An optimal method for the mixed postman problem

Chapter · January 2006

DOI: 10.1007/BFb0008937

CITATIONS
22

READS
478

5 authors, including:

Nicos Christofides
Imperial College London
104 PUBLICATIONS **10,373** CITATIONS
SEE PROFILE

Enrique Benavent
University of Valencia
33 PUBLICATIONS **1,932** CITATIONS
SEE PROFILE

Vicente Campos
University of Valencia
39 PUBLICATIONS **1,589** CITATIONS
SEE PROFILE

Angel Corberán
University of Valencia
105 PUBLICATIONS **2,564** CITATIONS
SEE PROFILE

Lecture Notes in Control and Information Sciences

Edited by A.V. Balakrishnan and M. Thoma

59

System Modelling and Optimization

Proceedings of the 11th IFIP Conference
Copenhagen, Denmark, July 25–29, 1983

Edited by P. Thoft-Christensen

Springer-Verlag
Berlin Heidelberg New York Tokyo

AN OPTIMAL METHOD FOR THE MIXED POSTMAN PROBLEM

N.Christofides Imperial College of London.UK

E.Benavent, V.Campos, A.Corberán and E.Mota Universidad de Valencia.Spain

1. INTRODUCCION

The problem of traversing every arc and edge of a mixed graph (with nonnegative associated costs) by a tour of minimum total cost, is known as the Mixed Chinese Postman Problem (MCPP). It is a more general case than the nondirected (CPP) and totally directed (DCPP) versions of this problem, and which were already solved efficiently by Edmonds and Johnson [1] .

The particular case of the MCPP in which every vertex of the graph is incident with an even number of lines (arcs or edges), is the only one that can be solved in polynomial time ([1] , [5]). The general case of the MCPP is an NP-Complete problem, as shown by Papadimitriou [7] .

In this paper we present an exact algorithm for the MCPP based on a branch and bound algorithm using Lagrangean bounds and also report some computational results.

In Section 2 we present an integer formulation of the MCPP which is equivalent to the one presented in [4] , and in which some redundant constraints have been added. In Section 3 we describe the heuristic algorithm used for finding an initial feasible solution, which can be considered equivalent to the one proposed by Frederickson [2] (known as MIXED 2).

2. A FORMULATION OF THE MCPP

Let G=(N,A,E) be a strongly connected mixed graph, where N is a set of vertices, A is a set of arcs and E is a set of edges.

We represent each arc $a \in A$, directed from vertex i to vertex j, by the ordered pair (i,j), and each edge $e \in E$, having terminal vertices i and j, by the nonordered pair <i,j>. In addition, we suppose that every line of graph G has a nonnegative associated cost c_{ij}.

Let $\Gamma_A(i)$ be the set of all final vertices of the arcs leaving vertex i, let $\Gamma_A^{-1}(i)$ be the set of all initial vertices of the arcs entering vertex i and let $\Gamma_E(i)$ be the set of all vertices linked with vertex i by an edge. The MCPP can be formulated as an integer problem as follows:

$$(P)\ \text{Min}\{ \sum_A c_{ij}x_{ij} + \sum_E c_{ij}(y_{ij}+y_{ji}) + \sum_A c_{ij} \}$$

s.t.:

$$\sum_{j\in\Gamma_A(i)} (1+x_{ij}) + \sum_{j\in\Gamma_E(i)} y_{ij} = \sum_{j\in\Gamma_A^{-1}(i)} (1+x_{ji}) + \sum_{j\in\Gamma_E(i)} y_{ji} \qquad \forall i\in N \qquad (1)$$

$$y_{ij} + y_{ji} \geq 1 \qquad \forall e=\langle i,j\rangle \in E \qquad (2)$$

$$x_{ij},\ y_{ij},\ y_{ji} \geq 0 \text{ and integer} \qquad \forall (i,j)\in A,\ \forall e=\langle i,j\rangle \in E \qquad (3)$$

$$\sum_{j\in\Gamma_A(i)} x_{ij} + \sum_{j\in\Gamma_A^{-1}(i)} x_{ji} + \sum_{j\in\Gamma_E(i)} (y_{ij}+y_{ji}-1) - 2w_i = b_i \qquad \forall i\in N \qquad (4)$$

$$w_i \geq 0 \text{ and integer} \qquad \forall i\in N \qquad (5)$$

where the arc variables x_{ij} represent the number of extra times that arc (i,j) must be traversed by the solution circuit; edge variables y_{ij}, y_{ji} represent the number of times that edge<i,j> must be traversed (from i to j and j to i, respectively) by the solution circuit, and: $b_i = 0$ if vertex i is incident with an even number of lines in G, and $b_i = 1$ if vertex i is incident with an odd number of lines in G.
Constraints (1) state that, relative to the solution tour, every vertex is symmetric (i.e. the indegree is equal to the outdegree). Constraints (2) force every edge of graph G to be traversed by the tour at least once. Constraints (4) and (5) are redundant and express the requirement that every vertex must (relative to the tour) have an even number of lines incident with it. The importance of these redundant constraints will become clear later (see Section 4).
Every feasible solution to problem P produces a feasible solution to the MCPP on G and viceversa. It can also be shown ([4] , [6]) that, if in an optimal solution of the MCPP the two variables associated with a given edge <i,j> are both non-zero, then both of them must be equal to one.

2.1 Graph transformations

(a) Replacing edges with arcs

Under certain conditions it is possible to transform the original graph G by converting some edges into arcs thus reducing the number of constraints of type (2). The conditions that must be hold for such a transformation to be possible are given by the following Propositions.

Definition 1. An edge <i,j> is a "cut-edge in the direction i to j " in G iff replacing it by the arc (j,i), the resulting graph is not strongly connected.

Definition 2. An edge <i,j> is a "cut-edge" in G iff it is a cut-edge in its two possible directions.

Proposition 1. If an edge <i,j> of graph G is a cut-edge only in one direction, then it will not be traversed in the opposite direction by any optimal solution of the MCPP.

Proposition 2. If an edge <i,j> of graph G is a cut-edge, then it must be traversed exactly once in each direction by every optimal solution of the MCPP.

The purpose of the apriori transformation of edges into arcs -implied by Propositions 1 and 2- is to try to convert the mixed CPP into the purely directed CPP which can then be easily solved.

(b) Repeating existing arcs

Definition 3. An arc $(i,j)\in A$ is a "cut-arc" iff by deleting (i,j) the resulting graph is not strongly connected.

The above definition implies that for a cut-arc (i,j) there exists in the graph a cut $(S,\bar{S})$, $i\in S$, $j\in\bar{S}$ such that: (i) the arc (i,j) is the only one directed from S to $\bar{S}$, and (ii) there is no edge between S and $\bar{S}$. This, in turn, implies that if the number of arcs from $\bar{S}$ to S is k, the cut-arc (i,j) must be traversed at least k times in any feasible solution. Hence, the lower bound on the variables x_{ij} in constraints (3) can be increased from 0 to k-1.

Note that transformations of type (a) above involve the substitution of some y-variables with x-variables in the formulation of problem P. Henceforth, when we refer to the graph and the problem formulation it will be relative to the graph that is produced after the above apriori transformations are applied.

3. A HEURISTIC ALGORITHM

As already pointed out in Section 1, the MCPP can be efficiently solved in the case of an even graph. Minieka [5] presents an algorithm for even graphs based on the resolution of a minimum cost flow problem on a transformed graph. From the solution to the flow problem, a direction is assigned to every edge of the original graph, transforming it into a symmetric graph. The algorithm can be extended to the general case of the MCPP, in order to obtain feasible solutions, and is as follows:

Step 1: Assign a direction to every edge of G, thus transforming edges into arcs and obtaining the directed graph $G_d=(N,A_d)$. Compute $\xi(i)=\left|\Gamma^{-1}_{A_d}(i)\right| - \left|\Gamma_{A_d}(i)\right|$ $\forall i\in N$.

Step 2: Construction of the directed graph $G_F=(N,A\cup A_F)$.

Corresponding to every edge $\langle i,j\rangle \in E$, put two arcs in parallel one in each direction in A_F, with costs equal to the cost of the edge. In addition, put an "artificial" arc of zero cost in the opposite direction to the one previously assigned to the edge in G_d. Arcs of A appear with their original costs.

Step 3: Solve in G_F the minimum cost flow problem, in which:

(a) every vertex $i\in N$ with $\xi(i)>0$ is a source with supply $\xi(i)$.

(b) every vertex $i\in N$ with $\xi(i)<0$ is a sink with demand $-\xi(i)$.

(c) the capacity of each artificial arc is 2 and that of the remaining arcs ∞ .

Step 4: Construction of the symmetric mixed graph $G_S=(N,A_S,E_S)$.

Let $f(i,j)$ and $f^a(i,j)$ represent the number of flow units circulating in the optimal solution through non-artificial and artificial arcs, respectively. From the resolution of the flow problem defined in Step 3, construct graph G_S as follows:

(a) Referring to the flow circulating through the arcs representing edges:

(1) if $f^a(i,j)=0$, put $1+f(j,i)$ arcs (j,i) in A_S.

(2) if $f^a(i,j)=2$, put $1+f(i,j)$ arcs (i,j) in A_S.

(3) if $f^a(i,j)=1$, put edge $\langle i,j\rangle$ in E_S.

(b) Referring to the flow circulating through the remaining arcs of G_F, put 1+f(i,j) arcs (i,j) in A_S.

Step 5: Obtaining a feasible solution.

If $E_S=\phi$, G_S represents the optimal solution to the MCPP on G.

If $E_S\neq\phi$, solve a minimum cost perfect matching on the odd vertices of the graph induced by E_S on G_S, adding to E_S a copy of every edge duplicated by the matching.

The output is an even and symmetric graph and, therefore, a feasible solution to the MCPP on G.

4. AN EXACT ALGORITHM FOR THE MCPP

We present in this Section a branch and bound algorithm for solving the MCPP which is based in the computation of lower bounds from two possible Lagrangean relaxations of problem P.

4.1 Lower bounds

Two lower bounds can be obtained to the value of the optimal solution of the MCPP. The first one is obtained by relaxing, in a Lagrangean way, the symmetry constraints (1), using multipliers u_i and solving the minimum cost perfect matching. The second one is obtained via the Lagrangean relaxation of constraints (2), using multipliers λ_{ij} (eliminating (4) and (5)) and resolving a minimum cost flow problem, as described below.

4.1.1 Lagrangean relaxation of symmetry constraints

Consider the original problem P and problem $PR_1(x,y,u)$ obtained by relaxing, in a Lagrangean way, constraints (1) using multipliers u_i. The formulation of problem $PR_1(x,y,u)$ is then as follows:

$$PR_1(x,y,u)\quad \text{Min } \{ \sum_{A} (c_{ij}+u_j-u_i)x_{ij} + \sum_{E_d} (c_{ij}+u_j-u_i)y_{ij} + \sum_{i\in N} D(i)u_i + \sum_{A} c_{ij}\}$$

s.t.: (2),(3),(4),(5) and u_i non-restricted.

where $E_d=\{ (i,j), (j,i) \mid \forall <i,j>\in E \}$ and $D(i)=|\Gamma_A^{-1}(i)| - |\Gamma_A(i)|$.

A. Solving problem $PR_1(x,y,u)$

Obviously, problem $PR_1(x,y,u)$ is bounded only when the multipliers u produce nonnegative modified costs $c'_{ij}=c_{ij}+u_j-u_i$.

Since the edge-variables y_{ij}, y_{ji}, associated to every edge <i,j> of G always appear as $y_{ij}+y_{ji}$, in the constraints of problem $PR_1(x,y,u)$, at least one of these two variables must be nonzero, and that variable with the lowest corresponding cost will certainly take a nonzero value. Therefore, defining variables $h_{ij}=y_{ij}+y_{ji}-1$, for every edge e= <i,j>$\in$E, with associated cost $\hat{c}_e=\min \{ c'_{ij}, c'_{ji}\}$, we obtain the following minimization problem, which is equivalent to $PR_1(x,y,u)$:

$$\text{Min}\{\sum_{A} c'_{ij}x_{ij} + \sum_{E} \hat{c}_e(h_{ij}+1) + \sum_{i\in N} D(i)u_i + \sum_{A} c_{ij}\}$$

s.t.:

$$\sum_{j\in\Gamma_A(i)} x_{ij} + \sum_{j\in\Gamma_A^{-1}(i)} x_{ji} + \sum_{j\in\Gamma_E(i)} h_{ij} - 2w_i = b_i \quad \forall i\in N$$

x_{ij}, h_{ij}, $w_i \geq 0$, and integer $\forall i,j$.

which is easily recognized as a minimum cost perfect matching on the odd degree vertices of graph G, ignoring the direction of the arcs. The modified costs are c'_{ij}, $\hat{c}_e$ and there is a constant term in the objective function (for a given u), i.e. :

$$K(u) = \sum_{E} \hat{c}_e + \sum_{i\in N} D(i)u_i + \sum_{A} c_{ij}.$$

The optimal solution to problem $PR_1(x,y,u)$, given u, will be obtained by assigning to the arc variables x_{ij} the values 1 or 0, depending on whether arc (i,j) has been used, or not, by the perfect matching. The values of y_{ij} are now given by: $y_{ij}=2$ if $c'_{ij}=\hat{c}_e$ and $h_{ij}=1$; $y_{ij}=1$ if $c'_{ij}=\hat{c}_e$ and $h_{ij}=0$; $y_{ij}=0$ otherwise.
The cost of the solution can be expressed as: $v(PR_1(x,y,u))=c(M^*(u)) + K(u)$, where $M^*(u)$ represents the set of arcs and edges in the optimal solution to the minimum cost perfect matching and $c(M^*(u))$ is the cost of the arcs in this set.

B. Selection of multipliers u

Several difficulties appear when applying subgradient optimization,in order to obtain a good approximation to the problem of $\max_u v(PR_1(x,y,u))$ and hence a good bound. The main difficulty is to compute at each iteration all the shortest distances among the odd degree vertices in graph G, before solving a minimum cost perfect matching problem. In addition, not all the multipliers produced by the subgradients can be used unchanged, since the modified costs c'_{ij} should always remain non-negative.
An alternative easy way of selecting multipliers u is to consider only one term of the objective function and to solve the problem

$$\max_u \sum_{i\in N} D(i)u_i$$

$$\text{s.t.: } u_i - u_j \leq c_{ij} \quad \forall (i,j)\in A\cup E_d.$$

The constraints being the non-negativity restrictions on the arc/edge modified costs. The optimal solution to the above problem can easily obtained from the values of the variables of its dual problem, which corresponds to the minimum cost flow problem defined by the objective function of P and constraints (1).
However, this selection of multipliers u has also not performed very well and has been computationally surpassed by a simple greedy procedure which tries to optimize approximately problem Q: $\max_u K(u)$

$$\text{s.t.: } u_i - u_j \leq c_{ij} \quad \forall (i,j)\in A\cup E_d$$

This was the method finally adopted and the computational results reported later were derived using this procedure.

4.1.2 Lagrangean relaxation of constraints (2)

In problem P, ignoring the redundant constraints (4) and (5), we can relax, in a Lagrangean fashion, constraints (2), associating to each one a non-negative scalar λ_{ij}, to obtain the minimization problem:

$PR_2(x,y,\lambda)$ Min $\sum_A c_{ij}x_{ij} + \sum_E (c_{ij}-\lambda_{ij})(y_{ij}+y_{ji}) + \sum_E \lambda_{ij} + \sum_A c_{ij}$

s.t.: (1) and (3).

A. Solving problem $PR_2(x,y,\lambda)$

Given a vector of multipliers $\lambda \geq 0$, constraints (1) and the non-negativity of variables x and y, define a minimum cost flow problem. Therefore, the optimal solution of $PR_2(x,y,\lambda)$ consists of the following assignment of values: $x_{ij}=f(i,j)$ $\forall (i,j)\in A$ and $y_{ij}=f(i,j)$ $\forall (i,j)\in E_d$, where f(i,j) represents the number of flow units transported through the arc (i,j), or through the edge <i,j> from i to j, in the optimal solution. The above problem is solved without capacity requirements on the arcs and edges of graph G, and from its solution we can construct a symmetric graph by repeating arcs (i,j), f(i,j) times. Note, however, that not every edge of G was necessarily traversed by the optimal flow f(i,j).

On the other hand, since demands and supplies are integer-valued, the problem $PR_2(x,y,\lambda)$ has the integrality property and, therefore (see [3]), the lower bound that can be obtained from this relaxation, though easily computed, will be, at most, equal to the bound obtained from the linear relaxation of P, ignoring constraints (4) and (5).

B. Selection of multipliers λ

The trivial selection $\lambda=0$, may produce a feasible solution to problem P, in which case the subproblem is solved optimally and the node is fathomed. This occurs when every edge of graph G has been traversed by a nonzero number of flow units in the solution (x°,y°) to the problem $PR_2(x,y,0)$. Otherwise, a heuristic procedure is applied to obtain new multipliers $\bar{\lambda}_{ij}$ as follows:

Let $E_0=\{ <i,j> \in E \mid y^\circ_{ij}+ y^\circ_{ji}=0 \}$: Non traversed edges.

$E_1=\{ <i,j> \in E \mid y^\circ_{ij}+ y^\circ_{ji}=1 \}$: Edges traversed once.

$E_2=\{ <i,j> \in E \mid y^\circ_{ij}+ y^\circ_{ji}\geq 2 \}$: Edges traversed more than once.

We now penalize edges <i,j> according to the following choice of $\bar{\lambda}_{ij}$:

$\bar{\lambda}_{ij}=c_{ij}$ $\forall <i,j>\in E_0$; $\bar{\lambda}_{ij}=c_{ij}/2$ $\forall <i,j>\in E_1$; $\bar{\lambda}_{ij}=0$ $\forall <i,j>\in E_2$.

These multipliers $\bar{\lambda}_{ij}$ have been used as the initial ones in the subgradient method, producing good computational results.

4.2 An upper bound

The upper bound used in the branch and bound method is obtained using the heuristic

algorithm described in Section 3. The values of the upper bound shown in the computational results, obtained on a collection of test problems, are given in Section 5. In 11 of the total of 34 tested problems, this bound was optimal. On average, this bound is within 3.11% of the optimal value.

4.3 The tree search

The branching is made on the edge-variables. In this way, each node is separated in two successors, one of them obtained by adding constraint $y_{ij} \geq 1$, whereas the other, by adding the alternative constraint $y_{ij}=0$.
Each constraint $y_{ij} \geq 1$ allows us to bound (at this node and its successors) the "paired" variable y_{ji} by 1, thus improving the lower bounds produced by $PR_2(x,y,\lambda)$.
On the other hand, constraint $y_{ij}=0$ transforms the associated edge into the arc (j,i), allowing the appearance of new cut-arcs and cut-edges, at this node and its successors.
At the level in which all edges of graph G have an assigned direction, the MCPP becomes the DCPP which is optimally solved by calculating the corresponding minimum cost flow problem.
We have used the depth-first branching strategy, due to its smaller memory requirements. The order in which the branching variables are chosen is made on the expectation of a similarity between the direction assigned to an edge by the upper bound and the one assigned to it by the optimal solution to the problem. The Table in Section 5 shows that, in most of the tested problems the optimal solution is found very quickly, with respect to the total number of nodes, justifying the adopted rule.

5. COMPUTATIONAL RESULTS

The algorithm described in the previous Section has been tested on a collection of 34 problems, randomly generated and costs chosen between 1 and 20. The algorithm was coded in FORTRAN and run on the UNIVAC 1100/60 computer. The times shown in the table are CPU seconds of the above computer and correspond to total execution time.
Problems up to 50 vertices, 66 arcs and 39 edges could be completely solved with a maximum time of 500 seconds.
The maximum number of subgradient iterations allowed to obtain the Lagrangean bounds was 20 at the root node, and only 10 at the internal ones. Finally, we present in the table the computational results, in which the three numbers in the first column represent, respectively, the total number of : vertices/arcs/edges in the original graph.

6. REFERENCES

[1] Edmonds, J. and Johnson, E. "Matching, Euler tours and the Chinese Postman", Math. Prog. 5, 1973, p. 88-124.
[2] Frederickson, G. N. "Approximation Algorithms for some Postman Problems", Journal of A.C.M., vol.26, 1979, p. 538-554.
[3] Geoffrion, A. "Lagrangean Relaxation for Integer Programming", Math. Prog. Study 2, 1974, p. 82-114.

[4] Kappauf, C. H. and Koehler, G. J. "The Mixed Postman Problem", Discrete App. Math. 1, 1979, p. 89-103.
[5] Minieka, E. "Optimization Algorithms for Networks and Graphs". Ed. Marcel Dekker, New York, 1978.
[6] Minieka, E. "The Chinese Postman Problem for Mixed Networks", Manag. Science, vol.25, 1979, p. 643-648.
[7] Papadimitriou, C. "On the Complexity of Edge Traversing", Journal of A.C.M., vol. 23, 1976, p. 544-554.

Problem	Upper bound	Opt. value	L.B. matching	L.B. flow	Total number nodes	Number of opt. node	Fathomed nodes by L.B.'s	CPU seconds
8/ 7/ 4	83	83	-	83	1	0	1	0.03
7/ 3/ 6	36	36	-	36	1	0	1	0.09
9/ 7/ 5	54	54	53	51	7	0	0	0.18
13/14/ 7	31	31	30	30	7	0	2	0.32
10/ 8/ 6	55	55	55	50	1	0	1	0.07
13/13/ 8	140	140	140	129	1	0	1	0.27
10/ 7/10	122	122	109	100	55	0	16	0.84
8/ 5/10	95	89	89	75	17	17	2	0.68
10/ 9/10	123	119	107	107	33	17	13	0.65
19/15/15	191	191	158	174	43	0	21	1.16
13/12/13	122	122	122	112	1	0	1	0.04
16/16/17	205	190	190	177	28	28	6	1.02
16/16/17	184	184	184	170	1	0	1	0.07
18/16/16	194	181	174	160	81	54	39	3.35
15/ 9/20	235	211	211	191	16	16	0	0.53
25/25/20	394	386	364	353	141	22	59	10.54
33/48/22	811	811	634	782	319	0	156	30.53
17/19/20	257	253	241	237	229	37	114	8.58
20/15/24	317	303	287	290	175	39	80	8.62
27/43/20	657	630	569	590	593	31	273	47.65
43/74/20	887	881	790	847	315	50	152	51.54
15/10/21	275	257	257	232	23	23	0	0.62
30/46/23	610	601	559	575	363	25	178	33.11
32/55/26	972	964	890	908	1631	23	808	194.06
21/12/26	338	308	308	260	102	102	31	5.22
22/19/25	1393	1190	1178	1073	507	118	247	30.82
28/47/25	793	791	677	748	2427	26	1192	232.31
41/52/29	1062	1040	898	1016	501	25	280	74.60
38/61/27	826	814	728	791	463	48	228	72.06
30/47/28	726	704	688	664	483	91	234	55.07
48/78/32	1170	1162	1056	1149	361	57	173	79.98
25/23/29	1438	1331	1331	1207	286	286	133	18.10
50/85/36	1598	1580	1449	1540	1917	42	949	507.37
50/66/39	1381	1339	1206	1309	1943	38	968	500.57

Table : Computational results