Annals of Operations Research 131, 283–304, 2004

An Ant Colony Optimization Algorithm for the Minimum Weight Vertex Cover Problem *

SHYONG JIAN SHYU ** sjshyu@mcu.edu.tw
Department of Computer Science and Information Engineering, Ming Chuan University, Tao-Yuan 333, Taiwan

PENG-YENG YIN and BERTRAND M.T. LIN
Department of Information Management, National Chi Nan University, Nan-Tou 545, Taiwan

Abstract. Given an undirected graph and a weighting function defined on the vertex set, the minimum weight vertex cover problem is to find a vertex subset whose total weight is minimum subject to the premise that the selected vertices cover all edges in the graph. In this paper, we introduce a meta-heuristic based upon the Ant Colony Optimization (ACO) approach, to find approximate solutions to the minimum weight vertex cover problem. In the literature, the ACO approach has been successfully applied to several well-known combinatorial optimization problems whose solutions might be in the form of paths on the associated graphs. A solution to the minimum weight vertex cover problem however needs not to constitute a path. The ACO algorithm proposed in this paper incorporates several new features so as to select vertices out of the vertex set whereas the total weight can be minimized as much as possible. Computational experiments are designed and conducted to study the performance of our proposed approach. Numerical results evince that the ACO algorithm demonstrates significant effectiveness and robustness in solving the minimum weight vertex cover problem.

Keywords: ant colony optimization, minimum weight vertex cover, meta-heuristic algorithm

1. Introduction

Given an undirected graph $G = (V, E)$ with weights defined on the vertices, the *minimum weight vertex cover* problem is to find a subset of V such that for each edge in E, at least one of its two end vertices belongs to the subset and the sum of the total weight of nodes in the subset is minimum. Formally, given an undirected graph $G = (V, E)$ and a weighting function $w : V \to \mathbb{Z}^+$, the minimum weight vertex cover problem seeks to find a subset $V' \subseteq V$ satisfying $\bigcup_{v \in V'} e(v) = E$, where $e(v)$ denotes the edges incident on vertex v, such that $\sum_{v \in V'} w(v)$ is minimized.

In Karp (1972), the decision version of the minimum vertex cover problem had been shown as NP-complete, even when it is restricted to a unit-weighted planar graph

* This research and the first author are supported by the National Science Council (NSC) of the Republic of China under grant NSC-91-2213-E-130-002. The second and the third author are supported in part by the NSC of the Republic of China under contract numbers 90-2213-E-130-006 and 91-2416-H-260-001, respectively.
** Corresponding author.

with the maximum vertex degree of three. In a theoretical point of view, the vertex cover problem is one of the core NP-complete problems that have been most frequently used for establishing NP-hardness (Garey and Johnson, 1979). In practical considerations, the minimum weight vertex cover problem can be used to model many real-world situations in the areas of circuit design, telecommunications, network flow, just to name a few.

Due to the computational intractability of the minimum weight vertex cover problem (hereafter, acronym MWVC will be used for convenience), many researchers have instead focused their attention on the design of approximation algorithms for delivering quality solutions in a reasonable time. Consider the case where all vertices have the same weight. Since the goal becomes the minimization of the cardinality of a subset V' of V such that for each edge (u, v) in E, at least one of u and v is in V', it is intuitive to successively select the vertex with the largest degree until all of the edges are covered by the vertices in V'. This straightforward heuristic can be further generalized and applied to MWVC. The generalization proposed and analyzed by Chvatal (1979) collects a vertex at each stage with the smallest ratio between its weight and current degree. Clarkson (1983) presented a heuristic algorithm that exhibits a performance guarantee of 2. The randomized algorithm proposed by Pitt (1985) randomly selects an end vertex of an arbitrary edge, say vertex u of (u, v), with a probability inversely proportional to its weight, i.e., $w(v)/(w(u) + w(v))$. They show that the algorithm also reveals a performance guarantee of 2. Dinur and Safra (2001) show that it is impossible to attain approximate solutions to the minimum vertex cover problem within any factor smaller than 1.36067 unless $\mathcal{P} = \mathcal{NP}$. For a comprehensive survey on the analysis of approximation algorithms for MWVC, the reader is referred to Bar-Yehuda and Even (1985), Monien and Speckenmeyer (1985), Motwani (1992), Paschos (1997), Hastad (2001) and Dinur and Safra (2001). Parameterized algorithms, inspired by fixed-parameter intractability theory (Downey and Fellows, 1995a), are also a direction of research for vertex cover problems (Downey and Fellows, 1995b; Balasubramanian, Fellows, and Raman, 1998; Chen, Kanj, and Jia, 1999; Niedermeter and Rossmanith, 1999). Alimonti and Kann (2000) studied another dimension about the APX-completeness of vertex cover problems. As a generalization of vertex cover, the set cover problem also carries the negative results about inapproximability. Status about inapproximability of set cover can be found in Paschos (1997), Feige (1998) and Hastad (2001).

In the optimization literature, meta-heuristics also foster a viable alternative for delivering quality approximate solutions. In this paper, we address the application of a newly developed meta-heuristic approach, the ant colony algorithm, to find approximate solutions to MWVC. Like genetic algorithms, simulated annealing and tabu search, the *Ant Colony Optimization* (ACO) is a meta-heuristic using natural metaphor to solve complex combinatorial optimization problems. It has demonstrated significant successes in dealing with the traveling salesman problem (Dorigo, Maniezzo, and Colorni, 1991; Dorigo and Gambardella, 1997a; Dorigo and Gambardella, 1997b), graph coloring problem (Costa and Hertz, 1997), quadratic assignment problem (Maniezzo, 1998), generalized minimal spanning tree problem (Shyu et al., 2003) and scheduling problems (Jayaraman et al., 2000; Shyu, Lin, and Yin, 2001). Due to the attractiveness about ant

behaviors and simplicity in implementations, ACO has been gaining more and more research attention since the early 1990's. We shall describe the main theme of ACO in the following section.

Unlike most of the optimization problems that have been addressed by ACO, a solution to MWVC does not necessarily constitute a path or a tree on the underlying graph. Therefore, we incorporate specific features into ACO to deal with such situations. The rest of the paper is organized as follows. The main concept and technical structure of ACO will be introduced in section 2. In section 3, we shall design an ACO algorithm to find approximate solutions to MWVC. In section 4, we design and conduct computational experiments to evaluate the performance of the proposed ACO algorithm. Numerical results and analysis are also included. Finally, section 5 gives some concluding remarks.

2. Ant colony optimization

ACO is a family of meta-heuristics that are inspired by the natural optimization mechanism conducted by real ants. Since the development of the first ACO system, called *Ant System* (Dorigo, Maniezzo, and Colorni, 1991), several refined versions have been proposed (Gambardella and Dorigo, 1995; Dorigo, Maniezzo, and Colorni, 1996; Stutzle and Hoos, 1997; Bullnheimer, Hartl, and Strauss, 1999) to improve its performance. For a comprehensive review on ACO, we refer the reader to Colorni, Dorigo, and Maniezzo (1991), Dorigo, Caro, and Gambardella (1999) and Dorigo, Bonabeau, and Theraulaz (2000). The general framework of the ACO systems is presented in figure 1. Basically, a problem under study is transformed into a weighted graph. Then, the ACO system iteratively distributes a set of artificial ants (ants, for short) onto the graph to construct tours corresponding to potential optimal solutions. The optimization mechanism of the ACO system is based on two important features. The first one is the probabilistic *state transition rule* that is applied when an ant is choosing the next vertex to visit (see step 2.2.1 in figure 1). The second one is the *pheromone updating rule* that

```
1.     Initialization
2.     Repeat          /*each iteration at this level is called a cycle*/
2.1.     Each ant is positioned on an arbitrary starting node
2.2.     Repeat        /*each iteration at this level is called a step*/
2.2.1.     Each ant moves to a next node according to the state transition rule
2.2.2.     Apply the local pheromone updating rule
         Until all ants have completed their tours
2.3.     Apply the global pheromone updating rule
       Until end_condition
```

Figure 1. General framework of the ACO system.

dynamically changes the preference degree for the edges that have been traveled through (see steps 2.2.2 and 2.3 in figure 1).

The state transition rule is a random proportional scheme that assigns the probability with which ant k on vertex i chooses vertex j as its next vertex to visit, and this probability can be calculated according to the following equation:

$$p_{ij}^{k} = \begin{cases} 1, & \text{if } q < q_0 \text{ and } j = \arg\max\limits_{r \in A_k} \{\tau_{ir}\eta_{ir}^{\beta}\}; \quad (1) \\ 0, & \text{if } q < q_0 \text{ and } j \neq \arg\max\limits_{r \in A_k} \{\tau_{ir}\eta_{ir}^{\beta}\}; \quad (2) \\ \dfrac{\tau_{ij}\eta_{ij}^{\beta}}{\sum_{r \in A_k} \tau_{ir}\eta_{ir}^{\beta}}, & \text{if } q \geqslant q_0; \quad (3) \end{cases}$$

where A_k is the set of vertices that are open or accessible to ant k, τ_{ij} is the dynamic desirability measure (called *pheromone* in a biological sense) about the access to the edge (i, j), η_{ij} is the static desirability measure about the same edge based on a problem-specific local heuristic, and β is the parameter controlling the relative significance between the two measures. The state transition rule can be triggered as follows. First, draw a random number q from the open interval $(0, 1)$, and if q is less than a specified threshold q_0, the edge with the maximal product $\tau_{ir}\eta_{ir}^{\beta}$ is always selected (see equations (1) and (2)); otherwise, the edge is selected according to the probability given in equation (3). Hence, the state transition rule is a controlled trade-off scheme between the exploitation search and the exploration search of the problem space.

The pheromone updating rule changes the dynamic desirability measure of accessing a specific edge based on the experiences cumulated over time. It can be reinforced using the following form:

$$\tau_{ij} = (1 - \alpha)\tau_{ij} + \alpha\Delta\tau_{ij}, \quad (4)$$

where α is the *evaporation rate* and $\Delta\tau_{ij}$ is the amount of new increment. There are two situations to perform the pheromone updating rule. One is called the *global pheromone updating rule* that is performed at the end of each *cycle* (see step 2.3 in figure 1). In this case, $\Delta\tau_{ij}$ is set to a value that reflects the quality of a completed tour. The other one is called the *local pheromone updating rule* that is performed at the end of each *step* (see step 2.2.2 in figure 1). In this case, $\Delta\tau_{ij}$ is usually constant. The role of the local pheromone updating rule is to monitor the movement pattern of the ants at each step, so we can prevent a specific edge from absorbing most of the ants' attention such that the search process will not get stuck in a local optimal solution.

The ACO approach, like neural networks, genetic algorithms and simulated annealing, is attractive since its optimization scheme is based on natural metaphors. In next section, we will present the application of ACO to MWVC.

3. Application of ACO to MWVC

The characteristics of MWVC are different from those of the problems that have been solved by ACO in the aspect that the solution to MWVC is an unordered subset of vertices, while the solutions to most ACO-solved problems so far are ordered or unordered subsets of edges (Dorigo and Gambardella, 1997a; Dorigo, Caro, and Gambardella, 1999; Shyu et al., 2003). Therefore, for MWVC, ACO should be conducted to explore the power set of the set of vertices instead of that of the set of edges. It becomes more challenging to accomplish the work concerning the transformation from MWVC to an appropriate graph, the development of a constructive local heuristic for the state transition rule, and the design of pheromone updating rules. The details of our implementations are presented in the following subsections.

3.1. Graph representation

Let $G = (V, E)$ denote the graph of some MWVC problem instance, the solution to this instance is an unordered vertex subset $V' \subseteq V$. To guarantee that there always exists a path in G, a sequence of unrepeated adjacent vertices, which covers exactly and only the vertex in V', we construct a complete graph $G_c = (V, E_c)$ involving the vertex set V of G such that every pair of vertices are connected by an edge in E_c. As a result, if we apply ACO on G_c, the ants can construct tours corresponding to any possible variations of V'. To preserve the connectivity information of G in G_c, for each ant k, we define a binary *connectivity* function $\psi_k : E_c \to \{0, 1\}$ as

$$\psi_k(i, j) = \begin{cases} 1, & \text{if edge } (i, j) \in E; \\ 0, & \text{if edge } (i, j) \in E_c - E. \end{cases} \tag{5}$$

Figure 2 gives an illustrative example. The original graph of an MWVC instance is shown in figure 2(a), where the letter in each circle denotes the vertex index and the number beside the circle is the associated weight. The optimal solution is $V' = \{B, D, E\}$

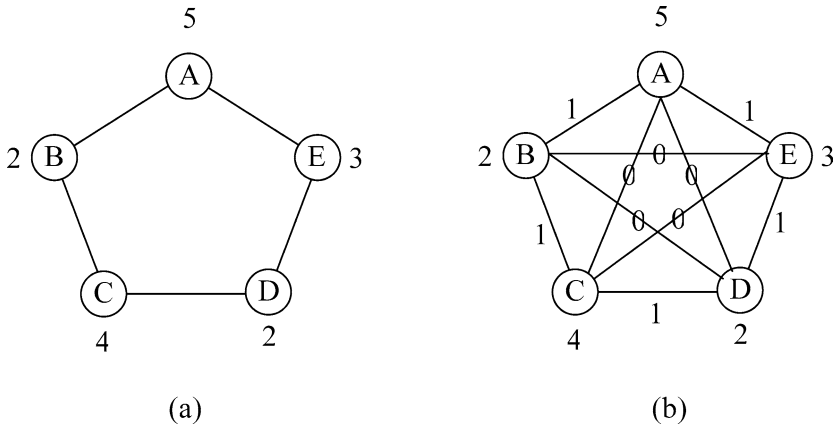

Figure 2. Graph representation of MWVC.

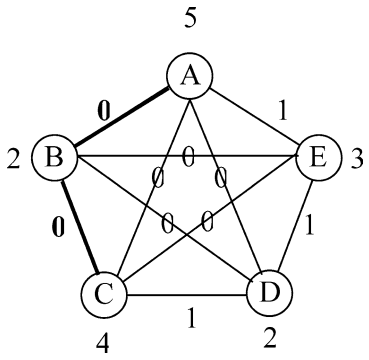

Figure 3. Illustrative example of the connectivity updating rule.

with a total weight of 7. However, there exists no path covering exactly and only the vertices of V'. By using our approach, a complete graph with the same set of vertices can be constructed as shown in figure 2(b), where the number superimposed on the edge indicates the initial value of the ψ_k function. In this new graph, we can find a path $B \rightarrow D \rightarrow E$ that exactly and only covers all the vertices of V'. The connectivity value of each edge is updated using the *connectivity updating rule* to dynamically evaluate the preference for each vertex and to determine whether the tour, a path corresponding to a solution, is completed. This will be described in the next subsection.

3.2. Connectivity updating rule

The connectivity value of an edge in E is set to 0 using the connectivity updating rule when one of its end vertices is accessed by an ant, say, ant k. Formally,

$$\psi_k(i, j) = 0, \quad \text{if edge } (i, j) \in E \text{ and either vertex } i \text{ or vertex } j \text{ is accessed by ant } k. \tag{6}$$

Figure 3 shows such an example where the connectivity values of edge (A, B) and edge (B, C) have been updated to 0 when vertex B is accessed. The purpose of the connectivity updating rule is twofold. First, the access preference for the vertex can be dynamically evaluated. More specifically, $C_j^k = \sum_{(r,j)\in E_c} \psi_k(r, j)$ evaluates the number of edges not covered in E connecting to vertex j. The larger the connectivity value of C_j^k is, the higher the preference for vertex j will be. Second, since the edges in E can be covered by different numbers of vertices in various ways, the ants may need different steps to complete their own tours. By using the connectivity updating rule, the ACO system will know that the tour of ant k has been completed when $C_j^k = 0$ for all j.

Before entering the next cycle, the connectivity values should be all reset using equation (5) to restore the connectivity information of each edge.

3.3. State transition rule

The state transition rule presented in this paper features in the following two aspects. First, the solutions to most of the previous ACO-solved problems are obtained through

the exploration of the power set of the edge set; therefore, the preference information (including pheromone and local heuristic value) is deposited on the edges. Since the solution to MWVC is obtained through the exploration of the power set of the vertex set, the preference information therefore should be stored on vertices. Second, the local heuristic used in most previous literature is static (that is, the value will not change during the optimization process); here, we devise a dynamic heuristic to reflect the reality that the access preference for a vertex changes over time depending on which vertices have been already selected. Based on the above concerns, we propose a state transition rule that defines the probability of choosing vertex j for ant k by

$$p_j^k = \begin{cases} 1, & \text{if } q < q_0 \text{ and } j = \arg\max\limits_{r \in A_k} \{\tau_r \eta_{rk}^\beta\}; \quad (7) \\ 0, & \text{if } q < q_0 \text{ and } j \neq \arg\max\limits_{r \in A_k} \{\tau_r \eta_{rk}^\beta\}; \quad (8) \\ \dfrac{\tau_j \eta_{jk}^\beta}{\sum_{r \in A_k} \tau_r \eta_{rk}^\beta}, & \text{if } q \geqslant q_0, \quad (9) \end{cases}$$

where A_k denotes the set of accessible vertices for ant k.

Note that the probability value p_j^k does not depend on which edge the ant uses to construct the tour, but on which vertex it visits next. It conforms to the fact that the quality of a solution depends only on the vertices selected but not on the order in which these vertices are visited. The value of variable τ_j, which gradually reflects the global preference for vertex j, is updated according to the quality of the final solution constructed in each cycle and will be described in the next subsection. The value of variable η_{jk}, which evaluates the local preference for vertex j for ant k to follow, changes dynamically and is given by

$$\eta_{jk} = \frac{\sum_{(r,j) \in E_c} \psi_k(r, j)}{w(j)}, \quad (10)$$

where η_{jk} is the number of edges not covered by ant k which are currently connected to vertex j divided by the weight of vertex j. Hence, the proposed dynamic local heuristic favors the vertex that has the minimum weight and retains the largest number of edges that are not yet covered.

3.4. Pheromone updating rule

In the proposed system, we apply the global and local pheromone updating rules as follows. First, at the end of each cycle, the intensities of the pheromone left on the vertices of the currently best solution are reinforced. Let vertex subset V_c' be the currently best solution, the pheromone left on vertex i will be updated according to the global updating rule as follows:

$$\tau_i = \begin{cases} (1 - \rho)\tau_i + \rho \Delta \tau_i, & \text{if } i \in V_c'; \\ (1 - \rho)\tau_i, & \text{otherwise,} \end{cases} \quad (11)$$

where

$$\Delta\tau_i = \frac{1}{\sum_{j \in V_c'} w(j)} \tag{12}$$

and $\rho \in (0, 1)$ is a parameter which simulates the *evaporation rate* of the pheromone intensity and enables the algorithm to forget bad decisions previously done.

Second, we activate the local pheromone updating rule to shuffle the solutions and prevent a very significant vertex from being chosen by all the ants. The local updating rule is performed at the end of each step when each ant has chosen a new vertex, say vertex i, the pheromone on the vertex is updated by

$$\tau_i = (1 - \varphi)\tau_i + \varphi\tau_0, \tag{13}$$

where $\varphi \in (0, 1)$ is a parameter adjusting the pheromone previously laid on vertex i and τ_0 is the same as the initial value of pheromone laid on each vertex. Note that the local updating rule decreases the pheromone intensity on the vertex just visited by an ant and makes the selected vertices less attractive to other ants. The effect of the process will direct the exploration session of an ant toward the vertices that have not yet been visited by other ants.

3.5. Stopping criterion

The stopping criterion of ACO could be a maximum number of iterations, a specified CPU time limit, or a given number of consecutive iterations within which no improvement on solutions is attained. Unless otherwise specified, in this paper we use the last one as the stopping criteria, i.e. the algorithm stops when no improvement on solutions is reported within a given number of consecutive iterations.

4. Computational experiments and numerical results

To test the effectiveness of our ACO algorithm for MWVC, we would like to study the following issues.

(1) Does our ACO algorithm effectively generate approximate solutions to MWVC as compared to some well-known heuristics?

(2) Does our ACO algorithm produce quality solutions as such meta-heuristics as tabu search or simulated annealing?

(3) Does the ACO algorithm perform well even when the problem instances of MWVC are quite large?

(4) How does the ACO algorithm behave if the weights and degrees of the vertices are interrelated?

The parameters involved in MWVC, which is represented as a graph $G = (V, E)$ with a weighting function w, include the number of vertices $|V| = n$, the number of

edges $|E| = m$, and the weights on the vertices $w(i)$, $1 \leqslant i \leqslant n$. The parameters of ACO were set as: $u = 10$, $\beta = 1$, $q_0 = \alpha = 0.9$, and $\tau_0 = n(n-a)/C$, where u is the number of ants participating in the experiments, C is the total cost of an initial approximate solution and a is the number of vertices found in this initial approximation. To find an initial approximate solution, we apply the greedy approach (Chvatal, 1979; Motwani, 1992) that iteratively selects the vertex with the minimum ratio between its weight and current degree. The platform of our experiments was a personal computer with an AMD 1.7 GHz CPU and 256 MB RAM. We used Borland C++ Builder as our programming language. The codes are available from the authors.

We design the following experimental settings for each of the above-mentioned issues.

Experiment 1. Comparison with existing heuristics

The number of vertices n was set to be 10, 15, 20 or 25. For each setting of n, we let m range from small to large values so that the degree matrices of the generated graphs could have different degrees of density. The weight $w(i)$ on vertex i was randomly selected from the uniform distribution interval [20, 120], $1 \leqslant i \leqslant n$. We implemented the following heuristics:

(1) GW. The method greedily selects the vertex with the minimum weight.

(2) GD. The method greedily selects the vertex with the maximum degree.

(3) REP (Pitt, 1985). The method randomly selects an end vertex of an arbitrary edge considering the probability inversely proportional to its weight.

(4) GM (Chvatal, 1979; Motwani, 1992). The method greedily selects the vertex with minimum ratio between its weight and current degree.

(5) MGM (Clarkson, 1983). The method greedily selects the vertex with minimum ratio between its weight and current degree where the weight is modified as the heuristic progresses.

(6) ACO. The method we proposed in this paper.

An enumerative algorithm (OPT) that derives the optimal solution was also implemented to provide a comparison basis. The pseudocodes of this enumerative algorithm are described in figure 4.

For each pair of n and m, ten problem instances were randomly generated and tested. The numerical results obtained by applying the six heuristics and the exact algorithm are averaged and tabulated in table 1.

The results listed in table 1 clearly evince that the ACO approach outperforms other heuristics in producing quality approximate solutions. The GM heuristic seems to be the second best. For detail inspections on the performance of the ACO approach, we also kept track of the numbers of hits on the optimal solutions, the average errors encountered and the execution time. The results are shown in table 2. Because other

```
1.        k = 1
2.        cost = ∞
3.        While (k < n)
3.1.        For each combination c, a combination of k vertices out
            of the n original vertices do
3.1.1.        If (total weights of all vertices of c < cost) and
              (vertices of c cover all edges in E)
3.1.1.1.        cost = total weight of all vertices of c
            Endfor
3.2.        k = k + 1
          Endwhile
```

Figure 4. Pseudocodes of the enumerative algorithm for solving MWVC.

Table 1
Solution values of approximate and exact algorithms for small problem instances.

n	m	GW	GD	REP	GM	MGM	ACO	OPT
10	10	509.5	334.9	351.2	299.8	305.6	284.0	284.0
10	20	531.6	437.0	493.9	412.3	430.8	398.7	398.7
10	30	547.3	449.0	540.5	451.8	454.7	431.3	431.3
10	40	564.1	540.0	595.2	521.2	549.2	508.5	508.5
15	20	784.9	528.4	631.1	473.3	500.8	441.9	441.9
15	40	803.5	648.1	760.4	611.2	646.6	574.2	570.4
15	60	895.5	818.5	871.9	770.2	809.1	729.0	726.2
15	80	937.1	880.4	983.3	847.8	889.1	814.6	807.5
15	100	942.3	906.5	969.7	912.1	925.1	880.0	880.0
20	20	1057.8	589.8	666.9	529.9	530.2	473.0	473.0
20	40	1019.4	730.6	915.9	707.1	720.4	661.4	659.3
20	60	1121.6	976.6	1117.1	909.6	927.8	861.8	861.8
20	80	1179.7	987.7	1147.1	959.8	989.3	905.4	898.0
20	100	1294.8	1129.7	1283.4	1069.6	1117.9	1026.8	1026.2
20	120	1258.4	1091.5	1195.2	1072.7	1125.3	1041.5	1038.2
25	40	1274.0	844.2	1063.1	792.7	829.6	756.6	756.6
25	80	1444.0	1091.9	1298.2	1073.5	1148.7	1009.6	1008.1
25	100	1528.1	1236.9	1480.9	1160.4	1225.8	1107.4	1106.9
25	150	1602.5	1347.7	1569.3	1333.4	1410.1	1264.0	1264.0
25	200	1606.0	1480.1	1604.7	1424.0	1466.0	1377.7	1373.4

heuristic approaches usually take less time in delivering solutions, the execution times of other heuristics are not listed.

From the results listed in table 2, we realize that the ACO approach is very successful in delivering optimal solutions to most of the MWVC test cases. That is, our ACO approach performs much better than other heuristics if the quality of the solution

Table 2
Solution qualities of GM and ACO and running efficiencies of OPT and ACO for small problem instances.

		Error ratio		Number of hits		CPU time (sec.)	
n	m	$\frac{\text{GM}-\text{OPT}}{\text{OPT}} \times 100\ (\%)$	$\frac{\text{ACO}-\text{OPT}}{\text{OPT}} \times 100\ (\%)$	GM	ACO	OPT	ACO
10	10	5.56	0.00	4	10	0.002	<0.001
	20	3.41	0.00	6	10	0.002	0.008
	30	4.75	0.00	4	10	0.000	0.003
	40	2.50	0.00	8	10	0.000	0.003
15	20	7.11	0.00	5	10	0.030	0.005
	40	7.15	0.67	3	9	0.059	0.011
	60	6.06	0.39	2	9	0.084	0.008
	80	4.99	0.88	4	8	0.092	0.010
	100	3.65	0.00	4	10	0.094	0.008
20	20	12.03	0.00	2	10	0.755	0.005
	40	7.25	0.32	0	8	2.186	0.016
	60	5.55	0.00	1	10	3.322	0.014
	80	6.88	0.82	3	9	3.944	0.016
	100	4.23	0.06	3	9	4.403	0.016
	120	3.32	0.32	2	7	4.742	0.017
25	40	4.77	0.00	3	10	72.492	0.019
	80	6.49	0.15	1	9	153.028	0.022
	100	4.83	0.05	2	9	175.669	0.025
	150	5.49	0.00	3	10	212.317	0.031
	200	3.68	0.31	4	8	234.927	0.030
Averaged error rate		5.49	0.20				

is of the major concern. Actually, in regard with computation time, the performance of ACO is also impressive. For example, to deal with the last ten instances for $n = 25$ and $m = 200$, the exact algorithm took about 234.927 seconds to find the optimal solution, while ACO with a minor error ratio took only 0.03 seconds in average. In summary, our ACO approach performs well when dealing with small problem instances of MWVC as compared with most of the existing heuristics.

Experiment 2. Comparison with other meta-heuristics

To contrast the performances of our ACO algorithms with some classical meta-heuristics, we implemented simulated annealing (SA) (Kirkpatrick, Gellatt, and Vechi, 1983; Johnson et al., 1989a) and tabu search (TS) (Glover, 1989, 1990) for the MWVC problems. Since GM is the most effective heuristic in experiment 1, we adopt GM to provide the initial solutions for both simulated annealing and tabu search approaches. Once a feasible solution S is found, we replace r, a random number ranging from 1 to $|S|/2$, vertices with larger ratios of the weights over their corresponding degrees, and append a sufficient number of new vertices using the GM heuristic to attain a new feasible solution S'. We consider the derived solution S' as a neighbor of S for both meta-heuristics.

Table 3
Solution qualities and efficiencies of SA and TA for small problem instances.

		Cost		Error rate		Number of hits		CPU time (sec.)	
n	m	SA	TS	$\frac{\text{SA}-\text{OPT}}{\text{OPT}} \times 100\ (\%)$	$\frac{\text{TS}-\text{OPT}}{\text{OPT}} \times 100\ (\%)$	SA	TS	SA	TS
10	10	299.7	299.7	5.53	5.53	4	4	0.016	0.009
	20	407.0	407.0	2.08	2.08	7	7	0.016	0.016
	30	451.8	451.8	4.75	4.75	4	4	0.016	0.017
	40	521.2	521.2	2.50	2.50	8	8	0.016	0.024
15	20	459.7	459.7	4.03	4.03	6	6	0.023	0.023
	40	598.9	596.1	5.00	4.51	4	4	0.031	0.031
	60	763.4	763.4	5.12	5.12	2	2	0.037	0.036
	80	847.4	847.4	4.94	4.94	4	4	0.039	0.038
	100	912.1	912.1	3.65	3.65	4	4	0.041	0.042
20	20	522.5	522.5	10.47	10.47	2	2	0.038	0.039
	40	704.1	704.1	6.80	6.80	0	0	0.044	0.045
	60	899.6	899.6	4.39	4.39	1	1	0.052	0.050
	80	927.9	927.9	3.33	3.33	5	5	0.055	0.055
	100	1067.1	1065.7	3.99	3.85	3	3	0.059	0.058
	120	1072.7	1072.7	3.32	3.32	2	2	0.064	0.063
25	40	792.7	792.7	4.77	4.77	3	3	0.059	0.056
	80	1041.7	1041.7	3.33	3.33	2	2	0.069	0.073
	100	1148.0	1148.0	3.71	3.71	2	2	0.080	0.078
	150	1318.1	1318.1	4.28	4.28	5	5	0.089	0.087
	200	1413.2	1413.2	2.90	2.90	4	4	0.100	0.097
Averaged error rate				4.44	4.41				

To save the paper length, we do not intend to introduce the basics of SA and TS that can be found in such papers as Johnson et al. (1989a, 1989b) and Glover (1989, 1990).

The testing instances used in experiment 1 were adopted here. We summarize the computational results in table 3. Note that each value in table 3 is the averaged result of ten problem instances.

From the results shown in table 3, we know that the performances of SA and TA for solving the MWVC instances are much similar in the solution qualities, the number of hits and the CPU time needed. Comparing the results in tables 2 and 3, we find that the ACO algorithm can produce more attractive solutions than both SA and TS in this experiment. The CPU times needed by the ACO algorithm to convergence are also less than those by SA and TS.

With the above results in hand, we would not intend to claim the superiority of the ACO algorithm over SA or TS in solving the MWVC problem. The performances of SA or TS might be further improved by sophisticated strategies. The performance analysis among such meta-heuristics might need more extensive computational experiments to come up with convincing statistical significance. What we assert in this paper is that the ACO algorithm is an appealing approach to solving the MWVC problem.

Experiment 3. Testing on moderate- and large-scale problems

The parameter setting adopted in this experiment was designed for moderate- and large-scale problems. Concerning the moderate-scale problems, the number of vertices n is 50, 100, 150, 200, 250 or 300. For each setting of n, we let m be ranged from small to large values. The weight $w(i)$ on vertex i, $1 \leqslant i \leqslant n$, was also randomly drawn from the interval [20, 120]. The heuristics were implemented as in experiments 1 and 2 and ten data instances were randomly generated and tested for each pair of n and m. The averaged results are summarized in table 4.

It is clear that the ACO approach still outperforms other heuristics or meta-heuristics for moderate-scale problems. Since GM, SA and TS produce better solutions than other approaches except for our ACO, we subsequently compared the solutions derived by GM, SA, TS and ACO to examine their relative effectiveness. The execution times of SA, TA and ACO are also of concern. The numerical results are shown in table 5.

We can see from table 5 that the approximate solutions found by GM exhibit an average deviation of about 4.12% from those by our ACO approach, while those by SA and TS are about 2.82% and 2.81%, respectively. For the CPU time consideration, ACO in average took about 6.6 seconds (for $n = 300$ and $m = 5000$) to report a solution to MWVC. The elapsed time is relatively small when compared with that needed for deriving an optimal solution. This is a reasonable and acceptable outcome for practical situations.

Concerning large-scale problems, n was tested for 500, 800 and 1000 by using GM, SA, TS and ACO. For each setting of n, we also let m be ranged from small to large values. We also removed the convergence conditions for the meta-heuristics and imposed an execution time limit of 5 minutes for $n = 500$ or 800, and 10 minutes for $n = 1000$. Ten problem instances from the same generation scheme as the above were tested for each pair of n and m. The averaged results are summarized in table 6.

As can be seen from table 6, our ACO approach gives more satisfactory solutions than GM, SA or TS approaches under the same CPU time limits.

Experiment 4. Testing on problems in which the weights and degrees are interrelated

In all previous experiments, the weight on each vertex was randomly drawn from the interval [20, 120]. However, for practical considerations, it would be quite reasonable to assume that the weight on each vertex is proportional to the degree of that vertex. The interrelationship can be justified that a larger degree (more transportation benefits) on a vertex might induce more weight (more running costs) on it. Let the weight $w(i)$ on vertex i be randomly distributed over the interval $[1, d(i)^2]$, where $d(i)$ is the degree of vertex i, $1 \leqslant i \leqslant n$. We consider both small- and moderate-size problems in this experiment.

For the small problem instances, n and m were set as in experiment 1. All of the heuristics implemented in experiments 1 and 2 were examined in this experiment. For the comparison purpose, the optimal solutions to the corresponding data instances were

Table 4
Solution values of approximate heuristics for moderate-scale problems.

n	m	GW	GD	REP	GM	MGM	SA	TS	ACO
50	50	2902.6	1467.4	1681.0	1343.7	1400.5	1339.4	1339.4	1282.1
	100	2977.9	1936.8	2404.8	1858.3	1972.5	1829.6	1826.8	1741.1
	250	3155.1	2503.8	2903.8	2416.0	2570.9	2398.2	2398.2	2287.4
	500	3161.5	2902.4	3095.7	2789.6	2908.3	2741.7	2760.8	2679.0
	750	3316.4	3083.1	3299.4	3052.8	3172.8	3032.0	3032.0	2959.0
	1000	3413.2	3280.5	3436.2	3234.7	3294.0	3234.7	3234.7	3211.2
100	100	6337.5	2926.0	3294.7	2676.7	2752.9	2663.2	2668.3	2552.9
	250	6261.7	4092.1	4814.4	3824.5	4061.4	3765.4	3765.4	3626.4
	500	6567.2	5049.7	5847.7	4896.2	5221.7	4791.8	4793.3	4692.1
	750	6484.9	5440.3	6218.2	5303.0	5626.7	5227.9	5221.7	5076.4
	1000	6830.0	5874.9	6661.3	5749.9	5988.8	5665.0	5649.1	5534.1
	2000	6823.6	6367.6	6758.6	6178.2	6468.1	6159.5	6170.5	6095.7
150	150	9213.2	4277.7	5073.4	3864.1	4010.3	3834.2	3837.4	3684.9
	250	9233.4	5256.2	6537.6	5011.1	5313.8	4953.3	4951.8	4769.7
	500	9620.6	6936.7	8275.4	6543.3	7039.8	6449.0	6452.5	6224.0
	750	9920.1	7605.0	8955.1	7291.3	7734.8	7220.3	7222.4	7014.7
	1000	9919.6	8052.5	9146.2	7747.7	8271.5	7623.9	7614.8	7441.8
	2000	10285.6	9106.4	9992.7	8936.0	9421.4	8767.3	8767.6	8631.2
	3000	10123.2	9322.5	10078.5	9178.9	9557.2	9070.2	9071.7	8950.2
200	250	12446.5	6382.5	7554.5	5887.8	6125.2	5841.7	5841.7	5588.7
	500	12529.1	8137.4	9897.1	7624.3	8087.6	7521.4	7518.6	7259.2
	750	13097.3	9328.8	11117.6	8721.3	9347.0	8590.9	8604.9	8349.8
	1000	13041.8	10075.6	11826.6	9729.2	10260.3	9537.6	9507.7	9262.2
	2000	13737.4	11563.1	12927.2	11346.1	12045.0	11076.8	11076.6	10916.5
	3000	13684.7	12280.5	13414.1	12029.5	12554.0	11829.5	11834.7	11689.1
250	250	15714.1	7194.8	8498.6	6508.3	6877.7	6475.8	6475.0	6197.8
	500	16222.3	9536.7	11460.2	8960.3	9504.4	8874.9	8869.4	8538.8
	750	16594.4	10937.3	13072.7	10344.9	10923.7	10205.6	10211.8	9869.4
	1000	16609.2	11885.3	14137.4	11313.0	12139.9	11183.4	11165.5	10866.6
	2000	16843.6	13936.5	15763.6	13362.9	14045.0	13110.3	13097.4	12917.7
	3000	17074.2	14686.2	16315.6	14291.0	15035.8	14058.0	14056.9	13882.5
	5000	16878.1	15558.1	16655.7	15082.4	15890.2	14957.8	14940.1	14801.8
300	300	18867.6	8414.3	9968.8	7699.2	7998.3	7634.9	7634.0	7342.7
	500	19293.1	10705.6	12898.8	10013.2	10653.1	9844.5	9844.4	9517.4
	750	19535.9	12534.0	14818.7	11700.6	12417.1	11570.5	11565.6	11166.9
	1000	19845.0	13543.8	16079.6	12823.4	13617.0	12618.0	12618.7	12241.7
	2000	20220.5	15899.7	18443.6	15481.9	16425.6	15211.5	15201.5	14894.9
	3000	20374.0	17250.7	19141.9	16495.4	17491.9	16265.5	16243.2	16054.1
	5000	20592.6	18350.4	19990.4	17872.0	18772.2	17681.6	17648.0	17545.4

also derived. We randomly generated and tested ten data instances for each pair of n and m. We summarize the averaged results in table 7.

As we can see from table 7, GM, SA, TS and ACO produce better solutions than other heuristics. Therefore, we record the error ratios and the average numbers of hits for these four approaches in table 8 for further inspections. The CPU times needed by SA, TS, ACO and OPT are also shown in table 8.

Table 5
Relative performances of GM, SA, TA with ACO for moderate-scale problems.

		Solution quality with respect to ACO			CPU time (sec.)		
n	m	$\frac{\text{GM}-\text{ACO}}{\text{ACO}} \times 100\ (\%)$	$\frac{\text{SA}-\text{ACO}}{\text{ACO}} \times 100\ (\%)$	$\frac{\text{TS}-\text{ACO}}{\text{ACO}} \times 100\ (\%)$	SA	TS	ACO
50	50	4.80	4.47	4.47	0.183	0.175	0.063
50	100	6.73	5.08	4.92	0.219	0.212	0.083
50	250	5.62	4.84	4.84	0.272	0.266	0.097
50	500	4.13	2.34	3.05	0.320	0.319	0.102
50	750	3.17	2.47	2.47	0.359	0.358	0.125
50	1000	0.73	0.73	0.73	0.370	0.358	0.117
100	100	4.85	4.32	4.52	0.667	0.631	0.273
100	250	5.46	3.83	3.83	0.828	0.805	0.367
100	500	4.35	2.12	2.16	0.941	0.962	0.433
100	750	4.46	2.98	2.86	1.000	1.020	0.502
100	1000	3.90	2.37	2.08	1.070	1.066	0.456
100	2000	1.35	1.05	1.23	1.253	1.214	0.589
150	150	4.86	4.05	4.14	1.456	1.375	0.691
150	250	5.06	3.85	3.82	1.641	1.556	0.891
150	500	5.13	3.62	3.67	1.892	1.884	1.194
150	750	3.94	2.93	2.96	1.984	1.979	1.042
150	1000	4.11	2.45	2.32	2.069	2.114	1.206
150	2000	3.53	1.58	1.58	2.299	2.400	1.103
150	3000	2.56	1.34	1.36	2.472	2.542	0.966
200	250	5.35	4.53	4.53	2.738	2.599	1.674
200	500	5.03	3.61	3.57	3.181	3.133	2.160
200	750	4.45	2.89	3.06	3.366	3.302	2.602
200	1000	5.04	2.97	2.65	3.489	3.675	2.221
200	2000	3.94	1.47	1.47	3.796	3.944	2.437
200	3000	2.91	1.20	1.25	3.982	4.092	2.497
250	250	5.01	4.49	4.47	4.047	3.821	2.273
250	500	4.94	3.94	3.87	4.717	4.623	4.016
250	750	4.82	3.41	3.47	5.098	5.114	4.047
250	1000	4.11	2.92	2.75	5.253	5.416	3.755
250	2000	3.45	1.49	1.39	5.802	5.924	3.942
250	3000	2.94	1.26	1.26	6.027	6.389	4.276
250	5000	1.90	1.05	0.93	6.361	6.860	3.842
300	300	4.86	3.98	3.97	5.794	5.527	4.322
300	500	5.21	3.44	3.44	6.539	6.527	5.178
300	750	4.78	3.61	3.57	7.067	6.955	6.055
300	1000	4.75	3.07	3.08	7.391	7.222	6.231
300	2000	3.94	2.13	2.06	8.048	8.409	6.488
300	3000	2.75	1.32	1.18	8.372	8.995	6.299
300	5000	1.86	0.78	0.58	8.762	9.122	6.558
Averaged solution quality		4.12	2.82	2.81			

The above results dictate that when the weights on vertices are proportional to the degrees, ACO still demonstrates an impressive performance and even produce optimal solutions in most of the test cases. The performance of the heuristic GM might fluctuate

Table 6
Relative solution qualities of GM, SA, TA with ACO for large-scale problems.

n	m	Solution quality with respect to ACO			CPU time limit (min.)
		$\frac{\text{GM}-\text{ACO}}{\text{ACO}} \times 100$ (%)	$\frac{\text{SA}-\text{ACO}}{\text{ACO}} \times 100$ (%)	$\frac{\text{TS}-\text{ACO}}{\text{ACO}} \times 100$ (%)	
500	500	4.59	3.91	4.18	5
	1000	2.70	2.32	1.83	
	2000	3.86	2.26	2.28	
	5000	3.56	0.79	0.78	
	10000	1.93	0.15	0.07	
800	500	3.88	3.57	3.57	5
	1000	4.60	3.13	3.20	
	2000	3.50	2.06	2.44	
	5000	3.75	1.07	0.66	
	10000	1.41	0.62	0.64	
1000	1000	4.30	3.65	3.65	10
	5000	3.38	1.06	1.41	
	10000	3.17	1.50	0.97	
	15000	2.19	0.75	0.57	
	20000	1.98	0.97	0.95	

Table 7
Solution values of approximate and exact algorithms for small problem instances where the weights were randomly generated from $[1, d(i)^2]$.

n	m	GW	GD	REP	GM	MGM	SA	TS	ACO	OPT
10	10	24.3	26.6	24.8	20.3	20.8	20.1	20.1	18.8	18.8
	20	63.7	69.2	69.9	54.4	59.3	54.4	54.4	51.1	51.1
	30	150.9	159.6	162.9	133.1	137.5	130.9	130.9	127.9	127.9
	40	280.4	292.3	296.0	274.2	280.4	274.2	274.2	268.3	268.3
15	20	50.8	48.1	47.5	39.1	41.2	38.5	38.5	34.7	34.7
	40	205.7	211.5	201.0	182.4	193.9	179.4	179.4	171.5	170.5
	60	423.5	427.2	458.2	384.5	404.2	377.2	377.2	360.8	360.5
	80	771.9	777.1	804.6	709.2	745.0	709.2	709.2	698.7	697.9
	100	1186.3	1215.9	1223.9	1162.8	1174.4	1162.8	1162.8	1137.8	1130.4
20	20	53.8	51.1	47.4	36.7	37.7	35.7	35.7	33.0	32.9
	40	157.1	148.2	147.5	127.5	138.0	126.0	126.0	111.8	111.6
	60	323.7	323.8	336.8	276.2	276.2	273.1	273.8	254.4	254.1
	80	557.5	543.4	545.9	483.7	502.5	475.3	475.3	453.1	452.2
	100	933.5	901.1	940.6	817.6	870.6	810.4	810.4	775.2	775.2
	120	1301.8	1294.7	1309.3	1156.6	1242.4	1156.2	1156.2	1125.5	1123.1
25	40	142.8	136.0	125.7	116.1	120.2	111.9	111.9	98.8	98.7
	80	489.5	480.4	497.3	404.8	423.0	399.9	399.9	373.3	372.7
	100	789.1	746.1	761.9	630.1	644.7	617.5	622.5	595.1	595.0
	150	1475.2	1476.8	1513.1	1334.7	1406.9	1323.4	1323.4	1291.7	1289.9
	200	3013.6	3041.9	3165.0	2831.7	2979.1	2819.8	2819.8	2713.1	2709.5

Table 8
Solution qualities and numbers of hits of GM, SA, TS and ACO and running efficiencies of SA, TS, ACO and OPT for small data instances where the weights were randomly generated from $[1, d(i)^2]$.

		Error ratio				Number of hits				CPU time (sec.)			
n	m	$\frac{GM-OPT}{OPT} \times 100$ (%)	$\frac{SA-OPT}{OPT} \times 100$ (%)	$\frac{TS-OPT}{OPT} \times 100$ (%)	$\frac{ACO-OPT}{OPT} \times 100$ (%)	GM	SA	TS	ACO	SA	TS	ACO	OPT
10	10	7.98	6.91	6.91	0.00	3	4	4	10	0.014	0.014	0.003	<0.001
	20	6.46	6.46	6.46	0.00	3	3	3	10	0.019	0.016	0.003	0.002
	30	4.07	2.35	2.35	0.00	5	6	6	10	0.017	0.017	0.003	0.000
	40	2.20	2.20	2.20	0.00	5	5	5	10	0.023	0.017	0.010	0.000
15	20	12.68	10.95	10.95	0.00	0	0	0	10	0.027	0.032	0.005	0.042
	40	6.98	5.22	5.22	0.59	1	2	2	8	0.033	0.033	0.010	0.067
	60	6.66	4.63	4.63	0.08	3	4	4	9	0.041	0.034	0.008	0.085
	80	1.62	1.62	1.62	0.11	4	4	4	9	0.041	0.036	0.014	0.089
	100	2.87	2.87	2.87	0.65	3	3	3	7	0.041	0.036	0.008	0.092
20	20	11.55	8.51	8.51	0.30	1	1	1	9	0.039	0.042	0.011	1.398
	40	14.25	12.90	12.90	0.18	0	1	1	8	0.047	0.047	0.017	2.673
	60	8.70	7.48	7.75	0.12	1	1	1	9	0.061	0.048	0.016	3.431
	80	6.97	5.11	5.11	0.20	2	2	2	8	0.060	0.056	0.016	4.024
	100	5.47	4.54	4.54	0.00	2	3	3	10	0.063	0.063	0.016	4.419
	120	2.98	2.95	2.95	0.21	2	2	2	8	0.069	0.063	0.017	4.745
25	40	17.63	13.37	13.37	0.10	0	0	0	9	0.081	0.072	0.025	98.436
	80	8.61	7.30	7.30	0.16	0	0	0	8	0.083	0.083	0.026	162.836
	100	5.90	3.78	4.62	0.02	1	1	1	9	0.081	0.081	0.028	180.842
	150	3.47	2.60	2.60	0.14	3	3	3	8	0.094	0.091	0.030	212.399
	200	4.51	4.07	4.07	0.13	2	2	2	9	0.101	0.097	0.030	234.686
Averaged error ratio		7.08	5.79	5.85	0.15								

Table 9
Solution values of approximate heuristics for moderate-scale problems where the weights were randomly generated from $[1, d(i)^2]$.

n	m	GW	GD	REP	GM	MGM	SA	TS	ACO
50	50	149.5	114.8	113.1	95.1	98.7	93.5	93.5	83.9
	100	409.1	377.9	355.0	305.2	312.7	299.9	299.9	276.2
	250	2455.0	2234.8	2319.0	2051.5	2138.4	1990.9	2006.7	1886.8
	500	9149.8	8771.6	9189.4	8196.6	8635.6	8115.5	8115.5	7915.9
	750	22323.9	21568.7	22246.7	20604.9	21676.0	20574.6	20604.9	20134.1
100	50	170.3	95.6	90.3	73.2	73.8	71.7	71.7	67.4
	100	324.2	234.6	224.9	186.1	198.9	183.7	184.1	169.1
	250	1417.8	1221.4	1150.2	995.5	1053.9	986.7	983.9	901.7
	500	5081.0	4515.4	4740.8	3991.8	4307.7	3937.8	3937.8	3726.7
	750	11049.5	10236.1	10564.3	9256.9	9771.9	9172.4	9172.4	8754.5
150	50	208.7	87.0	88.6	71.2	72.6	70.6	70.6	65.8
	100	323.3	211.2	196.5	159.6	165.3	157.9	157.9	144.7
	250	1042.8	880.7	848.5	692.5	735.1	679.5	679.1	625.7
	500	3453.6	3054.2	3148.9	2577.6	2734.9	2519.4	2526.2	2375.0
	750	7957.6	7121.5	7441.0	6236.1	6628.7	6090.8	6105.9	5799.2
200	50	245.5	79.5	74.3	62.0	63.2	61.2	61.2	59.6
	100	356.9	189.1	173.5	146.8	151.0	145.1	145.2	134.7
	250	916.3	697.1	658.2	543.2	576.4	537.0	537.0	488.7
	500	2880.6	2500.7	2368.3	2004.6	2157.8	1989.0	1989.0	1843.6
	750	6070.4	5188.2	5165.7	4422.9	4727.4	4376.4	4383.7	4112.8
250	250	817.1	620.0	602.7	469.1	492.2	462.1	463.1	423.2
	500	2465.9	2012.3	1933.5	1602.6	1697.9	1591.8	1591.8	1457.4
	750	5154.6	4291.7	4332.2	3564.1	3888.6	3512.1	3513.3	3315.9
	1000	8794.9	7627.4	7723.3	6554.6	6954.4	6438.7	6436.0	6058.2
	2000	33411.8	30102.2	31475.8	27360.2	29130.6	26925.4	26864.0	26149.1
	5000	196028.4	186524.1	193232.3	176245.2	183612.8	174037.5	173902.9	171917.2
300	250	861.0	588.7	534.0	447.3	469.9	441.2	441.3	403.9
	500	2208.2	1743.7	1648.1	1361.3	1451.1	1348.6	1347.4	1239.1
	750	4419.2	3659.0	3596.0	2924.2	3121.2	2878.7	2879.4	2678.2
	1000	7375.4	6265.0	6429.0	5274.3	5718.4	5229.9	5227.4	4895.5
	2000	27778.2	25292.9	26106.6	22432.5	23997.8	22061.2	21983.6	21295.2
	5000	166674.8	155972.5	163003.3	147406.4	154929.6	145276.6	145121.4	143243.5

when the characteristics of problem instances vary. The averaged error ratios of GM, SA and TS deteriorate to 7.08%, 5.79% and 5.85%, respectively, as compared to 5.49%, 4.44% and 4.41% that they have exhibited in experiments 1 (table 2) and 2 (table 3). The robustness of ACO over GM, SA and TS is significant in this experiment. Note that the CPU time needed by ACO is generally less than those needed by SA and TS in this experiment.

For the moderate-size instances, the values of n and m were chosen as in experiment 3. Similarly, ten randomly generated data instances were tested for each pair of n and m, where the weight $w(i)$ on vertex i were randomly distributed on the interval $[1, d(i)^2]$. The averaged results are summarized in table 9. Table 10 shows the relative

Table 10
Relative solution qualities of GM, SA, TA with ACO and running efficiencies of SA, TA and ACO for moderate-scale problems where the weights were randomly generated from $[1, d(i)^2]$.

		Solution quality with respect to ACO			CPU time (sec.)		
n	m	$\frac{\text{GM}-\text{ACO}}{\text{ACO}} \times 100\ (\%)$	$\frac{\text{SA}-\text{ACO}}{\text{ACO}} \times 100\ (\%)$	$\frac{\text{TS}-\text{ACO}}{\text{ACO}} \times 100\ (\%)$	SA	TS	ACO
50	50	13.35	11.44	11.44	0.202	0.196	0.072
	100	10.50	8.58	8.58	0.239	0.236	0.097
	250	8.73	5.52	6.35	0.281	0.280	0.111
	500	3.55	2.52	2.52	0.322	0.323	0.120
	750	2.34	2.19	2.34	0.359	0.345	0.111
100	50	8.61	6.38	6.38	0.577	0.547	0.184
	100	10.05	8.63	8.87	0.733	0.702	0.334
	250	10.40	9.43	9.12	0.905	0.878	0.514
	500	7.11	5.66	5.66	0.986	0.953	0.481
	750	5.74	4.77	4.77	1.036	1.027	0.444
150	50	8.21	7.29	7.29	1.144	1.038	0.292
	100	10.30	9.12	9.12	1.417	1.327	0.583
	250	10.68	8.60	8.53	1.845	1.814	1.387
	500	8.53	6.08	6.37	2.000	1.991	1.908
	750	7.53	5.03	5.29	2.078	2.044	1.295
200	50	4.03	2.68	2.68	1.908	1.723	0.463
	100	8.98	7.72	7.80	2.311	2.184	0.981
	250	11.15	9.88	9.88	3.061	2.997	2.413
	500	8.73	7.89	7.89	3.423	3.377	3.423
	750	7.54	6.41	6.59	3.544	3.559	3.600
250	250	10.85	9.19	9.43	4.526	4.327	3.311
	500	9.96	9.22	9.22	5.217	5.219	5.781
	750	7.49	5.92	5.95	5.420	5.483	5.983
	1000	8.19	6.28	6.24	5.561	5.738	6.267
	2000	4.63	2.97	2.73	5.802	6.089	4.859
	5000	2.52	1.23	1.16	6.420	6.616	4.856
300	250	10.75	9.23	9.26	6.076	5.847	5.372
	500	9.86	8.84	8.74	7.211	7.198	9.155
	750	9.19	7.49	7.51	7.620	7.700	10.994
	1000	7.74	6.83	6.78	7.875	7.947	9.045
	2000	5.34	3.60	3.23	8.331	8.616	7.242
	5000	2.91	1.42	1.31	8.816	9.239	6.533
Averaged solution quality		7.98	6.50	6.53			

solution qualities of the approximate solutions found by GM, SA and TS with respect to ACO, and the CPU times needed by SA, TS and ACO. We can see from the results that ACO still shows its relative significance over other heuristics. Note that in the point of view of solution quality, GM, SA and TS give deviations of 7.98%, 6.50% and 6.53% from ACO, respectively, in this experiment, while the corresponding deviations

are 4.12%, 2.82% and 2.81% in experiment 2 (table 5). Since the weights on vertices in this experiment were randomly generated over the interval $[1, d(i)^2]$, a vertex with a larger degree would be cline to having a heavier weight. The heuristic that GM deployed would be less competitive in such a situation as compared to ACO, even when GM was further improved by simulated annealing or tabu search. Hence, the proposed ACO approach for MWVC is not only effective but also robust.

5. Concluding remarks

In this paper, we have addressed a meta-heuristic scheme to compose approximate solutions to MWVC, which is already known as computationally intractable. Although in the literature many heuristics for MWVC with distinguished theoretical error bounds have been proposed, few works addressed the issue about the computational results of the various approaches. In this study, we have designed and implemented an ACO approach for solving MWVC. To apply the ACO approach to meet the problem characteristics, we have proposed a transformation from the original MWVC problem into a graph that is suitable for ants to traverse and compose approximate solutions to MWVC. Through a series of computational experiments, we have obtained numerical results that clearly demonstrate the effectiveness and robustness of the ACO approach.

We have also implemented other meta-heuristics including simulated annealing (SA) and tabu search (TS) in their simple versions to solve MWVC for the comparison purpose. The experimental results show that the performance of the ACO algorithm is better than that of SA or TS in our implementation. However, we would not claim that ACO would be superior to SA or TS in solving MWVC. Detail performance analysis among various meta-heuristics, such as SA, TS, genetic algorithms, neural network, etc., is not the main theme of our study, although it is worthy further investigations. The theme of this paper is to introduce and demonstrate the capability of ACO in dealing with MWVC. Our findings in the mean time broaden the application domain of the ACO approach.

The successful applications of the ACO approach to complex optimization problems are extending the study of meta-heuristics. For further research, it is of potential interest to apply the ACO approach to other problems that are not necessarily based on graphs. Extending the generic ACO model by incorporating specific behaviors of real ants or computer technologies, such as parallel processing, to enhance its problem-solving capability may be another research direction.

Acknowledgments

The authors are grateful to the anonymous referees for their constructive comments that have improved the presentation of the initial version of this work.

References

Alimonti, P. and V. Kann. (2000). "Some APX-Completeness Results for Cubic Graphs." *Theoretical Computer Science* 237, 123–134.

Balasubramanian, R., M.R. Fellows, and V. Raman. (1998). "An Improved Fixed-Parameter Algorithm for Vertex Cover." *Information Processing Letters* 65, 163–168.

Bar-Yehuda, R. and S. Even. (1985). "A Local-Ratio Theorem for Approximating with the Weighted Vertex Cover Problem." *Annals of Discrete Mathematics* 25, 27–45.

Bullnheimer, B., R.F. Hartl, and C. Strauss. (1999). "A New Rank-Based Version of the Ant System: A Computational Study." *Central European Journal of Operations Research and Economics* 7, 25–38.

Chen, J., I.A. Kanj, and W. Jia. (1999). "Vertex Cover: Further Observations and Further Improvements." In *The 25th International Workshop on Graph-Theoretical Concepts in Computer Science*.

Clarkson, K.L. (1983). "A Modification of the Greedy Algorithm for Vertex Cover." *Information Processing Letters* 16, 23–25.

Chvatal, V. (1979). "A Greedy-Heuristic for the Set Cover Problem." *Mathematics of Operations Research* 4, 233–235.

Colorni, A., M. Dorigo, and V. Maniezzo. (1991). "Distributed Optimization by Ant Colonies." In *Proceedings of European Conference on Artificial Life*, Paris, France. Elsevier Science Publishing, pp. 134–142.

Costa, D. and A. Hertz. (1997). "Ants Can Color Graphs." *Journal of the Operational Research Society* 48, 295–305.

Dinur, I. and S. Safra. (2001). "The Importance of Being Biased." Technical Report, Department of Computer Science, Tel Aviv University, Israel.

Dorigo, M., E. Bonabeau, and G. Theraulaz. (2000). "Ant Algorithm and Stigmergy." *Future Generation Computer Systems* 16, 851–871.

Dorigo, M., G.D. Caro, and L.M. Gambardella. (1999). "Ant Algorithms for Optimization." *Artificial Life* 5(3), 137–72.

Dorigo, M. and L.M. Gambardella. (1997a). "Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem." *IEEE Transactions on Evolutionary Computation* 1(1), 53–66.

Dorigo, M. and L.M. Gambardella. (1997b). "Ant Colonies for the Traveling Salesman Problem." *BioSystems* 43, 73–81.

Dorigo, M., V. Maniezzo, and A. Colorni. (1991). "Positive Feedback as a Search Strategy." Technical Report, 91-016, Politecnico idi Milano.

Dorigo, M., V. Maniezzo, and A. Colorni. (1996). "The Ant System: Optimization by a Colony of Cooperating Agents." *IEEE Transactions on Systems, Man, and Cybernetics – Part B* 26(1), 29–41.

Downey, R.G. and M.R. Fellows. (1995a). "Fixed-Parameter Tractability and Completeness I: Basic Results." *SIAM Journal on Computing* 24, 873–921.

Downey, R.G. and M.R. Fellows. (1995b). "Parameterized Computational Feasibility." In P. Clote and J. Remmel (eds.), *Feasible Mathematics*, Vol. II. Boston, pp. 219–244.

Feige, U. (1998). "A Threshold of $\ln n$ for Approximating Set Cover." *Journal of the ACM* 45(4), 634–652.

Gambardella, L.M. and M. Dorigo. (1995). "Ant-Q: A Reinforcement Learning Approach to the Traveling Salesman Problem." In A. Prieditis and S. Russell (eds.), *Proceedings of the Twelfth International Conference on Machine Learning, ML-95*, Tahoe City, CA. Morgan Kaufmann, pp. 252–260.

Garey, M.R. and D.S. Johnson. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. San Francisco, CA: W.H. Freeman.

Glover, F. (1989). "Tabu Search – Part I." *ORSA Journal on Computing* 1(3), 190–206.

Glover, F. (1990). "Tabu Search: A Tutorial." *Interface* 20, 74–94.

Hastad, J. (2001). "Some Optimal Inapproximability Results." *Journal of the ACM* 48(4), 798–859.

Jayaraman, V.K., B.D. Kulkarni, S. Karale, and P. Shelokar. (2000). "Ant Colony Framework for Optimal Design and Scheduling of Batch Plants." *Computers and Chemical Engineering* 24, 1901–1912.

Johnson, D.S., C.R. Aragon, L.A. McGeoch, and C. Schevon. (1989a). "Optimization by Simulated Annealing: An Experimental Evaluation, Part I: Graph Partitioning." *Operations Research* 37, 875–892.

Johnson, D.S., C.R. Aragon, L.A. McGeoch, and C. Schevon. (1989b). "Optimization by Simulated Annealing: An Experimental Evaluation, Part II: Graph Coloring and Number Partitioning." *Operations Research* 39, 378–406.

Karp, R.M. (1972). "Reducibility Among Combinatorial Problems." In R.E. Miller and J.W. Theater (eds.), *Complexity of Computer Computations*. New York: Plenum Press.

Kirkpatrick, S., C.D. Gellatt, and M.P. Vechi. (1983). "Optimization by Simulated Annealing." *Science* 220, 671–680.

Maniezzo, V. (1998). "Exact and Approximate Non-Deterministic Tree-Search Procedures for the Quadratic Assignment Problem." Research Report CSR 98-1, Scienze dell'Informazione, University Di Bologna, Sede Di Cesena, Italy.

Monien, B. and E. Speckenmeyer. (1985). "Ramsey Numbers and an Approximation Algorithm for the Vertex Cover Problem." *Acta Informatica* 22, 115–123.

Motwani, R. (1992). "Lecture Notes on Approximation Algorithms." Technical Report, STAN-CS-92-1435, Department of Computer Science, Stanford University.

Niedermeter, R. and P. Rossmanith. (1999). "Upper Bounds for Vertex Cover Further Improved." In *Proc. of STACS'99*, Lecture Notes in Computer Science, Vol. 1536, pp. 561–570.

Paschos, V.T. (1997). "A Survey of Approximately Optimal Solutions to Some Covering and Packing Problems." *ACM Computing Surveys* 29(2), 171–209.

Pitt, L. (1985). "A Simple Probabilistic Approximation Algorithm for Vertex Cover." Technical Report, YaleU/DCS/TR-404, Department of Computer Science, Yale University.

Shyu, S.J., B.M.T. Lin, and P.Y. Yin. (2001). "Application of Ant Colony Optimization for No-Wait Flowshop Scheduling Problem to Minimize the Total Completion Time." Presented at the *INFORMS Meeting*, Maui, Hawaii, June.

Shyu, S.J., P.Y. Yin, B.M.T. Lin, and M. Haouari. (2003). "Ant-Tree: An Ant Colony System for the Generalized Minimum Spanning Tree Problem." *Journal of Experimental & Theoretical Artificial Intelligence* 15(1), 103–112.

Stutzle, T. and H. Hoos. (1997). "Improvements on the Ant System: Introducing MAX-MIN Ant System." In *Proceedings of the International Conference on Artificial Neural Networks and Genetic Algorithms*, Springer, pp. 245–249.