Ant Algorithm Applied in the Minimal Cost Maximum Flow Problem*

Min Xie, Lixin Gao, and Haiwa Guan

Institute of Operations Research and Control Science,
Wenzhou University, Zhejiang, 325000, China
lxgao@wzu.edu.cn

Abstract. The minimal cost maximum flow problem is a classical combinational optimization problem. Based on the characteristic of ant algorithm and the minimal cost maximum flow problem, a graph mode is presented to use the ant algorithm to solve the minimal cost maximum flow problem. Simulation results show that the algorithm can efficiently solve minimal cost maximum flow problem in a relatively short time.

Keywords: ant algorithm; minimal cost maximum flow problem; directed network.

1 Introduction

The minimal cost maximum flow problem is an important part of network flows, which is the classical combinational optimization problem with many applications such as transportation problem, scheduling problem, etc. Recently, it has been applied in some new domains, such as coding network and wireless ad hoc networks. There are several algorithms which can solve minimal cost maximum flow problem, such as, Cycle-canceling algorithm, Successive shortest path algorithm, Primal-dual algorithm[1]. With the rapid development in the network services, the minimal cost maximum flow problem has recently become a hot spot.

Ant algorithm is first proposed by Dorigo M etc in 1990s, which utilize the similarity of food-seeking behavior of real ants and traveling salesman problem (TSP) and imitate the process of food-seeking behavior of real ants to solve TSP[2]. Ant algorithm has been applied successfully in many applications such as traveling salesman problem[3,4], quadratic assignment problem[5], job-shop scheduling problem[6] and the optimal path planning problem[7]. Actually, ant colony algorithm has been a major concern issue, we attempt to use ant colony algorithm to solve minimum cost maximum flow problem in this paper.

The minimal cost maximum flow problem is a linear programming problem which has close relation with graph theory. The minimal cost maximum flow problem has a structure which is fit to use the ant colony algorithm. In this paper, we attempt to use

* This work was supported by National Nature Science Foundation of China under Grant 60674071.

D.-S. Huang et al. (Eds.): ICIC 2008, LNAI 5227, pp. 201–208, 2008.

ant colony algorithm to solve minimum cost maximum flow problem. The remaining part of the paper is organized as follows. In section 2, we introduce the principle of ant colony algorithm. In section 3, we describe minimum cost maximum flow problem and transform the model of minimum cost maximum flow problem based on the characteristic of ant colony algorithm, then use ant colony algorithm to solve it. The computational experiments and results are given in section 4. Finally, we give the summary and the forecast.

2 Ant Algorithm[8-9]

Ant algorithm imitates the behavior of a colony of ants to solve problems. For example, it has been observed that a colony of ants is able to find the shortest path to a food source by marking their trails with a chemical substance called pheromone. As an ant moves and searches for food, it lays down pheromone along its path. As it decides where to move, it looks for pheromone trails and prefers to follow trails with higher levels of pheromone. The ant will lay a higher concentration of pheromone over its path if it takes the shorter path.

The operation of ant system can be illustrated by the classical traveling salesman problem. A traveling salesman problem is seeking for a round route covering all cities with minimal total distance. More formally, TSP can be represented by a complete weighted directed graph $G = (V, E)$ with n nodes, $V = \{1, 2, ..., n\}$ being the set of nodes, $E = \{(i, j)\}$ being the set of arcs. Suppose there are n cities and m ants. The probability that city j is selected to be visited immediately after city i can be written in a formula as follows:

$$p_{ij}^{k} = \begin{cases} \dfrac{[\tau_{ij}]^{\alpha} \cdot [\eta_{ij}]^{\beta}}{\sum_{s \in allowed_k} [\tau_{is}]^{\alpha} \cdot [\eta_{is}]^{\beta}} & \text{if} \quad j \in allowed_k \\ 0 & \text{else} \end{cases} \tag{1}$$

Where $allowed_k = \{0,1,...,n-1\} - tabu_k$ is the set of cities that have not been visited yet, $tabu_k$ is the set of cities that have been visited, τ_{ij} is the intensity of pheromone trail between cities i and j. $\eta_{ij} = 1/d_{ij}$ is the visibility of city j from city i, d_{ij} is the distance between cities i and j, α is the parameter to regulate the influence of τ_{ij} and β is the parameter to regulate the influence of η_{ij}.

This selection process is repeated until all ants have completed a tour. For each ant the length generated is calculated and the best tour found so far is updated. Then the trail levels are updated as follows: on a tour each ant leaves pheromone quantity given by Q/L_k, where Q is a constant and L_k is the length of its tour. Therefore there is more pheromone left per unit length on shorter tours. By analogy to nature, part of the pheromone evaporates, i.e. the existing pheromone trails are reduced by a factor $(1-\rho)$ before new pheromone is laid. This is done to avoid early convergence and is

regulated by a parameter ρ. The updating of the trail level τ_{ij} can be written in a formula as follows:

$$\tau_{ij}(t+n) = \rho \times \tau_{ij}(t) + \Delta\tau_{ij} \quad \rho \in (0,1) \tag{2}$$

$$\Delta\tau_{ij}^{k} = \begin{cases} \dfrac{Q}{L_k}, & \text{if ant k travels on edge } (i, j) \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

where t is the iteration counter, $\rho \in [0,1]$ is the parameter to regulate the reduction of τ_{ij}, $\Delta\tau_{ij} = \sum_{k=1}^{m} \Delta\tau_{ij}^{k}$, $\Delta\tau_{ij}^{k}$ is the increase of trail level on edge (i, j) caused by ant k, $\Delta\tau_{ij}$ is the total increase of trail level on edge (i, j), L_k is the tour length of ant k, m being the number of ants, $\tau_{ij}(t)$, $\Delta\tau_{ij}(t)$, $p_{ij}^{k}(t)$ can be expressed in different forms, decided according to specific issue, $\tau_{ij}^{k}(0) = C$ (cons tan t), $\Delta\tau_{ij}^{k} = 0$ $(i, j = 0,1,...,n-1)$.

3 Description of Problem and the Application of Ant Algorithm

3.1 Description of Problem

The minimal cost maximum flow problem is based on directed network. We consider a capacitated network $D = (V, A)$ with a nonnegative capacity c_{ij} and a nonnegative cost b_{ij} associated with every arc $(v_i, v_j) \in A$. To define the minimal cost maximum flow problem, we distinguish two special nodes in the network D, a source node v_s and a sink node v_t. Other nodes are called the intermediate nodes. Generally, this kind of network can be written as $D = (V, A, C, B)$. For each arc (v_i, v_j), v_i is called a predecessor of v_j, and v_j is called a successor of v_i. We wish to find the minimal cost maximum flow from the source node v_s to the sink node v_t that satisfies the arc capacities and mass balance constraints at all nodes. The minimal cost maximum flow problem can be stated as follows:

$$\text{Minimize } b(f) = \sum_{(v_i, v_j) \in A} b_{ij} f_{ij}$$

Subject to

$$\sum_{(v_i, v_j) \in A} f_{ij} - \sum_{(v_k, v_i) \in A} f_{ki} = \begin{cases} v(f) & (i = s) \\ 0 & (i \neq s, t) \\ -v(f) & (i = t) \end{cases} \tag{4}$$

$$0 \le f_{ij} \le c_{ij} \quad \text{for each } (v_i, v_j) \in A \tag{5}$$

We refer to a vector $f=\{f_{ij}\}$ satisfies (4) and (5) as a flow and the corresponding value of the scalar variable $v(f)$ as the value of the flow.

In order to facilitate and simplify the question, we only consider the situation which capacity and flow are integer. For the case the capacity and the flow are non-integer, we can choose a unit flow, and select a function g_{ij} for each arc (v_i, v_j) which causes it rapid convergence in the suboptimal solution.

3.2 Ant Algorithm Applied in Minimum Cost Maximum Flow Problem

We transform the minimum cost maximum flow problem as follows:

1. Store the nodes which connect with source node into the table S;
2. For each intermediate node, we select one arc which connects to intermediate node but not to source node, then store the arcs into the table P (to satisfy the constraints better, we select the arcs whose capacity are large enough)
3. Connect the arcs according to the arc subscript order(traverse the nodes according to order from infancy to maturity, then to each node traverse the arcs which connect to them according to order from infancy to maturity),then renumber the nodes with $u_1, \cdots u_t$ according to connection successively order, where the number t is one more than the quantity of table $A-P$;
4. For each arc which are renumbered, produce $c_{ij}+1$ virtual arcs according to capacity c_{ij}, the capacity of virtual arcs starts from 0 then adds 1 in turn. The flow f_{ij} of arc (v_i, v_j) is the integer of $[0, c_{ij}]$, which are chosen according to following probability choice formula:

$$p_{ijk}=\frac{[\tau_{ijk}]^{\alpha}}{\sum_{s\in[0,c_{ij}]}[\tau_{ijs}]^{\alpha}} \qquad k\in[0,c_{ij}] \tag{6}$$

where p_{ijk} is the probability of flow of arc (v_i, v_j) equals to k, τ_{ijk} being the intensity of pheromone trail between nodes i and j equals to k.

5. Calculates the flow of table P:

If arc (v_i, v_j) is the output arc, then

$$f_{ij}=\sum_{(v_k,v_i)\in A} f_{ki}-\sum_{\substack{(v_i,v_l)\in A\\ l\neq j}} f_{il} \tag{7}$$

If arc (v_i, v_j) is the input arc, then

$$f_{ij}=\sum_{(v_j,v_k)\in A} f_{jk}-\sum_{\substack{(v_l,v_j)\in A\\ l\neq i}} f_{lj} \tag{8}$$

Minimum cost maximum flow problem is to find a maximum flow f whose cost function $b(f)=\sum_{(v_i,v_j)\in A} b_{ij}f_{ij}$ reach the minimum. Capacities and balance constraints

can be transformed as each arc (v_i, v_j) in table P which satisfy $0 \le f_{ij} \le c_{ij}$, then minimum cost maximum flow problem can be transformed as following model:

$$F = \min \sum_{(v_i, v_j) \in A} b_{ij} f_{ij} \tag{9}$$

$$s.t. \quad 0 \le f_{ij} \le c_{ij} \quad (v_i, v_j) \in P \tag{10}$$

Now, we illustrate the step 4 in detail, if the capacity of arc (v_i, v_j) equals to 4, then generate 5 virtual arcs, the flow of arc are 0,1,2,3,4 independently.

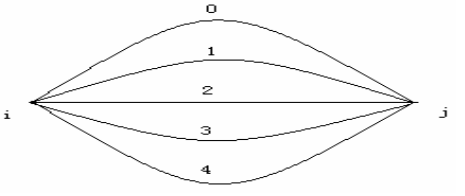

Fig. 1. Virtual path

To solve the minimum cost maximum flow problem, we can imitate real ant's behavior, set m ants in the source node, each ant use probability choice formula (6) to choose the quantity of flow in table S, until traverse all arcs in table S, then calculate the output flow of source node $v(f) = \sum_{(v_s, v_j) \in A} f_{sj}$, if its value is smaller than the value in the previous iteration (the first iterative compares to initial value), then redistribute the arc flow, until its value is bigger than the value in the previous iterative; then use probability choice formula (6) again to choose arc flow until traverse the table $A - S - P$, then calculate arc flow in table P according to formula (7) or (8). If arc flow of table P satisfy formula (10) and the value of objective function is better than the value in previous iterative (the first iterative compares to initial value) then update the trail level as follow formula

$$\tau_{ijk}(t+n) = \rho \times \tau_{ijk}(t) + \Delta\tau_{ijk} \quad \rho \in (0,1) \tag{11}$$

$$\Delta\tau_{ijk} = \begin{cases} \dfrac{Q}{F}, & \text{the flow of arc } (V_i, V_j) \text{equals to } k \\ 0, & \text{ot her s} \end{cases} \tag{12}$$

where F is the value of the objective function

The iteration is repeated until some termination conditions are met, such as a maximum number of iterations has been performed or not get better solution continual h times

If the capacity c_{ij} and arc flow f_{ij} are non-integer, then make the corresponding revision to the step 4 in 3.2, generate $\lfloor c_{ij}/\Delta \rfloor + 1$ virtual arcs according to capacity

c_{ij}, the capacity of virtual arcs starts from 0 then add Δ in turn. The smaller the Δ is, the more precise the solution is.

Algorithm 1

Step 1. Transform minimum cost maximum flow problem according to 3.2

Step 2. (parameter initial) Set $nc = 0$ initial maximum iteration NC, initial flow F_0 and initial cost B_0, set ant number m, set m ants in the source node, $\tau_{ij}(0) = C$, $\Delta\tau_{ij}(0) = 0$

Step 3. Choose arc flow in table S according to formula (10) until traversing all arcs in table S

Step 4. Calculate the output flow $v(f) = \sum_{(v_s, v_j) \in A} f_{sj}$. If $v(f) < F_0$, then go to step 3;

Step 5. Choose arc flow in table $A - S - P$ according to formula (6) until traversing all arcs in table $A - S - P$, then calculate arc flow in table P according to formula (7) or (8)

Step 6. Calculate arc flow in table P, if $0 \le f_{ij} \le c_{ij}$, then calculate the value of objective function according to formula (9), if its value is better than the value of previous iteration, record the current best solution, $F_0 = v(f)$, $B_0 = F$, else go to step 3, if it is not met the constraint $0 \le f_{ij} \le c_{ij}$ continual h times, end the iteration and output the result

Step 7. Update the best arcs according to global updating formula (11)

Step 8. Set $\Delta\tau_{ij} \leftarrow 0, nc \leftarrow nc + 1$ to all arcs

Step 9. If nc is smaller than maximum iteration, go to step 3, else end the iteration and output the result.

4 Numerical Examples

In order to confirm the validity of ant colony algorithm in the minimum cost maximal flow problem, we select the examples of reference [10] and reference [1] to show as figure 2 and Figure 3. The simulation carries on PC machine with the MATLAB. Compare the result which is obtained in this paper with the result which is obtained from [10] and [1]. The selected parameters of examples are shown in Table 1.

Table 1. Parameter setting

α	ρ	C	Q	m	NC	h	F_0	B_0
1	0.2	1	50	20	1000	30	2	1000

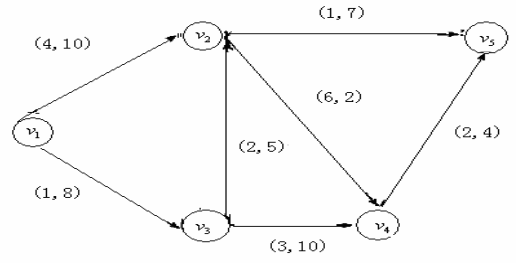

Fig. 2. Directed network

Example 1. Consider the minimum cost maximum flow problem shown in figure 2, where vertex v_1 is a source node, vertex v_5 is a sink node, the number of arc is (b_{ij}, c_{ij}), where b_{ij} shows the cost of unit flow of arc (v_i, v_j), c_{ij} shows the capacity of arc (v_i, v_j)。

Simulation result is

$v(f) = 11$, $F = 55$ $f_{12} = 3$, $f_{13} = 8$, $f_{24} = 0$, $f_{25} = 7$, $f_{32} = 4$, $f_{34} = 4$, $f_{45} = 4$.

We can find the minimum cost is 55 and the maximum flow is 11. The simulation result is the same as labeling algorithm in reference [10].

Example 2. Consider the minimum cost maximum flow problem shown in figure 3, where vertex v_1 is a source node, vertex v_7 is a sink node, the number of arc is (b_{ij}, c_{ij}), where b_{ij} shows the cost of unit flow of arc (v_i, v_j), c_{ij} shows the capacity of arc (v_i, v_j).

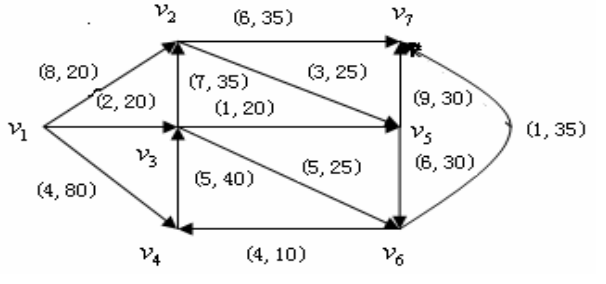

Fig. 3. Directed network

Simulation result is

$v(f) = 80$, $F = 1347$ $f_{12} = 20$, $f_{13} = 20$, $f_{14} = 40$, $f_{25} = 15$, $f_{27} = 34$, $f_{32} = 29$, $f_{35} = 9$, $f_{36} = 22$, $f_{43} = 40$, $f_{56} = 11$, $f_{57} = 13$, $f_{64} = 0$, $f_{67} = 33$.

We can obtain the minimum cost is 1347 and the maximum flow is 80. The simulation result is the same as the result getting by dual algorithm in reference [1].

The examples 1 and 2 show that the ant colony algorithm is feasible to solve the minimum cost and maximum flow problem.

5 Conclusion

For the minimal cost maximum flow problem, we analyze the characteristic of the minimal cost maximum flow problem, then transform the minimal cost maximum flow problem correspondingly, and use ant colony algorithm to solve the minimal cost maximal flow problem. The simulation results show that the ant colony algorithm can solve the minimum cost maximal flow problem efficiently. In order to facilitate and simplify the question, the scale of simulation selected in this paper is small. This algorithm is also effective to solve the more complex problem. To large capacity situation, we can increase the value of Δ to reduce the complexity of algorithm, but the precision of algorithm will be brought down. As the situation of non-integer and how to choose the unit flow to balance the complexity of algorithm and precision of algorithm will be the content in our further study.

References

1. Ravindra, K., Ahuja, T., James, B.: Network Flows: Theory, Algorithms, and Applications. China machine press, Beijing (2005)
2. Dorigo, M.: Optimization, Learning and Natural Algorithms, Milano Italy Dipartimento di Elettronica, Politecnico di Milano, Italy (1992)
3. Dorigo, M., Gambardella, L.: Ant Colony System: A Cooperative Learning Approach to the Traveling salesman Problem. IEEE Transactions on Evolutionary Computation 1(1), 53–66 (1997)
4. Gambardella, L., Dorigo, M.: Ant-Q: A Reinforcement Learning Approach to the Traveling Salesman Problem. In: Proceedings of the 12th International Conference on Machine Learning, pp. 252–260. Morgan Kaufmann, Tahoe City (1995)
5. Maniezzo, V., Colorni, A., Dorigo, M.: The Ant System Applied to the Quadratic Assignment Problem, IRIDIA/94-28. Belgium, Universite de Bruxelles (1994)
6. Colorni, A., Dorigo, M., Maniezzo, V., et al.: Ant System for Job-shop Scheduling. Belgian Journal of Operations Research, Statistics and Computer Science 34, 39–53 (1994)
7. Xie, M., Gao, L.: Ant Algorithm Applied in Optimal Path Planning. Computer Engineering and Application (2008)
8. Li, S., Chen, Y., Li, Y.: Ant Colony Algorithms with Applications. Harbin Industry University Press, Harbin (2004)
9. Duan, H.: Ant Colony Algorithm: Theory and Applications. Science Press, Beijing (2005)
10. Ning, X.: A Mixed Labelling Algorithm for Solving Minimum Cost Flow Problem in the Network. Systems Engineering-theory & Practice 5(3), 1 (1990)