Comput. Opns Res. Vol. 15, No. 6, pp. 577–584, 1988
Printed in Great Britain.

0305-0548/88 $3.00 + 0.00

A NEW ALGORITHM FOR THE DIRECTED CHINESE POSTMAN PROBLEM

YAXIONG LIN[1,*] and YONGCHANG ZHAO[2,†]

[1] Department of Industrial and Management Systems Engineering, Arizona State University, Tempe, AZ 85287, U.S.A.
[2] The Systems Engineering Research Institute, Shanghai Institute of Mechanical Engineering, Shanghai, China

(*Received December* 1986; *revised March* 1988)

Scope and Purpose—The Chinese Postman Problem (CPP) is to find a least cost way to traverse each arc of a network at least once and to return to the origin. Since its first appearance in *Chinese Mathematics* 1, 273–277 (1962), the CPP has been the focus of much research attention and found an abundance of applications, especially in municipal problems, such as the routing of street sweepers, household refuse collection vehicles, postmen, the spraying of roads, etc. This paper presents an algorithm for the "directed" Chinese Postman Problem, which, compared with the existing algorithms, is easier to be implemented and has a better computational complexity.

Abstract—The directed Chinese Postman Problem can be transformed into a minimum cost flow problem over a derived network, or be transformed into a transportation problem in which the coefficients are determined by a shortest path problem over an extended network. Based on the Complementary Slackness Theorem, this paper will present an algorithm for the directed Chinese Postman Problem, which can be applied directly to the original network and, compared with the existing algorithms, has a better computational complexity $O(kn^2)$ where k depends on the structure of the network. The algorithm is extended to the directed m-CPP case.

1. INTRODUCTION

The Chinese Postman Problem (CPP) is to find a least cost way to traverse each arc of a network at least once and to return to the origin. The original CPP was addressed by Kwan in 1962 [7]. Since then, it has been the focus of many researches. Several algorithms have been introduced and several problems have been derived from the original CPP [6], including the rural postman problem (RPP), the capacitated CPP (CCPP), etc. [8].

The CPP is divided into three categories: the directed CPP, undirected CPP, and the mixed CPP. The directed CPP is the CPP set up on the directed network. Similarly, the undirected CPP and the mixed CPP are defined. This paper is contributed to the directed CPP.

The path that traverses all arcs in the network at least once is a postman tour. To find an optimal postman tour on the network, one way is to find the Euler tour on the network if the network contains a Euler tour, or to enlarge the network by duplicating some of the arcs and then find the Euler tour on the enlarged network.

Edmonds and Johnson have proposed a procedure for finding the Euler tour (postman tour) on the Euler network [4]. With this procedure, we need only to provide an enlarged network, which is the Euler network and has the minimum total arc duplication cost.

The existing algorithms for the directed CPP involve the construction of an extended network which is related to, but differs from, the original network. The optimum solution to the directed CPP corresponds to a solution on the extended network. For example, in the Edmonds and Johnson's algorithm [4], it corresponds to the minimum cost flows on the network, while in the Beltrami and Bodin's algorithm [1], it corresponds to the solution of the transportation network on which the weights attached to the arcs are equal to the distance of the shortest path between the two corresponding nodes on the original network. The computational complexities of two

*Yaxiong Lin, currently the Ph.D. student in the Department of Industrial & Management Systems Engineering, Arizona State University, received his B.S. in 1983 and M.S. in 1986 from the Department of Systems Engineering, Shanghai Institute of Mechanical Engineering, Shanghai, China. Research interests include: graph theory, networks, mathematical programming, modeling and simulation. This work was done when he was at Shanghai Institute of Mechanical Engineering.

†Yongchang Zhao, professor of the Systems Engineering Research Institute, Shanghai Institute of Mechanical Engineering. Research interests include: graph theory, networks, mathematical programming, circuits & systems, signal flow graph and its applications, transportation planning.

algorithms are $O(n^3)$ and $O(m * n^2)$, respectively. Keeping these in mind, we will present an algorithm that can be applied directly to the original network and has a better computational complexity.

The algorithm is developed on the basis of the Complementary Slackness Theorem in the linear programming. Section 2 studies the formulation of the directed CPP: the linear formulations and its dual formulation. The Optimality Conditions are then derived. Section 3 shows the procedures of the algorithm. An example is given for illustration. In section 4, a brief discussion on the optimality of the algorithm and the proof of the calculation of the cost are given. Section 5 gives the computational complexity analysis of the presented algorithm. The algorithm's extension to the m-CPP is also discussed in section 5.

2. FORMULATIONS AND THE OPTIMALITY CONDITIONS

It has been proved that in the directed network, a postman tour exists if and only if the network is strongly connected, therefore, it is assumed that such a condition exists. In addition, it is assumed that all arc weights on the network are nonnegative.

Let $N(V, A)$ denote the directed network, where V is the node set and A the arc set. Let $d(i, j)$ denote the weight of arc (i, j) in the network N.

As stated before, the directed CPP will be solved just by giving an enlarged Euler network. An integer programming is formulated to obtain the number of duplicates of the arc to be included in the enlarged Euler network. Let $f(i, j)$ represent the number of arc (i, j) that must be included in the enlarged Euler network. $f(i, j)$ is equivalent to the number of times the arc (i, j) is traversed and again is equivalent to the flow on the arc (i, j) in [4]. As the result, $f(i, j)$ should be greater or equal to 1. Due to its cycle structure, the postman tour is subject to the flow conservation condition at each node, which is best expressed as the even and symmetric network structure. The integer programming is given below:

$$\text{Min} \sum_{(i,j)\in A} f(i, j)\, d(i, j) \tag{P}$$

such that

$$\sum_{j\in V} f(i, j) = \sum_{j\in V} f(j, i), \qquad i \in V \tag{1}$$

$$f(i, j) \geqslant 1 \text{ and integer}, \qquad (i, j) \in A. \tag{2}$$

The objective is to minimize the total weight in the enlarged network. Equation (1) is called the node constraint or the flow conservation constraint. It guarantees that the enlarged network contains a Euler tour. Equation (2), called the arc constraint, guarantees that each arc is traversed at least once in the postman tour.

Let $p(i)$ be the dual variable of the ith constraint in the node constraint and $R(i, j)$ the dual variable of arc (i, j) in the arc constraint. The dual formulation of (P) is given below:

$$\text{Max} \sum_{(i,j)\in A} R(i, j) \tag{DP}$$

such that

$$-p(i) + p(j) + R(i, j) \leqslant d(i, j), \qquad (i, j) \in A \tag{3}$$

$$R(i, j) \geqslant 0, \qquad (i, j) \in A \tag{4}$$

$$p(i) \text{ free}, \qquad i \in A \tag{5}$$

Given the solution is feasible [subject to equations (1)–(5)], the conditions that guarantee the solution to be optimal can be obtained by applying the Complementary Slackness Theorem to the formulations (P) and (DP). These conditions are derived and shown below:

$$\begin{cases} [-p(i) + p(j) - d(i, j)] f(i, j) = 0, & (i, j) \in A \quad (6) \\ [f(i, j) - 1] R(i, j) = 0, & (i, j) \in A. \quad (7) \end{cases}$$

The necessary and sufficient conditions for the solution to be both feasible and optimal to (P) and (DP) are obtained by combining equations (1)–(7). The resulting equations are referred to as the Optimality Conditions:

$$\text{Optimality Conditions (OC)}\begin{cases} R(i,j) = d(i,j) + p(i) - p(j), & (i,j)\in A & (8)\\ \forall (i,j)\in A: \quad \text{if } f(i,j)=1, \quad \text{then } R(i,j)\geqslant 0 \\ \qquad\qquad\quad \text{if } f(i,j)\geqslant 2, \quad \text{then } R(i,j)=0 & & (9)\\ \sum\limits_{j\in V} f(i,j) = \sum\limits_{j\in V} f(j,i), & i\in V, & (10)\\ f(i,j)\geqslant 1 \text{ and integer}, & (i,j)\in A. & (11) \end{cases}$$

Any solution satisfying the Optimality Conditions (OC) is an optimal solution to the directed CPP. In the next section, we will develop an algorithm to generate the solution that satisfies the (OC).

3. ALGORITHM FOR THE DIRECTED CPP

In this section, the detailed algorithm will be given. The algorithm starts with an initial solution which is not necessarily feasible but should satisfy equations (8)–(10) in the (OC). The initial solution is then modified at each iteration until the Optimality Conditions are satisfied.

The formulation (P) represents a minimum cost flow problem in which the flow on arc (i,j) corresponds to the value of $f(i,j)$ in (P). The arc (i,j) is called feasible if $f(i,j)\geqslant 1$, otherwise, it is unfeasible $[f(i,j)=0]$.

The initial solution was set to $f(i,j)=0$ (zero flow) and $R(i,j)=d(i,j)$ for all $(i,j)\in A$ and $p(i)=0$ for all $i\in V$. The optimal solution contains only the feasible arcs because of equation (11) in (OC). Therefore, the unfeasible arcs must be changed into the feasible arcs before the optimal solution is obtained.

Since equation (10) must hold throughout the iterations, the flow of the arc (i,j), or $f(i,j)$, cannot be changed arbitrarily. A closed arc chain, here we refer to as a cycle, is found for changing the flow of the arc. The flow of the arc on the cycle can be changed by sending a unit of flow along the cycle, that is, increase $f(i,j)$ by 1 if arc (i,j) is along the orientation of the cycle or decrease $f(i,j)$ by 1 if arc (i,j) is against the orientation of the cycle. It can be proved that equation (10) still holds after this flow adjustment.

The cycle can be found with respect to an unfeasible arc. For example, given the unfeasible arc (t,s), the cycle is composed of $SP(s,t)$ which is the shortest path from node s to node t, and the arc (t,s) itself. The cycle is denoted by $C(t,s)$.

The network on which the shortest path is obtained has the same structure as the original network, except the weights of the arcs. The weights need to be calculated at each iteration due to the changes of dual variables of the nodes, say $p(i)$.

In order to ensure that the algorithm will stop within the limited number of iterations, the arcs are changed only from unfeasible into feasible, that is, $f(i,j)$ will not be decreased when it is equal to 0 or 1.

The algorithm is described as follows with an illustration of the algorithm using the example network shown in Fig. 1 [1].

Step 1: *Set the initial solution:* $f(i,j)=0$, $R(i,j)=d(i,j)$, $w(i,j)=0$, *for all* $(i,j)\in A$ *and* $p(i)=0$ *for all* $i\in V$.

The initial solution for the example network is shown in Fig. 2, where the numbers in the parentheses attached to arc (i,j) represent $f(i,j)$, $w(i,j)$, and $R(i,j)$, respectively.

Step 2: *Check the status of the arcs.*

If all arcs are feasible, the algorithm stops and the optimal enlarged Euler network is found. Else, select an unfeasible arc, say (t,s), *and go to Step 3.*

For the initial solution, all arcs are unfeasible. Select arc (1, 2). In Fig. 2, it is denoted by $(t,s)=(1,2)$.

Step 3: *Using the $w(i, j)$ as the weight of the arcs, find the shortest path from node s to node i, $SP(s, i)$, $i \in V$.*
Form the cycle (t, s) by combining arc (t, s) and $SP(s, t)$. The orientation of the cycle is set to the direction of arc (t, s).
For all arcs $(i, j) \in C(t, s)$, do:
(1) If (i, j) is along the orientation, then $f(i, j) = f(i, j) + 1$;
(2) If (i, j) is against the orientation, then $f(i, j) = f(i, j) - 1$.
The Dijkstra's shortest path algorithm is applied to obtain $SP(s, i)$, $i \in V$. Let $d(i)$ = length of $SP(s, i)$.
For example, the cycle obtained at the iteration 1 (Fig. 2) is $C(1, 2) = 1\text{–}2\text{–}3\text{–}4\text{–}1$. $d(1) = d(2) = \cdots = d(5) = 0$, since $w(i, j) = 0$ for all $(i, j) \in A$. The flows of the arcs on the cycle, namely $f(1, 2)$, $f(2, 3)$, $f(3, 4)$, and $f(4, 1)$, are increased from 0 to 1.

Step 4: *For all $(i, j) \in A$, calculate:*

$$R(i, j) = min\{d(i), d(t)\} + R(i, j) - min\{d(j), d(t)\}; \tag{12}$$

if $f(i, j) = 0$, then set $w(i, j) = 0$ and $w(j, i) = \infty$;
if $f(i, j) = 1$, then set $w(i, j) = R(i, j)$ and $w(j, i) = \infty$;
if $f(i, j) \geq 2$, then set $w(i, j) = w(j, i) = 0$.
Go to Step 2.
From equation (12), we know that it is not necessary to calculate those $d(i)$ $(i \neq t, i \in V)$ that $d(i) \geq d(t)$.
At iteration 1, all $R(i, j)$ remain unchanged: $R(i, j) = d(i, j)$, since $d(i) = 0$ for all $i = 1, 2, \ldots, 5$; all $R(i, j)$ are not changed; $w(1, 2) = R(1, 2) = 2$, $w(2, 3) = R(2, 3) = 1$, $w(3, 4) = R(3, 4) = 3$, $w(4, 1) = R(4, 1) = 5$, and $w(1, 5) = w(3, 5) = w(5, 4) = 0$. The iteration 1 is done.
The results of iterations 2 and 3 are shown in Figs 3 and 4, respectively. After 3 iterations, all arcs are already in feasible status (Fig. 5), thus the procedure stops. The enlarged Euler network is shown in Fig. 6.

It will be proved that the total cost of the postman tour is determined by the dual variables of the arcs by the following formula:

$$\text{Total-Cost} = \sum_{(i,j) \in A} R(i, j). \tag{13}$$

In the above example, the total cost is calculated as follows:

$$\begin{aligned} \text{Total-Cost} &= R(1, 2) + R(1, 5) + R(2, 3) + R(3, 4) + R(3, 5) + R(4, 1) + R(5, 4) \\ &= 0 + 15 + 0 + 11 + 19 + 0 + 0 \\ &= 45. \end{aligned}$$

4. FURTHER DISCUSSION OF THE ALGORITHM

In this section, we will show three propositions which guarantee that the algorithm yields the optimal solution. The proof of these propositions are given in the appendix. Also in this section, the proof of the formula [equation (13)] for calculating the cost of the postman tour will be given.

Proposition 1

All $R(i, j)$, $f(i, j)$, $(i, j) \in A$, and $p(i)$, $i \in V$ generated in the algorithm satisfy the (OC).

This proposition shows that the algorithm will yield the optimal solution to the directed CPP.

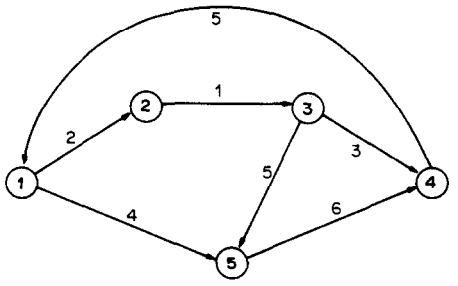
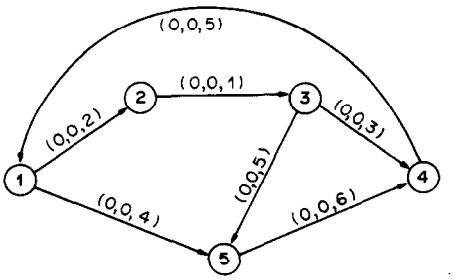
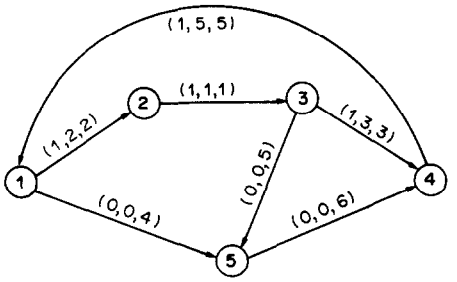


Fig. 1. Example directed network.



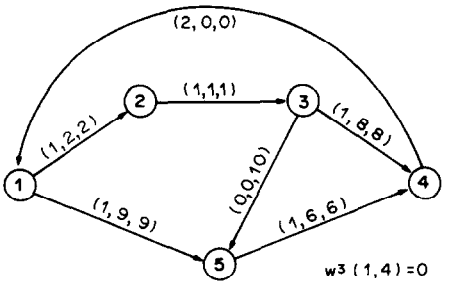
(t, s) = (1,2); SP(s,t) = 2-3-4-1.

Fig. 2. Iteration 1.



(t, s) = (1, 5); SP(s,t) = 5-4-1

Fig. 3. Iteration 2.



(t, s) = (3,5); SP(s,t) = 5-4-1-2-3

Fig. 4. Iteration 3.

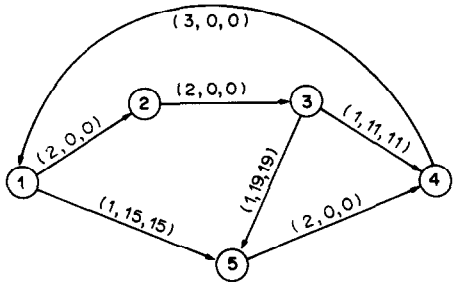
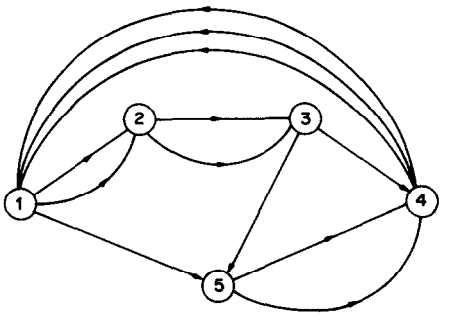


Fig. 5. Iteration 4.



Total-cost = 45

Fig. 6. Enlarged Euler network.

Definition 1

Define the arc sets I, IR, and P as follows:

$$I = \{(i,j): f(i,j) = 1 \text{ and } R(i,j) \geqslant 0,\ (i,j) \in A\};$$

$$IR = \{(i,j): f(i,j) \geqslant 2 \text{ and } R(i,j) = 0,\ (i,j) \in A\};$$

$$P = \{(i,j): f(i,j) = 0 \text{ and } R(i,j) = 0,\ (i,j) \in A\}.$$

As such, $A = I \cup IR \cup$ and $I \cap IR = I \cap P = IR \cap P = \varnothing$. The arc sets I and IR are feasible arc sets, P is the unfeasible arc set. The optimal solution can only consist of arcs either in I or in IR. The solution is not optimal as long as there is an arc belonging to P.

Proposition 2

At each iteration, at least one arc leaves P and enters I.

Definition 2

Define $(i,j)^k$ as the arc (i,j) at the kth iteration of the algorithm. $f^k(i,j)$, $R^k(i,j)$, $w^k(i,j)$, and $d^k(i)$ are defined as the variables at the kth iteration.

Proposition 3

If $(i, j)^k \in I \cup IR$, then $(i, j)^k \in I \cup IR$.

The propositions 2 and 3 guarantee that the algorithm will stop within the limited number of iterations.

The objective in (P) and the objective in (DP) are equal when the solution is optimal. Since the $R(i, j)$ obtained from the algorithm satisfy (OC) (proposition 1), the objectives of (P) and (DP) are equal to the sum of the $R(i, j)$ at the final iteration. Therefore, the total cost of the optimal postman tour is determined by equation (13).

5. COMPLEXITY ANALYSIS AND THE ALGORITHM EXTENSION

The computational complexity of the algorithm may be determined by studying the performance of the algorithm in the worst cases.

At each iteration of the algorithm, at least one arc in P (unfeasible arcs) is changed into either I or IR. In the worst case, in order to change all the arcs in P into either I or IR, the algorithm will need $m-n+1$ iterations, where m is the number of arcs and n the number of nodes on the network $N(V, A)$. For example, for the network with 4 nodes and 7 arcs in Fig. 7(a) and the network with 3 nodes and 6 arcs on Fig. 7(b), each will require at most 4 iterations.

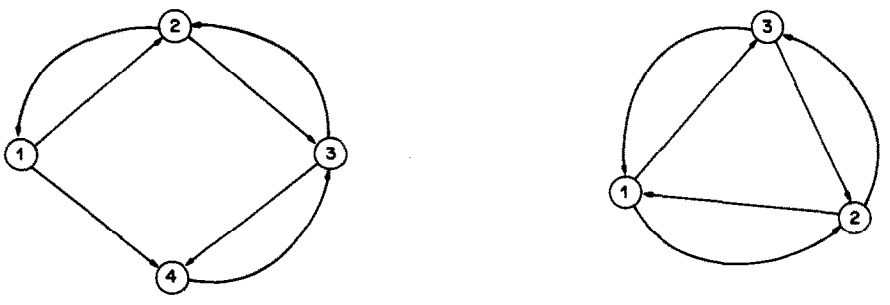
In step 3 of the algorithm, in fact, it is unnecessary to calculate shortest paths from s to all nodes. This is obvious by considering equation (12) in Step 4, from which we can derive the fact that those $d(i)$ with $d(i) > d(t)$ can be set to the value of $d(t)$ without affecting the accuracy of $R(i, j)$. In this case, the complexity of this shortest path algorithm is $O(3n^2/2)$.

Therefore, at each iteration, the algorithm will require $3n^2/2$ computations, in addition to $3m$ computations for calculating the arc weights $w(i, j)$, $(i, j) \in A$. However, it seems impossible that these two worst cases will occur simultaneously in the same problem. In case they do occur in the same problem, the algorithm will require $(m-n+1) * (3n^2/2+3m)$ computations, or the algorithm for the directed CPP has the complexity of $O(kn^2)$. The constant k is closely related to the structure of the network. For some problem instances, such as the sparse network, k could be much smaller than m and n. That is, the computational complexity $O(kn^2)$ is better than $O(n^3)$ that the algorithm in [4] has and $O(m * n^2)$ that the algorithm in [1] has.

The same algorithm presented for the directed CPP can be extended to the m-CPP which stands for m Chinese Postmen Problem. In the m-CPP, m subtours must be determined. Every arc in the network must be traversed by at least one of the m subtours and the total cost is minimized.

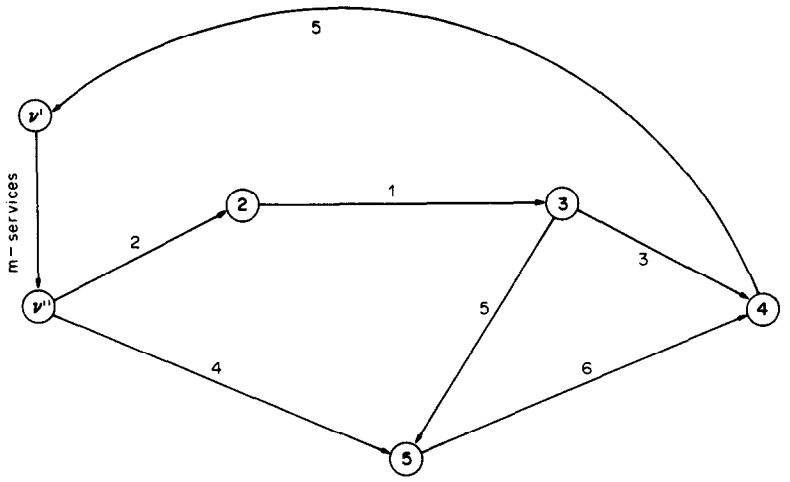
First, the network should be slightly altered. The node v^* representing the post office or the depot of the vehicle is split into two artificial nodes v' and v''. All arcs incident into v^* are set to be incident into v' and all arcs incident out of v^* are set incident out of v''. In addition, an artificial arc (v', v'') with no cost is added. This artificial arc requires being traversed m times in order to generate m subtours each of which is connected to v^*. For example, for the network in Fig. 1, $v^* = v_1$ and the modified network is shown in Fig. 8.

Second, the algorithm must be modified to handle the arc requiring m traverses. This is accomplished by generalizing the definition of unfeasible arcs (P) and the feasible arcs (I, IR). The



(a) Graph with 4 nodes 7 arcs (b) Graph with 3 nodes 6 arcs

Fig. 7. Example graphs for complexity analysis.



Except indicated, all arcs require one service

Fig. 8. Example modified network for m-CPP.

unfeasible arcs now represent the arcs that do not have the required traverses and the feasible arcs represent the arcs that have already had the required traverses. With these changes, the algorithm can be applied to the modified network shown above (Fig. 8).

6. SUMMARY

This paper studied the formulations of the directed CPP and developed a new algorithm. In the worst case, the algorithm requires $m-n+1$ iterations to achieve the optimal solution. The optimality proof and the computational complexity analysis of the algorithm are provided. With slight modifications, the algorithm can be applied to the directed m-CPP.

Acknowledgements—The authors are grateful to the anonymous referees for their valuable comments and suggestions. This work is supported by the foundation of Chinese Natural Sciences under Grant GWAN-1674.

REFERENCES

1. E. J. Beltrami and L. D. Bodin, Networks and vehicle routing for municipal waste collection. *Networks* **4**, 65–94 (1974).
2. E. Minieka, The Chinese Postman Problem for mixed networks. *Mgmt Sci.* **25**, 643–648 (1979).
3. E. Minieka, *Optimization Algorithms for Networks and Graphs*. Marcel Dekker, New York (1978).
4. J. Edmonds and E. L. Johnson, Matching, Euler tours and the Chinese Postman. *Math. Progr.* **5**, 88–124 (1973).
5. J. Edmonds and R. M. Karp, Theoretical improvement in algorithmic efficient for network flow problems. *J. Ass. Comput. Mach.* **19**, 248–264 (1972).
6. L. Bodin, B. Golden *et al.*, Routing and scheduling of vehicles and crews, the state of the art. *Comput. Opns Res.* **10**, 111–114 (1983).
7. Mei-Ko Kwan, Graphic programming using odd or even points. *Chinese Math.* **1**, 273–277 (1962).
8. Mei-Ko Kwan, A survey on the Chinese Postman Problem. *J. Math. Res. Exposit.* **4**, 113–119 (1984) (in Chinese).

APPENDIX

Proof of Proposition 1

It is sufficient to prove that $R^k(i,j)$, $(i,j)\in A$, satisfy the constraints of (DP).
By induction on k, we show that $R^k(i,j)\geqslant 0$ for all $(i,j)\in A$.
Obviously, for $k=0$, $R^0(i,j)=d(i,j)\geqslant 0$.
Suppose $R^{k-1}(i,j)\geqslant 0$. From Step 4 of the algorithm, we have

$$d^k(j)\leqslant d^k(i)+w^k(i,j) \tag{14}$$

$$w^k(i,j)\leqslant R^k(i,j). \tag{15}$$

Thus it follows that

$$d^k(i)+R^{k-1}(i,j)-d^k(j)\geqslant 0. \tag{16}$$

Applying equation (16) to equation (13), we obtain

$$R^k(i, j) \geq 0.$$

Therefore, it holds that $R^k(i, j) \geq 0$ for all $(i, j) \in A$.
Secondly, we show that $R^k(i, j)$ $[(i, j) \in A]$ satisfy the first constraint of (DP). Let

$$p(i) = p^k(i) = p^{k-1}(i) + \Delta p^k(i) \tag{17}$$

$$\Delta p^k(i) = \text{Min}\{d^k(i), d^k(t)\}. \tag{18}$$

From the algorithm, $p^0(i) = 0, d^0(i) = 0$ for all $i \in V$. As the result, $p^1(i) = 0$, for all $i \in V$. Therefore, $\Delta p^1(i) = 0$ for all $i \in V$.

For $k = 1$,

$$\begin{aligned} R^1(i, j) &= \Delta p^1(i) + R^0(i, j) - \Delta p^1(j) \\ &= p^1(i) + d(i, j) - p^1(j). \end{aligned}$$

Suppose $R^{k-1}(i, j) = p^{k-1}(i) + d(i, j) - p^{k-1}(j)$. From equation (13), we have:

$$\begin{aligned} R^k(i, j) &= \Delta p^k(i) + R^{k-1}(i, j) - \Delta p^k(j) \\ &= \Delta p^k(i) + [p^{k-1}(i) + d(i, j) - p^{k-1}(j)] - \Delta p^k(j) \\ &= p^k(i) + d(i, j) - p^k(j). \end{aligned}$$

Therefore, $R^k(i, j)$, $(i, j) \in A$ satisfy the constraints of (DP).

Proof of Proposition 2

In Step 3 at the kth iteration, the arc (t, s) in P is selected and $f(t, s)$ is increased from 0 to 1. According to the definition, arc (t, s) becomes a feasible arc: $(t, s) \in I$.

Proof of Proposition 3

The proposition is true when the following four situations hold:

(a) $(i, j)^{k-1} \in I$
  (a1) $(i, j)^k \in I$ $f(i, j)$ not changed;
  (a2) $(i, j)^k \in IR$ $f(i, j)$ increased.

(b) $(i, j)^{k-1} \in IR$
  (b1) $(i, j)^k \in I$ $f(i, j)$ decreased;
  (b2) $(i, j)^k \in IR$ $f(i, j)$ either increased or decreased.

The situations (a1) and (b1) are obvious. We will concentrate on the proof of (a2) and (b2).

(a2) Because $f^{k-1}(i, j) = 1$ and $f^k(i, j) = 2$, the orientation of arc (i, j) must be the same as the orientation of the cycle and $w^k(i, j) = R^{k-1}(i, j)$.

$$\begin{aligned} \Delta p^k(j) &= \Delta p^k(i) + w^k(i, j) \\ &= \Delta p^k(i) + R^{k-1}(i, j). \end{aligned} \tag{19}$$

It follows from equations (13) and (18) that $R^k(i, j) = 0$. Therefore, $(i, j)^k \in IR$.

(b2) Since $f^{k-1}(i, j) \geq 2$, according to the algorithm,

$$R^k(i, j) = w^k(i, j) = w^k(j, i) = 0,$$

$$\Delta p^k(i) = \Delta p^k(j).$$

Similarly, we have $R^k(i, j) = 0$ and therefore, $(i, j) \in IR$.