A 3/2-approximation algorithm for the Windy Postman Problem*

Balaji Raghavachari† Jeyakesavan Veerasamy‡

February 9, 2001

Abstract

The windy postman problem is a generalization of the Chinese postman problem in which the edge-traversal costs are asymmetric. The problem is to compute a shortest tour that traverses every edge of a given graph at least once, and it is known to be NP-hard. An approximation algorithm with a performance ratio of $\frac{3}{2}$ for the windy postman problem is presented.

Keywords: Windy postman problem, Asymmetric postman problem, Chinese postman problem, Combinatorial optimization, Approximation algorithms, Linear programming.

1 Introduction

Given a connected, weighted, undirected graph, the Chinese postman problem (CPP) is to find a shortest closed walk of the graph in which each edge is traversed *at least* once. It is the optimization version of the Euler tour problem, which asks whether there is a closed walk such that each edge is traversed *exactly* once. Problems such as mail delivery, newspaper delivery, trash pick-up, and snow removal can be modeled by CPP. CPP can be solved in polynomial time if all edges of the graph are undirected, or if all edges are directed. However, if the input graph is mixed, i.e., if it contains both undirected and directed edges, CPP was shown to be NP-hard by Papadimitriou [12]. The Mixed Postman Problem (MPP) is a generalization of CPP, and it enables us to model one-way and two-way streets.

*A preliminary version of the paper appeared in SODA 1999 [15]

†Department of Computer Science, The University of Texas at Dallas, Richardson, TX 75083-0688. Research supported by the National Science Foundation under grants CCR-9409625 and CCR-9820902. Email: rbk@utdallas.edu.

‡Department of Computer Science, The University of Texas at Dallas, and Samsung Telecommunications America, Inc. Email: veerasam@utdallas.edu.

Following variation of CPP occurs in several practical scenarios: due to wind (or slope, or some other reason), the cost of traversing an edge in one direction may be cheaper than traversing the edge in the opposite direction. Such a problem is called the **Windy Postman Problem** (WPP), and it is a generalization of CPP and MPP. Note that the underlying graph is an undirected graph, so the edges can be traversed in any direction, but with different costs. This problem is also NP-hard [2, 9]. Undirected CPP, directed CPP, and MPP are all special cases of WPP.

Previous work: Minieka [10] introduced the Windy postman problem (WPP). Guan [9] presented an approximation algorithm by repeating the following steps: randomly select a cycle, compute the cost of both orientations of the cycle, and include the cheaper orientation in the solution. It can be shown that the performance ratio of Guan's algorithm is unbounded. He also proved that if both orientations of every cycle of the graph results in same cost, then the problem can be solved in polynomial time.

Win [16] proved that the WPP is solvable in polynomial time if the input graph is Eulerian, i.e., if all nodes have even degree. Win also presented a Linear Program (LP) to orient the edges and form directed tours. He proved that the LP gives an integer solution for Eulerian graphs, and a half-integral solution for general graphs. For general graphs, Win suggested to add a minimum-cost T-join of odd-degree nodes based on the average costs of the edges to make the graph Eulerian and then obtain an integral solution to the LP. Since a minimum-cost T-join duplicates each edge at most once, Win's algorithm has an approximation ratio of 2. Several other papers have appeared in the literature on the WPP, its LP formulation, and related solution techniques [8, 13]. The reader is referred to one of several comprehensive surveys [1, 2, 5, 6, 17] for more references to postman problems.

Edmonds and Johnson [4], Frederickson [7], Christofides *et al* [3], and Ragavachari and Veerasamy [14] have presented approximation algorithms for the Mixed Postman Problem (MPP). Frederickson showed that the algorithm of [4] finds a tour which costs at most twice the cost of an optimal tour (i.e., approximation ratio of 2). He also presented a mixed strategy algorithm, which generates two solutions using two different heuristics, and then selects the least cost one. He proved that the approximation ratio of the mixed strategy algorithm is $\frac{5}{3}$. Raghavachari and Veerasamy [14] modified the mixed approach and obtained an approximation algorithm for MPP whose approximation ratio is $\frac{3}{2}$.

Our results: Even though several articles have appeared in the literature on WPP after Win's article in 1989, his result has been the best approximation algorithm for WPP in terms of the proven worst case ratio till now. We present an approximation algorithm for WPP that builds on our earlier work on MPP [14]. The extension of the algorithm from MPP to WPP is made difficult by the asymmetry in the cost of traversing an edge in its two orientations. We show how to overcome this obstacle and we obtain an approximation ratio of $\frac{3}{2}$. Our algorithm works by modifying Win's LP formulation of the WPP and combines it with Frederickson's mixed strategy.

Significance of our result: In many instances, problems on graphs with asymmetric costs seem much harder than the same problem when the cost function is symmetric. Examples include the Traveling salesman problem and the Steiner tree problem among many others. In metric spaces, the asymmetric versions of these problems seem to differ dramatically from their symmetric counterparts in terms of their approximability. The main obstacle seems to arise from the fact that, given a cycle, the cost of traversing the edges of the cycle in one direction could be arbitrarily large compared to the cost of traversing the cycle in the opposite direction. Prior to our work, the best known approximation ratio for the symmetric and asymmetric versions of the postman problem were $\frac{3}{2}$ and 2 respectively. We close this gap by developing an approximation algorithm for the windy postman problem with an approximation ratio of $\frac{3}{2}$.

2 Preliminaries

Problem statement: Given an undirected graph $G = (V, E)$, where V is a set of vertices, and E is a multi-set of edges, and an asymmetric, positive cost function $c = (c_{ij}, c_{ji})$ for each edge $(i, j) \in E$, the problem is to find a minimum-cost tour traversing each edge at least once. Note that c_{ij} can be arbitrarily different from c_{ji} for each edge (i, j). We assume that the graph is connected, i.e., there exists a path from any vertex u to any vertex v, since the problem is clearly infeasible for graphs that are not connected. The output is a tour which may traverse each edge several times. Therefore, except for traversing each edge once, the traversal could always use a shortest path between any two nodes.

Definitions: Intermediate solutions of WPP may have undirected edges, duplicated undirected edges, oriented edges, and duplicated oriented edges. When there is no confusion, we will refer to undirected edges as edges, and oriented edges as arcs. A *cut* is a partition of the vertex set into S and $V - S$. It is called *non-trivial* when neither side is empty. An edge *crosses* the cut if its one end-point is in S and the other end-point is in $V - S$. Let $outdegree(v)$ be the number of outgoing arcs from v. Similarly, $indegree(v)$ is the number of incoming arcs into v. Let $degree(v)$ be the total number of edges and arcs incident to v. We say v has even degree if $degree(v)$ is even. A graph has even degree if all its vertices have even degree. An undirected graph is Eulerian if and only if it has even degree. Let $surplus(v) = outdegree(v) - indegree(v)$. If $surplus(v)$ is negative, we may call it as a *deficit*.

Cost of a subgraph: the *average* cost of an edge (i, j) is defined to be the average of its two directional costs, $(c_{ij} + c_{ji})/2$. The *orientation* cost of an arc (i, j) is its directional cost minus the average cost of the corresponding edge, $(c_{ij} - c_{ji})/2$. We define the cost of an arbitrary multiset of edges and arcs, S, to be $C(S)$, the sum of the directional costs of its arcs, and the average cost of its edges. This provides a measure to evaluate the solutions in various stages of the algorithm. Based on this definition, the cost of the input graph is the sum of the average cost of its edges, since it has no arcs. The cost of the final solution graph is the sum of the directional costs of its arcs, since it has no edges. Observe that unlike CPP and MPP, the cost of the input graph is *not* a lower bound on the cost of an optimal solution to the WPP.

T-joins: given an undirected graph G and a subset of vertices T, a *T-join* is a subset of edges in which the degree of each vertex in T is odd and the degree of all other vertices is even. For any set of vertices T, there always exists a T-join as long as G is connected and the cardinality of T is even. Finding a minimum cost T-join in a graph is equivalent to finding a minimum cost matching in its closure graph (a graph in which the cost of each edge (u, v) is the length of a shortest path in G between u and v). Refer to the book by Nemhauser and Wolsey [11] for more details on T-joins.

3 Linear Program (LP) formulation of WPP

Let $G = (V, E)$ be the input graph, and for each edge $(i, j) \in E$, let x_{ij} and x_{ji} be the usage of the edge from i to j and from j to i respectively. Then the LP formulation of the WPP is

$$
\begin{aligned}
\min Z(G) &= \sum_{(i,j)\in E} (c_{ij}x_{ij} + c_{ji}x_{ji}) \\
&\text{subject to} \\
x_{ij} + x_{ji} &\geq 1, \ \forall (i,j) \in E && (1) \\
\sum_{j:(i,j)\in E} (x_{ij} - x_{ji}) &= 0, \ \forall i \in V && (2) \\
x_{ij}, x_{ji} &\geq 0, \ \forall (i,j) \in E && (3)
\end{aligned}
$$

In other words, this LP formulation minimizes the total traversal cost, while satisfying the following conditions: the total usage of each edge should be at least 1 (equation (1)), the number of incoming edges should be equal to the number of outgoing edges at each node (equation (2)), and the usage of an edge in both directions should be non-negative (equation (3)). Intuitively, this LP creates circular flows in the graph by forming several directed cycles. Addition of each such cycle can be associated with corresponding reduction in cost of the solution. If additional constraints are added stating that all x_{ij}-values are integers, then the resulting integer program describes WPP completely.

Lemma 1 (Win [16]) *For an Eulerian input graph G, all (x_{ij}, x_{ji}) components of the vertices of the polyhedron P corresponding to the WPP LP are integral. For arbitrary graphs, the components of the vertices of the polyhedron P are half-integral.*

Win [16] has shown that the solution of a suitably designed flow problem yields such a solution to the LP. Without loss of generality, we assume that the optimal solution to the LP found by an algorithm is one of the vertices of LP polyhedron structure formed by the constraints. If the input is an Eulerian graph, there are no undirected edges in the output graph. For general graphs, there can be undirected edges (corresponding to $x_{ij} = x_{ji} = \frac{1}{2}$) present in the output, in addition to directed edges. Note that an optimal solution to the LP will duplicate and orient edges any number of times as long as it can form directed cycles which decrease the cost of the solution.

Let G be an arbitrary input graph to the LP, and let x^* be an optimal solution to the LP, whose value is $Z^*(G)$. We construct a mixed multigraph $G_{LP} = (V, U, D)$ from x^* (with undirected edges U and arcs D) as follows: if $x_{ij} \geq 1$, then we place x_{ij} copies of the arc (i, j) in D; if $x_{ij} = \frac{1}{2}$, then we add the undirected edge (i, j) to U. G_{LP}, a mixed graph, is an interpretation of the optimal solution, and it contains G as an underlying undirected graph. It can be verified that the cost of G_{LP} is $Z^*(G)$. Also, observe that $Z^*(G) \leq C(G)$, because $x_{ij} = x_{ji} = \frac{1}{2}$ for all $(i, j) \in E$ is a feasible solution for the problem whose cost is $C(G)$.

4 Solving the LP

In this section, we will present an algorithm for solving the LP using a minimum-cost flow algorithm. This algorithm is similar to the one presented by Win to solve the LP on Eulerian windy graphs. We have modified it to solve the LP for general windy graphs. This is also similar to the algorithm presented by Brucker [2] to find an optimal INOUT augmentation for mixed graphs. The INOUT problem is that of finding a minimum-weight graph that contains a given mixed graph as a subgraph, such the indegree of each node equals its outdegree.

---

1. Input: A connected graph $G = (V, E)$ with a cost function $c = (c_{ij}, c_{ji})$ for edges of E.
2. For each edge $(i, j) \in E$, if $c_{ij} \leq c_{ji}$, then orient the edge from i to j, else orient the edge from j to i. We obtain a digraph $D_G = (V, A)$.
3. Construct a digraph $D' = (V, A')$ with arc set A', arc costs, arc capacities and node demands defined as follows:
   (a) For each arc $(i, j) \in A$, define three arcs in A', namely, one copy of (i, j) with cost c_{ij} and infinite capacity, one copy of (j, i) with cost c_{ji} and infinite capacity, and one additional copy of (j, i), denoted by $(j, i)'$, with cost $(c_{ji} - c_{ij})/2$ and capacity 2. We will refer to $(j, i)'$ as an "artificial" arc.
   (b) For each node $u \in V$, the demand is its surplus in D_G, i.e., $outdegree(u) - indegree(u)$. Note that a node with negative demand is considered as a supply node.
4. Find a minimum-cost flow in D' satisfying the demands and supplies of all nodes. Let the output $y^* = (y^*_a)_{a \in A'}$ be an optimal flow, and y^*_{ij}, y^*_{ji}, and $y^*_{ji'}$ denote the flow values along the arcs $(i, j), (j, i) and (j, i)'$ respectively.
5. Construct the output graph $G_{LP} = (V, U, D)$ using the following rules:
   (a) If $y^*_{ji'} = 0$, then put $(y^*_{ij} + 1)$ copies of arcs in the output graph. y^*_{ji} will be 0 in this case.
   (b) If $y^*_{ji'} = 2$, then put $(y^*_{ji} + 1)$ copies of arcs in the output graph. y^*_{ij} will be 0 in this case.
   (c) If $y^*_{ji'} = 1$, then introduce an undirected edge (i, j) in the output graph. Both y^*_{ij} and y^*_{ji} will be 0 in this case.

Recall that the solution creates circular flows in the graph by forming several directed cycles. This algorithm comes up with such cycles by first directing the edges in the least cost direction, and then coming up with cycles using the minimum-cost flow. Note that, when the input graph has even degree, flow imbalances created in the second step at each node are even, and all the actual flow quantities can be kept even throughout the algorithm. So, $y^*_{ji'} = 1$ never occurs in the flow output (Step 5(c)), hence there are no undirected edges in the solution. Therefore, this LP solution is an optimal solution for even-degree input graphs.

4.1 Overview of the WPP algorithm

Our algorithm is similar in its overall structure to the algorithms for MPP [7, 14]. But it differs significantly in some of the crucial steps. Our algorithm finds two solutions for the windy postman problem and selects the solution of minimum-cost between the two. Both solutions are based on the half-integral optimal solution x^* to the LP formulation of the problem.

The first solution S_1 is computed as follows. Consider the set of fractional edges:

$$U = \{(i,j) \in E \mid x^*_{ij} = 1/2\}.$$

Without loss of generality, we assume that the subgraph induced by the edges of U is acyclic, since any cycle can be oriented in one of its two directions without changing the cost of the solution. Since all x-values are either 0 or $\frac{1}{2}$ or a positive integer, $(i,j) \in U$ iff $(j,i) \in U$. It can be verified that the following solution, S_1, is an integral feasible solution for the WPP instance:

$$x'_{ij} = \begin{cases} 1 & \text{if } (i,j) \in U \\ x_{ij} & \text{if } (i,j) \notin U \end{cases}$$

In other words, S_1 contains all the edges oriented by the LP, and duplicated "undirected" edges. Such duplicated edges can be oriented in opposite directions. Solution S_1 is preferable if the cost of the integral edges in the LP solution is more than the cost of the fractional edges.

The second solution S_2 is obtained as follows. Suppose we can find an Eulerian subgraph S of G that contains all the edges U. Then we can take the two Eulerian orientations of S and "add" each of them to the LP solution, and then select the smaller of the two solutions as S_2. Given an integral solution (not necessarily feasible), we add another oriented tour to it as follows. For edges in U, we orient them in the direction of the Eulerian tour of S. For an edge $(i,j) \notin U$ which is used by the tour of S, if $x_{ij} > 0$ then increment x_{ij} by 1 (by adding another duplicate edge).

Otherwise $x_{ji} > 0$, in which case, we decrement x_{ji} by 1. In order to make sure that the resulting solution is feasible, we have to avoid making $x_{ij} = x_{ji} = 0$ for any edge. Therefore, before adding an orientation of S to the LP solution, we double all integral x-values in the LP optimal solution.

The main difficulty in making the above scheme work is that the graph G may not contain an Eulerian subgraph S that includes U. This happens when there is a cut that is crossed only by an odd number of edges of U. In this case, we are forced to duplicate some edges of U, and this cost has to be accounted. To handle this case, we prove an improved lower bound for the WPP by showing that the cost of duplicating a few edges of U can be accounted in the analysis. Solution S_2 is preferable when the cost of the fractional edges U in the LP solution is more than the cost of the integral edges.

Our algorithm returns the minimum-cost solution out of S_1 and S_2, and we show that its cost is at most $\frac{3}{2}$ times the cost of an integer optimal solution.

4.2 Algorithm Mixed_wpp

The approximation algorithm for WPP is given below:

---

1. Input: A connected graph $G = (V, E)$ with a cost function $c = (c_{ij}, c_{ji})$ for edges of E.
2. Solve the LP for WPP on G and let its output be $G_{LP} = (V, U, D)$. If it contains any undirected cycles, direct them arbitrarily. This does not affect the cost of the solution.
3. Algorithm1: Duplicate each edge of U and direct them in opposite directions. This forms the first solution S_1 for the problem.
4. Algorithm2:

   (a) Consider the subgraph of G_{LP} containing all its arcs and none of its edges. In that subgraph, each arc is part of one of the strongly connected components (SCC's). Shrink each such SCC in G_{LP} to a super node. For two super nodes A and B, set the cost of the edge between A and B to be the minimum of the cost of any edge (v, w) where $v \in A$ and $w \in B$. Find a minimum-cost T-join X that has odd degree at all odd degree super nodes and even degree at other super nodes. X contains a subset of U. Add X to G_{LP} and orient the two copies of each edge in X in opposite directions.

(b) Now each SCC has an even number of edges connecting it to the rest of the graph. Ignoring the direction of arcs, find a minimum-cost T-join L of odd-degree nodes within each SCC, where the cost of an edge (i, j) is its average cost, $(c_{ij} + c_{ji})/2$.

(c) Now, duplicate all the arcs within each SCC.

(d) The union of the undirected edges and the T-join L (in which the directions of the arcs are disregarded) forms an Eulerian graph, and we add its Eulerian orientation to the solution as follows. Starting from an end-node of any undirected edge, walk on the edge and continue walking along the T-join. Whenever the walk goes along the orientation of the arc, add an additional copy of that arc. If the walk goes against the orientation of an arc, remove one of the copies of that arc. When the other end-node of the T-join is reached, walk along undirected edge to reach another odd-degree node, and then walk along the T-join again as defined above. The walk ends when the initial vertex is reached, orienting all the undirected edges along the way. If undirected edges still exist after completing the walk, restart the walk from an end-node of one of such edges.

(e) Finally all undirected edges become oriented. In this solution, all arcs of D that are part of the T-join (computed in Step 4b), will have either one or three copies, and all other arcs of D will have two copies. Let us call this graph as S_3. Now re-do Step 4d, but walk along each cycle in the opposite direction. This will result in another solution S_4. Choose the solution with minimum cost between S_3 and S_4, and let us refer to it as S_2.

5. Output the solution with the lower cost between S_1 and S_2.

---

5 Improved Lower Bound for WPP

Consider an input graph $G = (V, E)$. Let C^* be the cost of an optimal postman tour of G. Suppose, given G as input, LP outputs $G_{LP} = (V, U, D)$. The set $U \subseteq E$ contains the edges of G that were not oriented by LP. The set D contains arcs that satisfy $indegree = outdegree$ at each vertex. Let $C(D)$ and $C(U)$ be the total cost of D and U respectively. Note that $C(D) + C(U) = Z^*(G)$. The subgraph induced by D may contain several SCC's. Suppose we shrink each such SCC into a single vertex and form an undirected graph UG. Recall that the cost of an edge is the average of the costs in its two orientations. Suppose we add a minimum-cost T-join of odd degree nodes to UG, i.e., we are adding a minimum-cost subset of U to make the degree of each vertex in UG even. Let its weight be $C(X)$. We show the following improved lower bound on C^*.

Lemma 2 $Z^*(G) + C(X) \leq C^*$.

In order to prove the above lemma, we first show that when we find an optimal LP solution, if some undirected edges are not oriented (edges in the set U) by LP, then adding additional copies of such edges to the input graph does not change an optimal solution to the LP.

Lemma 3 *Let $G = (V, E)$ be the input graph. Let $G_{LP} = (V, U, D)$ be an optimal LP solution. Let $U = \{u_1, ..., u_k\}$ be the undirected edges of G that were not oriented by LP. Define a new input graph H by adding n_i copies of u_i $(i = 1, \ldots, k)$ to G. Let the set of such additional edges be Y. We claim that $Z^*(H) = Z^*(G) + C(Y)$.*

Proof. Let us write the LP in a different format. Let $G = (V, E)$ be the input graph, and for each edge $(i, j) \in E$, let c_{ij} and c_{ji} be the directional costs. Let d_{ij} and d_{ji} be the orientation costs of that edge. Then the new LP formulation of the WPP is

$$\min Z(G) = \frac{1}{2}\sum_{(i,j)\in E}(c_{ij}+c_{ji}) + \sum_{(i,j)\in E}(d_{ij}x_{ij}+d_{ji}x_{ji}) + \sum_{(i,j)\in E}(c_{ij}y_{ij}+c_{ji}y_{ji})$$

subject to

$$x_{ij}, x_{ji} \leq 1, \ \forall (i,j) \in E \tag{4}$$

$$\sum_{j:(i,j)\in E}(x_{ij}+y_{ij}) - (x_{ji}+y_{ji}) = 0, \ \forall i \in V \tag{5}$$

$$x_{ij}, x_{ji}, y_{ij}, y_{ji} \geq 0, \ \forall (i,j) \in E \tag{6}$$

in which x_{ij} and x_{ji} are the usages of the edge from i to j, and from j to i respectively for orienting the edge, and y_{ij} and y_{ji} are the usages of extra copies of the edge from i to j, and from j to i respectively. This LP forumation has three components: the first component is the average cost of the graph and it does not have any LP variables. The second component captures the orientation cost for the whole graph. Note that this cost is zero or negative in an optimal solution to the LP. The third component captures the additional cost of adding extra copies of edges by the LP.

Now let us consider the components of $G_{LP} = (V, U, D)$. By definition, for each edge (i, j) in U, $x_{ij} = x_{ji} = y_{ij} = y_{ji} = 0$ since these edges are neither oriented or duplicated. For each arc (i, j) in D, $x_{ij} = 1$, $x_{ji} = 0$, $y_{ij} \geq 0$, and $y_{ji} = 0$.

When we add copies of additional edges Y to G to get new input graph H, cost of the first component of the LP is increased by the corresponding average cost $C(Y)$. The addition of extra copies of edges has the following effect on the constraints: for each edge $u_i = (j, k)$, the first constraint of the LP can be written as $x_{jk}, x_{kj} \leq n_i + 1$. In other words, the slack in the corresponding constraint is increased. However, corresponding original constraints were not tight. By LP duality, increasing the capacity of unsaturated edges does not affect the complementary slackness conditions and hence an optimal solution to the LP remains optimal even after the capacities of the edges of U are increased. Hence, $Z^*(H) = Z^*(G) + C(Y)$. ∎

Proof of Lemma 2. Let G^* be an optimal solution for WPP, whose cost is C^*. Consider the nodes of UG, the nodes corresponding to the strongly connected components of D. G^* needs to have additional T-join edges (subset of U) between these components to satisfy the even-degree condition for each component. This T-join costs at least $C(X)$. By Lemma 3, we know that adding X to input graph G will result in an optimal LP solution whose cost is $Z^*(G) + C(X)$. Since the LP is a relaxation of the WPP, the cost of the optimal WPP tour is at least $Z^*(G) + C(X)$. ∎

6 Analysis

The algorithm first runs the LP on input graph $G = (V, E)$ and obtains $G_{LP} = (V, U, D)$, with cost $Z^*(G) = C(D) + C(U)$.

Lemma 4 ALGORITHM1 *outputs a solution that costs no more than* $2C^* - C(D)$.

Proof. ALGORITHM1 duplicates all edges of U, so the cost of solution returned by ALGORITHM1 is $C(D) + 2C(U)$. Since $C(D) + C(U) = Z^*(G) \leq C^*$, we get $C(S_1) \leq 2C^* - C(D)$. ∎

Lemma 5 ALGORITHM2 *outputs a solution that costs no more than* $C^* + C(D)$.

Proof. After adding a minimum-cost T-join of odd-degree directed components (X) and doubling directed components (D), the cost of the intermediate graph is $C(U) + 2C(D) + C(X) = Z^*(G) +$

$C(D)+C(X)$. This is at most $C^*+C(D)$ by Lemma 2. Now, consider the two directed walks computed in Steps 4(d) and 4(e). If one solution adds a particular arc, the other solution deletes that arc. Also, both solutions orient undirected edges in opposite directions. Therefore the sum of the cost of both solutions is at most $2C^*+2C(D)$. Hence the smaller of the two solutions is at most $C^*+C(D)$. ∎

Theorem 1 *Algorithm* MIXED_WPP *finds a solution of cost at most* $\frac{3}{2}C^*$ *to the windy postman problem.*

Proof. When $C(D) \geq \frac{C^*}{2}$, by Lemma 4, ALGORITHM1 outputs a solution with cost at most $2C^*-C(D)$, which is at most $\frac{3}{2}C^*$. When $C(D) \leq \frac{C^*}{2}$, by Lemma 5, ALGORITHM2 outputs a solution with cost at most $C^*+C(D)$, which is less than or equal to $\frac{3}{2}C^*$. Since, algorithm MIXED_WPP chooses the solution with lower cost in Step 5, the cost of solution it outputs is at most $\frac{3}{2}C^*$. ∎

7 Example

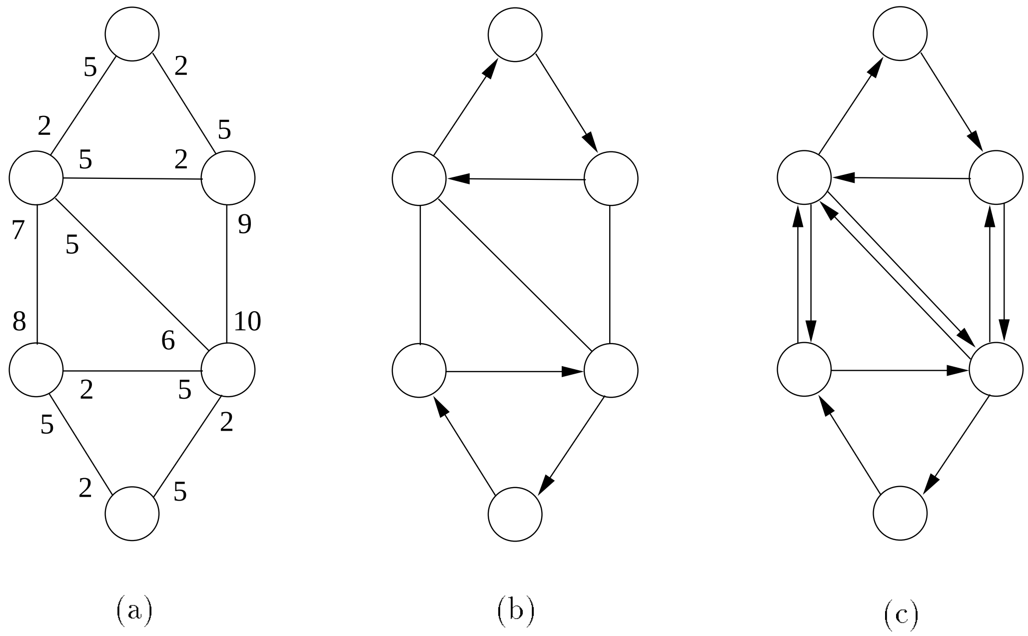


Figure 1: An example for WPP algorithm: Part 1

Figure 1 (a) specifies an input graph for WPP. The directional costs of edges are specified near the tail end of the corresponding arcs. After running the LP in Step 2, we get the graph in Figure 1 (b), which contains two SCC's. ALGORITHM1 duplicates all the undirected edges and orients them in opposite directions, and we get solution S_1 (Figure 1 (c)). As part of ALGORITHM2, we compute a minimum-cost T-join and add it to G_{LP} (Figure 2 (a)). We duplicate all the SCC's in Figure 2 (b). Now the algorithm computes a T-join within each SCC. The T-join edges are shown as dashed lines in Figure 2 (c). Note that the union of the undirected edges and the T-join edges form an Eulerian graph. Adding two orientations of the T-join, we get S_3 (Figure 2 (d)) and S_4 (Figure 2 (e)) respectively. The minimum cost solution among Figure 1 (c), Figure 2 (d), and Figure 2 (e) is Figure 2 (e), with cost 50.

8 Conclusion

We presented an $\frac{3}{2}$ approximation algorithm for the windy postman problem. We used critical properties of intermediate solutions to strengthen the lower bound for WPP optimal solution. We obtained a better approximation ratio by introducing new enhancements to handle the directional costs of edges.

References

[1] A. A. Assad and B. L. Golden. *Arc routing methods and applications*. Elsevier, 1995.

[2] P. Brucker. The Chinese postman problem for mixed networks. In *Proceedings of the International Workshop on Graph theoretic Concepts in Computer Science*, volume 100 of *LNCS*, pages 354–366. Springer, 1980.

[3] N. Christofides, E. Benavent, V. Campos, A. Corberan, and E. Mota. An optimal method for the mixed postman problem. In *System Modelling and Optimization*, volume 59 of *Notes in Control and Information Sciences*. Springer, 1984.

[4] J. Edmonds and E. L. Johnson. Matching, Euler tours and the Chinese postman. *Math. Programming*, 5:88–124, 1973.

[5] H. A. Eiselt. Arc routing problems, Part I: The Chinese postman problem. *Operations Research*, 43:231–242, 1995.

[6] H. A. Eiselt. Arc routing problems, Part II: The rural postman problem. *Operations Research*, 43:399–414, 1995.

[7] G. N. Frederickson. Approximation algorithms for some postman problems. *J. Assoc. Comput. Mach.*, 26:538–554, 1979.

[8] M. Grotchel and Z. Win. A cutting plane algorithm for the windy postman problem. *Math. Programming*, 55:339–358, 1992.

[9] M. Guan. On the windy postman problem. *Discrete Applied Mathematics*, 9:41–44, 1984.

[10] E. Minieka. The Chinese postman problem for mixed networks. *Management Science*, 25:643–648, 1979.

[11] G. L. Nemhauser and Wolsey L. A. *Integer programming and combinatorial optimization.* John Wiley, New York, 1988.

[12] C. H. Papadimitriou. On the complexity of edge traversing. *J. Assoc. Comput. Mach.*, 23:544–554, 1976.

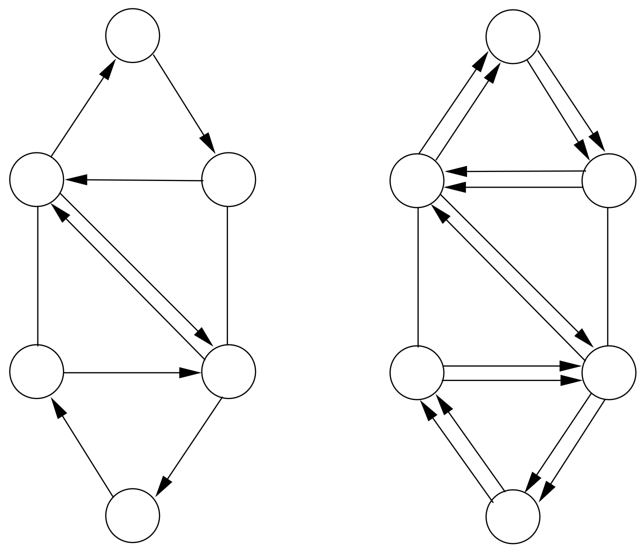
[13] W. L. Pearn and M. L. Li. Algorithms for the windy postman problem. *Computers & Operations Research*, 21:641–651, 1994.

[14] B. Raghavachari and J. Veerasamy. Approximation algorithms for mixed postman problem. *SIAM Journal on Discrete Mathematics*, 12:425–433, 1999.

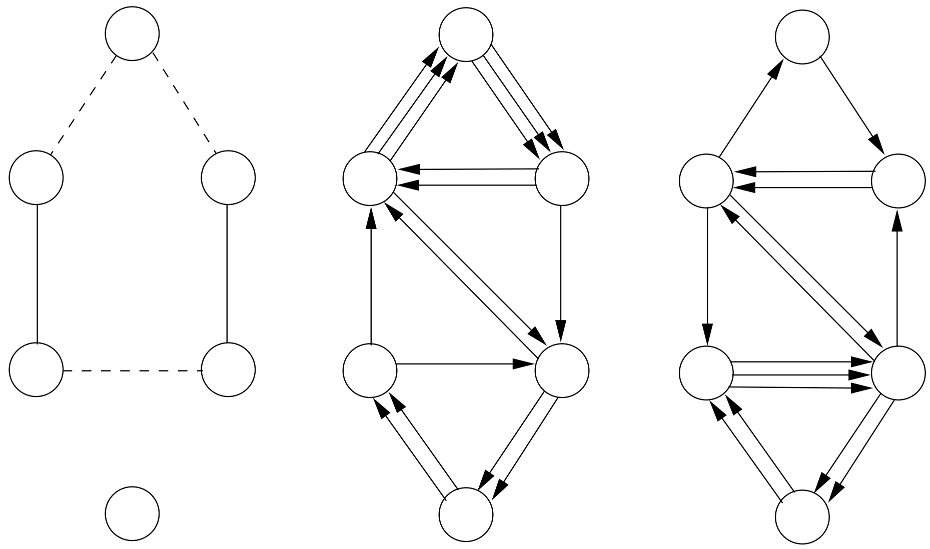
[15] B. Raghavachari and J. Veerasamy. Approximation algorithms for the asymmetric postman problem. In *Proceedings of 10th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 734–741, 1999.

[16] Z. Win. On the windy postman problem on Eulerian graphs. *Math. Programming*, 44:97–112, 1989.

[17] H. Yan and G. L. Thompson. Finding postal carrier walk paths in mixed graphs. *Computational Optimization and Applications*, 9:229–247, 1998.



(a) (b)



(c) (d) (d)

Figure 2: An example for WPP algorithm: Part 2