	2004
	Aiex et al.	2002
	Mavridou et al.	1998
Scatter Search	Adenso-Diaz et al.	2006
	Greistorfer	2003

2. ANT COLONY SYSTEM FOR THE QAP

The following list and explanation of the symbols are used in our Ant Colony Optimization (ACO) for the Quadratic Assignment Problem algorithm.

n	Population size
q_0	State transition control parameter
P_{ij}	The probability of assigning facility i to site j
Q	Pheromone persistence parameter
U	The set of the available locations/sites
α	Relative importance of pheromone trail
β	Relative importance of local heuristic
η_{ij}	The local heuristic of combination (facility i ,site j)
ρ	Pheromone persistence constant
τ_0	Initial pheromone intensity
τ_{ij}	The pheromone intensity of assigning facility i to site j

Our Ant Colony System (ACS) algorithm for the Quadratic Assignment Problem stems from the ACS algorithm proposed by Gambardella and Dorigo (1997). There are a number of different ACO variants related to the ant colony system. In the classical ACS, an ant will have initially an empty assignment solution and a list of candidate facilities for selection to be put into the specified locations. Then randomly it will assign a facility to a location according to the amount of pheromone on the edge between facilities and sites at the first running time. For successive running times, the ant will progressively select a facility from the candidate list based on the pheromone calculating mechanisms and put it into random locations until completing the solution, i.e. all facilities are assigned to corresponding gates. After assigning the amount of pheromone on the edge, an ant updates the pheromone value locally and globally until reaching the stopping criteria. Moreover, the calculating mechanisms of the "specific heuristic (η)", "Offline updating pheromone rule", Online updating pheromone rule are modified and a "local search (swap)" is applied on the best ant. In this regard, our ACS essentially consists of the following iteration including three main steps:

- Each ant constructs the solution via the state transition rule and updates the pheromone information immediately based on each constructed solution.
- An update of the global pheromone information is done according to the iteration-best solution, i.e. the best solution found in the current iteration.
- A local search is applied to improve the iteration-best solution.

The explanation of the flow chart of our ACO to solve QAP is shown in Figure 1.

2.1 Route Construction

In our ACS, prior to performing the state transition rule, the specific heuristic (η) should be discussed for it will be used in the state transition rule. For this experiment, distances defined as Manhattan distances (symmetrical) are used. A flow matrix which is also symmetrical between facilities i and sites j are also used as initial data. However, the η_{ij} is taken as the savings of combining two nodes i and j on route as contrary to supplying them on two different routes. The η_{ij} is calculated using the following:

$$\eta_{ij} = \frac{1}{d_{ij}} \tag{5}$$

where d_{ij} represents the distance between nodes i and j.

The state transition rule is the key point in running the algorithm, which is a deduction engine in selecting the facilities and putting it into a specified location. To guarantee the speed of finding the best way (convergence), once at a decision point, the ants make their choices based on the pheromone on the assignments and cost of the selective assignments. It is clear that if more ants select the specific allocation, the more pheromone is dropped and this allocation becomes the shortest one. To take advantage completely of the guiding information and to avoid early convergence of finding an assignment, two selection methods are applied. There are two different ways to use the mechanisms in the state transition rule which are divided by a random probability q. One way will occur if the $q \le q_0$ and the other will do likewise if the $q > q_0$ where the q_0 is pre-determined during the parameters initialization. When $q \le q_0$, each facility will be selected based on the following equation:

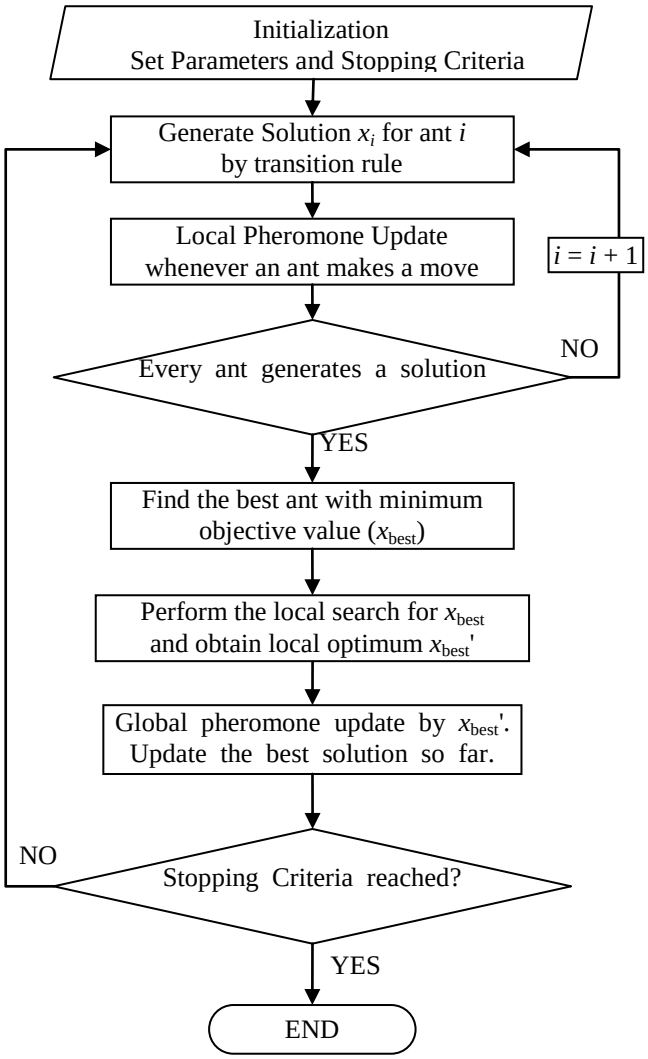

Figure 1 Flow chart for the ACO program for QAP

$$\max_{j \in U} \left[(\tau_{ij})(\eta_{ij})^{\beta} \right] \tag{6}$$

This formula brings forth a very important component to the ACO. In this case, (i, j) represents an edge between point i and j, and τ_{ij} stands for the pheromone on edge (i, j). η_{ij} is the desirability of edge (i, j). q_0 is a user-defined parameter with $(0 \le q_0 \le 1)$, β is the parameter controlling the relative importance of the desirability. U is the set of sites available for facilities i.

Moreover when $q > q_0$ each facility will be selected based on a probability calculating mechanism. This selection mechanism acts similar to the "Roulette Wheel Selection" function. The probability to assign the facility i to the location j is given below:

$$Prob_{i,j} = \frac{(\tau_{ij})(\eta_{ij})^{\beta}}{\sum_{v \in U} (\tau_{iv})(\eta_{iv})^{\beta}} \tag{7}$$

where U is the set of locations. The "roulette wheel" is known as a selection strategy since its mechanism is a simulation of the function of a roulette wheel. Every location has its percentage in the roulette wheel and the bigger this percentage is, the larger the width of slot in the wheel, so the probability of that location also becomes larger.

2.2 Pheromone Updating Rule

The pheromone updating of ACS comprises of the global and local pheromone updating rule. A certain amount of pheromone is deposited when an ant goes by. This can be classified as a continuous process, but it can be regarded as a discrete release by two rules which are the local updating pheromone rule and global updating pheromone rule. The former occurs while the ant is constructing its path, because it modifies the amount of pheromone on the used edges by applying this rule. This simply means that the ant applies this rule that encourages the generation of different solutions to those yet found; that is, it improves the ants' solutions before updating the pheromone trails. This rule also implies that ants don't follow the same path travelled whereby a quick convergence to the solution may occur. Early convergence happens when too many ants gather in a wrong path and the pheromone becomes so dense that a better route cannot be discovered. The local update rule can be formulated as (8).

$$\tau_{ij}^{new} = \rho * \tau_{ij}^{old} + (1 - \rho) * \Delta\tau_{ij} \tag{8}$$

Furthermore, in ACS only the iteration-best ant is permitted to add pheromone after each iteration. The global updating pheromone rule is crucial to guarantee better results. The global pheromone value also referred to as the offline pheromone update is responsible to update the pheromone at the end of the construction process. This offline update is performed only by the best ant; that is, only edges that were visited by the best ant are updated. The offline update rule shows that the feasibility solution has no solution. The

pheromone value is updated globally after all the ants have completed the gates assignment task and local search. The global pheromone updating rule is calculated based on the equation (9).

$$\tau_{ij}^{new} = \rho * \tau_{ij}^{old} + (1 - \rho) * \Delta\tau_{ij} \quad (9)$$

where

$$\Delta\tau_{ij} = 1/f(x_{best}') \quad (10)$$

2.3 Local Search

The mechanism of the local search designed in our ACO is that in the neighborhood of a permutation of feasible solutions is obtained by swapping two facilities' locations. What actually occurs is that it randomly selects two facilities from all facilities, and makes the interchange of their positions. The purpose of the local search is to guarantee an efficient way to improve solutions. That is, if in the neighborhood of the current assignment f(x_{best}) a better assignment is found f(x_{best}'), it replaces the current assignment and the local search is continued from f(x_{best}') as shown in Figure 1. This mechanism is also vital in the overall performance of the ACO.

3. NUMERICAL RESULTS

In order to test our ACO algorithm for the Quadratic Assignment Problem, we test its solution quality, robustness, and computational time in different instances. In this section we will acquire insights into better understanding the characteristics of our heuristic approach. By running the heuristic across Nugent et al. (1968) and Palubeckis' (2000) test problems of various sizes and characteristics, it may be able to show how well our heuristic performs for the QAP. These test problems are also used as benchmarks to test our QAP. Nugent's et al. (1968) problem instances are probably the most used according to the QAPLIB. Palubeckis' (2000) test problems are a new set of QAP instances containing provably optimal values of the objective function which will be discussed further.

3.1 QAP instances

Our proposed ACO for the Quadratic Assignment Problem is coded in Visual C++ 6.0, and run on a personal computer with an AMD Athlon (TM) XP 3800+ (2.20 GHz) CPU and 512 MB RAM, under Windows XP Operating system.

For the QAP we took Nugent et al. (1968) test problems as one of two sets of benchmarks to compare our results obtained from our ACO algorithm. The set of problem instances contains size of 5, 6, 7, 8, 12, 15, 20, and 30 facilities. We used Nugent's instances sizes of 12, 15, 17, 16a, 16b, 17, 18, 20, 21, 22, 24, 25, 27, 28 and 30 found in the QAPLIB. It is known that these problems possess distance matrices stemmed from $n_1 \times n_2$ grid (a generalized quadrangle) and their distances are defined as Manhattan distances between grid points, and multiple global optima is involved in those QAP instances. Another characteristic is that these globally optimal solutions are at the maximum possible distances from other solutions. It is important to note that the solution values for the Nugent's instances were re-found several times, and it is a continuous task to achieve unique results.

Furthermore, we adopted problem provided by Palubeckis (2000) to test our ACO algorithm for larger instances. These QAP instances which can also be found in the QAPLIB (http://www.opt.math.tu-graz.ac.at/qaplib/) are very difficult to solve due to the algorithm used to generate these instances. Palubeckis (2000) explained that the algorithm differed from similar existing generators in that the sizes of the graphs are larger, i.e. affecting the construction of the flow matrix; meaning up to the problem size. Similar to Nugent's problem instances, these QAP instances are also of rectilinear version. To test the robustness of our ACO algorithm, we used Palubeckis' problem sizes of 20, 30, 40, 50, 60, 70, 80, and 100. The results will be discussed in later sections.

3.2 Parameter Sensitivity Analysis

In order to obtain valid and accurate results in our ACO algorithm we tested different parameters to solve the assignment of facilities to sites. We used Nugent et al.'s problem instances 14, 16a, 17, 20, 22, 24, 25, 28 and 30 to find the best parameters. The default value for each parameter is set as $\beta = 2$, $\rho = 0.2$, $q_0 = 0.5$, $b = 10$. The number of iterations (N) set as default was the problem size (n) tested. When one parameter is tested, the others are set as their default values in the experiment. These values tested are $\beta \in \{1, 2, 3, 4, 5\}$, $\rho \in \{0.1, 0.2, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$, $q_0 \in \{0.1, 0.2, 0.4, 0.5, 0.6, 0.8, 0.9\}$, $b \in \{5, 10, 15, n\}$, and $N \in \{n/2, n/3, n/5, n\}$. The average percentage deviations (GAP %) and computational time in seconds over 20 runs are summarized in the tables below. The number of iterations chosen was in accordance to the problem size. In light of this analysis we achieved the following measures of performance in our ACO algorithm:

- Solution quality which measures the lowest value of the objective function (least cost) as an important factor considered in our results.
- Robustness is known as the ability of a heuristic to perform well over a wide range of test problems and is usually captured through measures of variability. We must understand the importance that whenever a heuristic is not consistent, meaning that it gives the best solution in one (or very few instances), but performs very poorly in most other instances is not considered effective.
- Speed of execution is taken as the computational time which is a measurement to study the trade-off between quality of solution and time required to run a case.

In this sense, we have seen that the average percentage

deviations not only depict the performance of the algorithm, but also show the robustness in the same context. For our analysis, we used average percentage deviations; however, the least cost (optimal solution) is adopted in the final solution.

Table 2 portrays the performance of parameter b on the solution quality. We observe that b = 15 provides the best results by the average percentage deviations (Gap %) over the best known solutions (Nugent's benchmarks). Hence, we can observe the consistency of our ACO algorithm provided that it gives best solution on every problem instance tested.

Table 2 Average Gap and CPU time for different b

Problem	b			
	5	10	15	n
14	0.50	0.48	0.72	0.67
16a	0.58	0.68	0.57	0.63
17	0.57	0.59	0.33	0.49
20	0.56	0.60	0.46	0.50
22	0.28	0.22	0.29	0.30
24	2.91	0.80	0.72	0.89
25	0.22	0.27	0.20	0.20
28	0.92	0.86	0.99	1.08
30	0.69	0.66	0.85	0.78
Avg. Gap (%)	0.80	0.57	**0.57**	0.62
Avg. Time (sec)	4.61	4.26	4.20	4.10

Table 3 shows the relative effect of pheromone on distance savings in our ACO algorithm. The impact of parameter β on the solution quality also shows the effectiveness and consistency to obtain optimal solutions ornear optimal solutions as compared with the benchmarks used. The best average percentage deviations is obtained with value β = 3.

Table 3 Average Gap and CPU time for different β

Problem	β				
	1	2	3	4	5
14	1.01	0.48	0.60	0.98	0.74
16a	0.63	0.68	0.54	0.65	0.55
17	0.48	0.59	0.42	0.46	0.67
20	0.55	0.60	0.40	0.50	0.49
22	0.16	0.22	0.32	0.34	0.24
24	0.70	0.80	0.68	0.90	0.86
25	0.27	0.27	0.25	0.22	0.21
28	0.89	0.86	1.06	1.45	1.39
30	0.74	0.66	0.70	0.59	0.74
Avg. Gap (%)	0.60	0.57	**0.55**	0.67	0.65
Avg. Time (sec)	4.13	4.26	4.75	4.61	4.96

Table 4 portrays the impact of parameter ρ on the solution quality. Once more the results of the deposited pheromone discounted by a factor ρ provide us with clear evidence that our ACO algorithm is consistent. The results are optimal values or near optimal values as compared with the benchmark used. The best average percentage deviations is obtained with value ρ = 0.1.

Table 4 Average Gap and CPU time for different ρ

Problem	ρ							
	0.1	0.2	0.4	0.5	0.6	0.7	0.8	0.9
14	0.45	0.48	0.92	1.12	1.12	0.80	1.23	1.16
16a	0.60	0.68	0.52	0.71	0.53	0.75	0.78	0.83
17	0.61	0.59	0.44	0.40	0.47	0.70	0.81	0.54
20	0.58	0.60	0.43	0.58	0.54	0.30	0.52	0.75
22	0.25	0.22	0.40	0.24	0.38	0.42	0.46	0.66
24	0.66	0.80	0.69	0.61	0.83	0.72	0.83	0.93
25	0.34	0.27	0.19	0.19	0.17	0.16	0.11	0.21
28	0.82	0.86	1.11	1.00	1.03	1.15	1.36	1.44
30	0.62	0.66	0.72	0.70	0.55	0.41	0.34	0.54
Avg. Gap	**0.55**	0.57	0.60	0.62	0.62	0.60	0.72	0.78
Avg. Time	3.68	4.26	4.11	3.49	3.63	3.78	3.60	3.86

Table 5 shows the impact of parameter q_0 on the solution quality. The nature of the parameter q_0 determines the relative importance of exploitation versus exploration. With a high value of q_0 we can observe that our ACO algorithm is consistent for the results are optimal values or near optimal values as compared with the benchmark used. The better solutions are found when q_0 = 0.5 giving clear evidence that our ACO algorithm is consistent for the results are optimal values or near optimal values as compared with the benchmark used.

Table 5 Average Gap and CPU time for different q_0

Problem	q_0						
	0.1	0.2	0.4	0.5	0.6	0.8	0.9
14	0.85	0.86	0.63	0.48	1.09	1.19	1.77
16a	0.50	0.76	0.62	0.68	0.91	1.03	1.34
17	0.38	0.49	0.41	0.59	0.47	1.03	1.13
20	0.65	0.68	0.65	0.60	0.70	0.76	1.07
22	0.21	0.17	0.26	0.22	0.22	0.64	0.60
24	0.78	0.80	0.63	0.80	0.70	0.91	1.08
25	0.24	0.29	0.23	0.27	0.27	0.40	0.51
28	0.85	0.75	1.07	0.86	1.03	0.93	1.47
30	0.80	0.62	0.69	0.66	0.60	0.61	0.71
Avg. Gap	0.58	0.60	0.58	**0.57**	0.67	0.83	1.07
Avg. Time	5.67	5.67	6.55	4.26	4.88	5.38	2.94

As a result based on the parameter sensitivity analysis made, the parameters selected for our ACO algorithm in the experiment are: b = 15, β = 3, ρ = 0.1, q_0 = 0.5. The number of iterations (N) used for each run is set as for each problem size tested as shown in Table 6. Consequently, after performing 20 runs, the least costs (optimal values) are obtained and reported.

Table 6 Average Gap and CPU time for different iterations

Problem	iterations			
	n/2	*n*/3	*n*/5	*n*
14	0.98	1.39	2.08	0.48
16a	0.94	1.47	1.89	0.68
17	0.85	1.01	1.25	0.59
20	1.11	1.37	0.60	0.60
22	0.58	0.78	1.02	0.22
24	1.16	1.36	1.92	0.80
25	0.42	0.56	0.71	0.27
28	1.15	1.34	1.65	0.86
30	0.81	0.93	1.09	0.66
Avg. Gap (%)	0.89	1.13	1.36	**0.57**
Avg. Time (sec)	2.32	1.81	1.07	4.26

3.3 The result of Ant Colony System for the QAP

To test our Ant Colony Optimization algorithm for the Quadratic Assignment Problem, we tested our ACO on the benchmark problems of Nugent et al. (1968) and Palubeckis (2000). In many researches, these instances are applied to test their algorithms for efficiency and effectiveness. These problems can be downloaded from the website of QAPLIB.

In our case we used Nugent et al.'s instances from 12 to 30 locations. Also, we run Palubeckis' problem instances of 20, 30, 40, 50, 60, 70, 80, and 100 which are considered to be large instances and very difficult to solve. We compared the results obtained by a multi-start descent (MSD) algorithm by Palubeckis. These problems are never solved to optimality. We found that very high quality solutions in short computation time can be achieved when applying ACO algorithm to solve Nugent's QAP instances. Tables 7 and 8 show the results obtained from our ACO program. Table 9 provides a comparison of our ACO for the Quadratic Assignment Problem with Multi-Start Descent heuristic for the Palubeckis' large problem instances.

4. CONCLUSION

In this research, we developed an ACO algorithm to solve the Quadratic Assignment Problem. An Ant Colony Optimization algorithm is known to be one of the most effective heuristic approaches in present day. ACO is an approximate algorithm used to obtain good enough solutions to hard combinatorial optimization problems in a reasonable amount of computational time. When compared to some of the best heuristics for the QAP, ACO is among one of the best as far as real world, irregular, and structured problems are concerned.

The QAP literature dwells in the fact that such a problem is generally recognized as a very difficult combinatorial optimization problem. To solve the QAP only heuristic algorithms have the capacity to meet such a demand. It is known that exact algorithms can only solve in a reasonable amount of time only small instances of the QAP; that is, with the number of objects less than 30.

Table 7 QAP-ACO results for Nugent's Instances

Prob.	BKS	Best Values	Min GAP	Max GAP	Avg. GAP	Best No*	Time (sec)
Nug12	578	578	0.00	2.08	1.28	1	0.04
Nug14	1014	1014	0.00	2.76	0.96	1	0.10
Nug15	1150	1150	0.00	1.39	0.68	1	0.13
Nug16a	1610	1610	0.00	1.24	0.38	8	0.21
Nug16b	1240	1240	0.00	1.94	0.53	14	0.19
Nug17	1732	1732	0.00	1.04	0.54	1	0.28
Nug18	1930	1930	0.00	1.76	0.97	3	0.43
Nug20	2570	2570	0.00	1.56	0.61	4	0.92
Nug21	2438	2442	0.16	1.23	0.44	0	1.31
Nug22	3596	3596	0.00	1.00	0.46	1	2.30
Nug24	3488	3488	0.00	1.55	0.82	2	3.00
Nug25	3744	3748	0.11	0.53	0.23	0	3.61
Nug27	5234	5234	0.00	2.14	0.78	3	7.80
Nug28	5166	5172	0.12	1.97	1.17	0	9.21
Nug30	6124	6124	0.00	1.37	0.65	2	11.43
Avg.	2774	2775	0.03	1.57	0.70	2.73	2.73

*represents number of times optimal solutions are obtained.

Table 8 QAP-ACO results for Palubeckis' Instances

Prob.	BKS	Best Values	Min GAP	Max GAP	Avg. GAP	Time (sec)
Palu20	81536	81817	0.25	0.46	0.34	1.16
Palu30	271092	272654	0.52	0.63	0.58	10.28
Palu40	837900	840930	0.34	0.39	0.36	62.48
Palu50	1840356	1847422	0.36	0.41	0.38	226.43
Palu60	2967464	2978898	0.37	0.40	0.39	605.83
Palu70	5815290	5832460	0.28	0.31	0.30	1464.32
Palu80	6597966	6618736	0.31	0.32	0.31	3042.79
Palu100	15008994	15048806	0.26	0.27	0.27	6062.05
Avg.	4177575	4190215	0.34	0.40	0.37	1434.42

This study has used the special library called QAPLIB, and has been taken as benchmarks to test the effectiveness and robustness of our ACO algorithm. The standard test problems used were of instances from Nugent et al. (1968) and Palubeckis (2000). The ACO proved to solve both set of test problems to near optimal solutions or optimal solutions. Nugent's problem sizes were solved to optimality. However, we learnt that indeed large problems instances cannot be solved to optimality as described in the QAP literature.

Table 9 MSD Algorithm vs. Our ACO

Problem	MSD	Our ACO	Gap (%)
Palu20	81536	81817	0.34
Palu30	272080	272654	0.21
Palu40	840308	840930	0.07
Palu50	1846876	1847422	0.03
Palu60	2978216	2978898	0.02
Palu70	5831954	5832460	0.01
Palu80	6618290	6618736	0.01
Palu100	15047406	15048806	0.01
Avg.	4189583	4190215	0.02

REFERENCES

[1] Adams, W.P., Guignard, M., Hahn, P. and Hightower, W.L. (2007) A level-2 reformulation-linearization technique bound for the quadratic assignment problem, *European Journal of Operational Research*, 180, 983-996.

[2] Adenso-Diaz, B., Garc á-Carbajal, S. and Lozano, S. (2006) An Empirical Investigation on Parallelization Strategies for Scatter Search, *European Journal of Operational Research*, 169, 490-507.

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

[4] Aiex, R.M., Resende, M.G.C. and Ribeiro, C.C. (2002) Probability distribution of solution time in GRASP: An experimental investigation, *Journal of Heuristics*, 8, 343-373.

[5] Arostegui Jr., M.A., Kadipasaoglu, S.N. and Khumawala, B.M. (2006) An empirical comparison of tabu search, simulated annealing and genetic algorithms for facilities location problems, *International Journal of Production Economics*, 103, 742-754.

[6] Brixius, N.W., Anstreicher, K.M. (2001) Solving quadratic assignment problems using convex quadratic programming relaxations. *Optimization Methods and Software*, 16, 49-68.

[7] Chwif, L., Marcos, R., Barretto, P. and Moscato, L.A. (1998) A Solution to the facility layout problem using simulated annealing, *Computers in Industry*, 36, 125-132.

[8] Dell'Amico, M., D áz, J.C.D., Iori, M., Montanari, R. (2009) The single-finger keyboard layout problem, *Computers & Operations Research*, 36, 3002-3012.

[9] Erdoğan, G. and Tansel, B. (2007) A branch-and-cut algorithm for quadratic assignment problems based on linearization, *Computers & Operations Research*, 34, 1085-1106.

[10] Gambardella, L.M. and Dorigo, M. (1996) Solving Symmetric and Asymmetric TSPs by Ant Colonies, *Proceedings of IEEE Conference on Evolutionary Computation*, 622-627.

[11] Greistorfer, P. (2003) A tabu scatter search metaheuristic for the arc routing problem, *Computers & Industrial Engineering*, 44, 249-266.

[12] Haghani, A., Chen, M.C. (1998) Optimizing gate assignments at airport terminals, *Transportation Research Part A: Policy and Practice*, 32, 437-454.

[13] James, T., Rego, C. and Glover, F. (2009) A cooperative parallel tabu search algorithm for the quadratic assignment problem, *European Journal of Operational Research*, 195, 810-826.

[14] Koopmans, T.C. and Beckmann, M.J. (1957) Assignment problems and the location of economic activities. *Econometrica*, 25, 53-76.

[15] Loiola, E.M., Abreu, N.M.M., Boaventura-Netto, P.O., Hahn, P. and Querido, T. (2007) A Survey for the Quadratic Assignment Problem, *European Journal of Operational Research*, 176, 657-690.

[16] Mavridou, T., Pardalos, P.M., Pitsoulis, L.S. and Resende M.G.C. (1998) A GRASP for the biquadratic assignment problem, *European Journal of Operational Research*, 105, 613-621.

[17] Mckendall Jr., A.R. and Jaramillo J.R. (2006) A tabu search heuristic for the dynamic space allocation problem, *Computers & Operations Research*, 33, 768-789.

[18] Mckendall Jr., A.R. (2008) Improved tabu search heuristics for the dynamic space allocation problem, *Computers & Operations Research*, 35, 3347-3359.

[19] Misevicius, A. (2004) An improved hybrid genetic algorithm: New results for the quadratic assignment problem, *Knowledge-Based Systems*, 17, 65-73.

[20] Nugent, C.E., Vollmann, T.E., Ruml, J. (1968). An experimental comparison of techniques for the assignment of facilities to locations. *Operations Research*, 16, 150–173.

[21] Palubeckis, G. (2000) An Algorithm for Construction of Test Cases for the Quadratic Assignment Problem, *Informatica*, 11, 281-296.

[22] Pardalos, P.M., Rendl, F. and Wolkowicz, H. (1994), *The quadratic assignment problem: a survey and recent developments. In P.M. Pardalos and H. Wolkowicz (eds.), Quadratic assignment and related problems, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Volume 16 (AMS)*, 1-42.

[23] Qahri Saremi, H., Abedin, B., Meimand Kermani, A. (2008) Website structure improvement: Quadratic assignment problem approach and ant colony meta-heuristic technique, *Applied Mathematics and Computation*, 195, 285-298.

[24] Oliveira, C.A.S., Pardalos, P.M. and Resende, M.G.C. (2004) GRASP with path-rethinking for the quadratic assignment problem, *Lecture Notes in Computer Science*, 3059, 356-368.

[25] Steinberg, L. (1961) The backboard wiring problem: A placement algorithm. *SIAM Review*, 3, 37-50.

[26] Tate, D.M. and Smith, A.E. (1995) A genetic approach to the quadratic assignment problem, *Computers & Operations Research*, 22, 73-83.

AUTHOR BIOGRAPHIES

Kun Chih Wu is a Ph.D. student in Industrial Engineering and Management Department at Yuan Ze University, Taiwan. He also received his M.S. degree in Industrial Engineering and Management from the Yuan Ze University.
His primary research interests include packing problem, logistics management, and heuristic algorithms. His email address is <s968907@mail.yzu.edu.tw>

Ching-Jung Ting is an associate professor in the Industrial Engineering and Management Department at Yuan Ze University, Taiwan. He received his B.B.A. from National Chiao Tung University, Taiwan, M.S. in Civil Engineering from Northwestern University, and Ph.D. in Civil Engineering from University of Maryland, College Park, respectively. His primary research interests include supply chain management, logistics management, metaheuristics, and transportation system analysis. His email address is <ietingcj@saturn.yzu.edu.tw>

Lionel F. Gonzalez Casanova is an MBA student at the College of Management/Yuan Ze University. He currently received his M.Sc. in Industrial Engineering and Management at Yuan Ze University. His primary research interests include the Gate Assignment Problem, Innovation Management technology and heuristic algorithms. His email address is <s997137@mail.yzu.edu.tw>