

# Asymmetric Traveling Salesman Path and Directed Latency Problems

Zachary Friggstad\*

Mohammad R. Salavatipour†

Zoya Svitkina‡

Department of Computing Science  
University of Alberta  
Edmonton, Alberta T6G 2E8, Canada

July 6, 2009

## Abstract

We study integrality gaps and approximability of two closely related problems on directed graphs. Given a set  $V$  of  $n$  nodes in an underlying asymmetric metric  $d$  and two specified nodes  $s$  and  $t$ , both problems ask to find an  $s$ - $t$  path visiting all other nodes. In the *asymmetric traveling salesman path problem* (ATSP), the objective is to minimize the total cost of this path. In the *directed latency problem*, the objective is to minimize the sum of distances on this path from  $s$  to each node. Both of these problems are NP-hard. The best known approximation algorithms for ATSP have ratios  $O(\log n)$  [5, 7], but only a bound of  $O(\sqrt{n})$  for the integrality gap of its linear programming relaxation has been known. For directed latency, the best previously known approximation algorithm has a guarantee of  $O(n^{1/2+\epsilon})$ , for any constant  $\epsilon > 0$  [19].

We present a new algorithm for the ATSP problem that has approximation ratio of  $O(\log n)$ , matching the best known ones, but whose analysis also bounds the integrality gap of the LP relaxation of ATSP by the same factor. This solves an open problem posed in [5]. We then pursue a deeper study of this LP and its variations and their use in approximating directed latency. Our second major result is an  $O(\log n)$ -approximation to the directed latency problem. This also places an  $O(\log n)$  bound on the integrality gap of a new LP relaxation of the latency problem that we introduce. We note that this is essentially the best possible ratio unless an asymptotically better approximation exists for the well-studied asymmetric traveling salesman tour problem.

## 1 Introduction

Let  $G = (V, E)$  be a complete directed graph on a set of  $n$  nodes and let  $d : E \rightarrow \mathbb{R}^+$  be a cost function satisfying the directed triangle inequality  $d_{uw} \leq d_{uv} + d_{vw}$  for all  $u, v, w \in V$ . However,  $d$  is not necessarily symmetric: it may be that  $d_{uv} \neq d_{vu}$  for some nodes  $u, v \in V$ . In the metric Asymmetric Traveling Salesman Path Problem (ATSP), we are also given two distinct nodes  $s, t \in V$ . The goal is to find a path  $s = v_1, v_2, \dots, v_n = t$  that visits all nodes in  $V$  while minimizing

---

\*Supported by NSERC and iCORE scholarships

†Supported by NSERC and an Alberta Ingenuity New Faculty award

‡Supported by Alberta Ingenuity

the sum  $\sum_{i=1}^{n-1} d_{v_i v_{i+1}}$ . ATSP can be used to model scenarios such as minimizing the total cost of travel for a salesperson trying to visit each client on a list of potential customers. This is a variant of the classical Asymmetric Traveling Salesman Problem (ATSP), where the goal is to find a minimum-cost *cycle* visiting all nodes.

Related to ATSP is the directed latency problem. On the same input, the goal is to find a path  $s = v_1, v_2, \dots, v_n = t$  that minimizes the sum of latencies of the nodes. Here, the latency of node  $v_i$  in the path is defined as  $\sum_{j=1}^{i-1} d_{v_j v_{j+1}}$ . The objective can be thought of as minimizing the total waiting time of clients or the average response time. There are possible variations in the problem definition, such as asking for a cycle instead of a path, or specifying only  $s$  but not  $t$ , but they easily reduce to the version that we consider. Other names used in the literature for this problem are the *deliveryman problem* [17] and the *traveling repairman problem* [20].

## 1.1 Previous Work

Both ATSP and the directed latency problem are closely related to the classical traveling salesman problem (TSP), which asks to find the cheapest Hamiltonian cycle in a complete undirected graph with edge costs [15, 11]. In general weighted graphs, TSP is not approximable. However, in most practical settings it can be assumed that edge costs satisfy the triangle inequality (i.e.  $d_{uv} + d_{vw} \geq d_{uw}$ ). Though metric TSP is still NP-hard, the well-known algorithm of Christofides [6] has an approximation ratio of 1.5. Later the analysis in [24] showed that this approximation algorithm actually bounds the integrality gap of a linear programming relaxation for TSP known as the Held-Karp LP. This integrality gap is also known to be at least  $4/3 - o(1)$ . Furthermore, for all  $\epsilon > 0$ , approximating TSP to within a factor of  $220/219 - \epsilon$  is NP-hard [21].

In contrast to the simple constant-factor approximation for TSP, the best known approximation algorithm for its asymmetric version, ATSP, only guarantees a solution of cost within a  $O(\log n)$  factor of the optimum [9, 13, 7]. The ATSP approximation algorithm of [9] was shown to upper-bound the directed Held-Karp LP integrality gap by  $\log_2 n$  in [23], and a different proof that bounds the integrality gap of a slightly weaker LP was obtained in [18]. The best known lower bound on this integrality gap is essentially 2 [3], and tightening these bounds remains an important open problem. ATSP is NP-hard to approximate within  $117/116 - \epsilon$  [21].

The path version of the problem, ATSP, has been studied much less than ATSP, but there are some recent results concerning its approximability. An  $O(\sqrt{n})$  approximation algorithm for it was given by Lam and Newman [14], which was subsequently improved to  $O(\log n)$  by Chekuri and Pal [5]. Feige and Singh [7] improve upon this guarantee by a constant factor and also show that the approximability of ATSP and ATSP are within a constant factor of each other, i.e. an  $\alpha$ -approximation for one implies an  $O(\alpha)$ -approximation for the other. However, all these algorithms are combinatorial and do not bound the integrality gap of the LP relaxation for ATSP. This integrality gap is considered by Nagarajan and Ravi [19], who show that it is at most  $O(\sqrt{n})$ .

The metric minimum latency problem is NP-hard for both the symmetric and directed versions since an exact algorithm for either of these could be used to efficiently solve an NP-hard Hamiltonian path problem. The first constant-factor approximation for metric minimum latency on undirected graphs was developed by Blum et al. [2]. This was subsequently improved in a series of papers from 144 to 21.55 [10], then to 9.28 [1] and ultimately to 3.59 [4]. Blum et al. [2] also observe that there is some constant  $c$  such that there is no  $c$ -approximation for minimum latency unless  $P = NP$ . For directed graphs, Nagarajan and Ravi [19] give a  $O(\rho n^\epsilon \epsilon^{-3})$  approximation where  $\rho$  is the integrality gap of an LP relaxation for ATSP. Using their  $O(\sqrt{n})$  upper bound on this

integrality gap, they obtain a guarantee of  $O(n^{1/2+\epsilon})$  (with running time exponential in  $1/\epsilon$ ), which is the best approximation ratio known for this problem before our present results.

## 1.2 Our Results

In this paper we study both the ATSP and the directed latency problem. The natural LP relaxation for ATSP is the following one with  $\alpha = 1$ :

$$\min \sum_{e \in E} d_e x_e \tag{1}$$

$$\text{s.t. } x(\delta^+(u)) = x(\delta^-(u)) \quad \forall u \in V \setminus \{s, t\} \tag{2}$$

$$x(\delta^+(s)) = x(\delta^-(t)) = 1 \tag{3}$$

$$x(\delta^-(s)) = x(\delta^+(t)) = 0 \tag{4}$$

$$x(\delta^-(S)) \geq \alpha \quad \forall S \subset V, s \notin S \tag{5}$$

$$x_e \geq 0 \quad \forall e$$

Here  $\delta^+(\cdot)$  denotes the set of outgoing edges from a vertex or a set of vertices, and  $\delta^-(\cdot)$  denotes the set of incoming edges. A variable  $x_e$  indicates that edge  $e$  is included in a solution. Let us refer to this linear program as  $\text{LP}(\alpha)$ , as we study it for different values of  $\alpha$  in constraints (5). We begin in Section 2 by proving that the integrality gap of  $\text{LP}(\alpha = 1)$  is  $O(\log n)$ .

**Theorem 1.1** *If  $L$  is the cost of a feasible solution to LP (1) with  $\alpha = 1$ , then one can find, in polynomial time, a Hamiltonian path from  $s$  to  $t$  with cost at most  $(2 \log n + 1) \cdot L$ .<sup>1</sup>*

We note that, despite bounding the integrality gap, our algorithm is actually combinatorial and does not require solving the LP. We strengthen the result of Theorem 1.1 by extending it to any  $\alpha$  with  $\frac{1}{2} < \alpha \leq 1$ . This captures the LP of [19], which has  $\alpha = \frac{2}{3}$ , and is also used in our algorithm for the directed latency problem. We prove the following theorem in Section 3.

**Theorem 1.2** *If  $L$  is the cost of a feasible solution to LP (1) with  $\frac{1}{2} < \alpha \leq 1$ , then one can find, in polynomial time, a Hamiltonian path from  $s$  to  $t$  with cost at most  $\frac{6 \log n + 3}{2\alpha - 1} \cdot L$ .*

It is worth observing that this theorem, together with the results of [19], imply a polylogarithmic approximation algorithm for the directed latency problem which runs in quasi-polynomial time. However, to obtain a polynomial-time approximation, we use a completely different approach. For that we consider  $\text{LP}(\alpha)$  for values of  $\alpha$  that include  $\alpha \leq \frac{1}{2}$ . If we allow  $\alpha \leq \frac{1}{2}$  then  $\text{LP}(\alpha)$ , as a relaxation of ATSP, can be shown to have an unbounded integrality gap. However, we prove the following theorem in Section 4.

**Theorem 1.3** *If  $L$  is the cost of a feasible solution to LP (1) with  $\alpha = \frac{1}{k}$ , for integer  $k$ , then one can find, in polynomial time, a collection of at most  $k \cdot \log n$  paths from  $s$  to  $t$ , such that each vertex of  $G$  appears on at least one path, and the total cost of all these paths is at most  $kL \log n$ .*

Given these results concerning  $\text{LP}(\alpha)$ , we study a particular LP relaxation for the directed latency problem in Section 5. We improve upon the  $O(n^{1/2+\epsilon})$ -approximation of [19] substantially by proving the following:

---

<sup>1</sup>All logarithms in this paper are base 2.

**Theorem 1.4** *A solution to the directed latency problem can be found in polynomial time that has cost no more than  $O(\log n) \cdot L$ , where  $L$  is the value of LP relaxation (8), which is also a lower bound on the integer optimum.*

As noted in [19], an  $\alpha$ -approximation to the directed minimum latency problem would imply an  $O(\alpha)$ -approximation for ATSP. Thus, improving the asymptotic guarantee of our result would require new ideas for solving the well-studied ATSP.

## 2 Integrality gap of ATSP

We show that LP relaxation (1) of ATSP with  $\alpha = 1$  has integrality gap of  $O(\log n)$ . Let  $x^*$  be its optimal fractional solution, and let  $L$  be its cost. We define a *path-cycle cover* on a subset of vertices  $W \subseteq V$  containing  $s$  and  $t$  to be the union of one  $s$ - $t$  path and zero or more cycles, such that each  $v \in W$  occurs in exactly one of these subgraphs. The cost of a path-cycle cover is the sum of costs of its edges.

Our approach is an extension of the algorithm by Frieze et al. [9], analyzed by Williamson [23] to bound the integrality gap for ATSP. That algorithm finds a minimum-cost cycle cover on the current set of vertices, chooses an arbitrary representative vertex for each cycle, deletes other vertices, and repeats, at the end combining all the cycle covers into a Hamiltonian cycle. As this is repeated at most  $\log n$  times, and the cost of each cycle cover is at most the cost of the LP, the upper bound of  $\log n$  on the integrality gap is obtained. In our algorithm for ATSP, the analogue of a cycle cover is a path-cycle cover (also used in [14]), whose cost is at most the cost of the LP solution (Lemma 2.1). At the end we combine the edges of  $O(\log n)$  path-cycle covers to produce a Hamiltonian path. However, the whole procedure is more involved than in the case of ATSP cycle. For example, we don't choose arbitrary representative vertices, but use an amortized analysis to ensure that each vertex only serves as a representative a bounded number of times.

The following lemma bounds the cost of a path-cycle cover on an arbitrary subset  $W$  in terms of the LP cost. However, we note that a path-cycle cover of minimum cost can be found by a combinatorial algorithm, using a reduction to minimum-cost perfect matching, as explained in [14].

**Lemma 2.1** *For any subset  $W \subseteq V$  that includes  $s$  and  $t$ , there is a path-cycle cover of  $W$  of total cost at most  $L$ .*

We make use of the following splitting-off theorem, as also done in [18], where splitting off edges  $yv$  and  $vx$  refers to replacing these edges with the edge  $yx$  (unless  $y = x$ , in which case the two edges are just deleted).

**Theorem 2.2 (Frank [8] and Jackson [12])** *Let  $G = (V, E)$  be a Eulerian directed graph and  $vx \in E$ . There exists an edge  $yv \in E$  such that splitting off  $yv$  and  $vx$  does not reduce the directed connectivity from  $u$  to  $w$  for any  $u, w \in V \setminus \{v\}$ .*

This theorem also applies to weighted Eulerian graphs, as weighted edges can be replaced by multiple parallel edges, producing an unweighted Eulerian multigraph.

**Proof of Lemma 2.1:** Consider the graph  $G'$  obtained from  $G$  by assigning capacities  $x_e^*$  to edges  $e \in E$  and adding a dummy edge from  $t$  to  $s$  with unit capacity. From constraints (2)-(4)

of  $\text{LP}(\alpha = 1)$  it follows that this is a weighted Eulerian graph, i.e. one in which the weighted out-degree of every vertex is equal to its weighted in-degree. Constraints (5) and the max-flow min-cut theorem imply that for any  $v \in V$ , the directed connectivity from  $s$  to  $v$  in  $G'$  is at least  $\alpha = 1$ .

We apply the splitting-off operation on  $G'$ , as guaranteed by Theorem 2.2, to vertices in  $V \setminus W$  until all of them are disconnected from the rest of the graph. Let  $G''$  be the resulting graph on  $W$  and let  $x'$  be its edge capacities except for the dummy edge  $ts$  (which was unaffected by the splitting-off process). By Theorem 2.2, the directed connectivity from  $s$  to any  $v \in W$  does not decrease from the splitting-off operations, which means that in  $G''$  it is still at least  $\alpha$ . This ensures that  $x'$  satisfies constraints (5) for all sets  $S \subset W, s \notin S$ , and is a feasible solution to  $\text{LP}(\alpha)$  on the subset  $W$  of vertices. Furthermore, the triangle inequality implies that the cost of  $x'$  is no more than that of  $x^*$ , namely  $L$ .

Now we make the observation that if we remove from  $\text{LP}(\alpha = 1)$  constraints (5) for all but singleton sets, the resulting LP is equivalent to a circulation problem, and thus has an integer optimal solution. Since there is a feasible solution to  $\text{LP}(\alpha = 1)$  on the set  $W$  of cost at most  $L$  (namely  $x'$ ), and removing a constraint can only decrease the optimal objective value, it means that there is an integer solution to the following program that costs no more than  $L$ :

$$\min \sum_e d_e x_e \tag{6}$$

$$\begin{aligned} \text{s.t. } & x(\delta^+(u)) = x(\delta^-(u)) \geq 1 && \forall u \in W, u \neq s, t \\ & x(\delta^+(s)) = x(\delta^-(t)) = 1 && \tag{7} \\ & x(\delta^-(s)) = x(\delta^+(t)) = 0 \\ & x_e \geq 0 && \forall e \end{aligned}$$

In principle, this integer solution can have  $x(\delta^+(u)) > 1$  for some nodes  $u$ . In this case, we find a Euler tour of each component of the resulting graph (with dummy edge  $ts$  added in) and shortcut it over any repeated vertices. This ensures that  $x(\delta^+(u)) = 1$  for all  $u$  without increasing the cost. But such a solution is precisely a path-cycle cover of  $W$ .  $\square$

We consider Algorithm 1. Roughly speaking, the idea is to find a path-cycle cover, select a representative node for each cycle, delete the other cycle nodes, and repeat. Actually, a representative is selected for a component more general than a simple cycle, namely a union of one or more cycles. We ensure that each vertex is selected as a representative at most  $\log n$  times, which means that after  $2 \log n + 1$  iterations, each surviving vertex has participated in the acyclic part of the path-cycle covers at least  $\log n + 1$  times. This allows us at the end to find an  $s$ - $t$  path which spans all the surviving vertices,  $W$ , and consists entirely of edges in the acyclic part,  $F$ , of the union of all the path-cycle covers, using a technique of [19]. Then we insert into it the subpaths obtained from the cyclic part,  $H$ , of the union of path-cycle covers, connected through their representative vertices. We occasionally treat subgraphs satisfying appropriate degree constraints as flows or circulations.

**Lemma 2.3** *During the course of the algorithm, no label  $l_v$  exceeds the value  $\log n$ .*

We first prove an auxiliary claim.

**Claim 2.4** *In each component  $A$  found by Algorithm 1 on line 6, there are two distinct nodes  $x$  and  $y$  such that  $d_x = d_y = 1$ .*

---

**Algorithm 1**

---

```
1: Let a set  $W \leftarrow V$ ; integer labels  $l_v \leftarrow 0$  for all  $v \in V$ ; flow  $F \leftarrow \emptyset$  and circulation  $H \leftarrow \emptyset$ 
2: for  $2 \log_2 n + 1$  iterations do
3:   Find the minimum-cost  $s$ - $t$  path-cycle cover  $F'$  on  $W$ 
4:    $F \leftarrow F + F'$  ▷  $F$  is acyclic before this operation
5:   Find a path-cycle decomposition of  $F$ , with cycles  $C_1 \dots C_k$  and paths  $P_1 \dots P_h$ , such that
    $\bigcup_i P_i$  is acyclic
6:   for each connected component  $A$  of  $\bigcup_j C_j$  do ▷  $A$  is a circulation
7:     For each vertex  $u \in A$ , let  $d_u$  be the in-degree of  $u$  in  $A$ 
8:     Find a “representative” node  $v \in A$  minimizing  $l_v + d_v$ 
9:      $F \leftarrow F - A$  ▷ subtract flows
10:    for each  $w \in A$ ,  $w \neq v$ , and for each path  $P_i$ 
11:      if  $w \in P_i$  then modify  $F$  by shortcutting  $P_i$  over  $w$ 
12:    Remove all nodes in  $A$ , except  $v$ , from  $W$  ▷ Note: they don't participate in  $F$  anymore
13:     $H \leftarrow H + A$  ▷ add circulations
14:     $l_v \leftarrow l_v + d_v$ 
15:  end for
16: end for
17: Let  $P$  be an  $s$ - $t$  path consisting of nodes in  $W$  in the order found by topologically sorting  $F$ 
▷  $F$  is an acyclic flow on the nodes  $W$ 
18: for every connected component  $X$  of  $H$  of size  $|X| > 1$  do
19:   Find a Euler tour of  $X$ , shortcut over nodes that appear more than once
20:   Incorporate the resulting cycle into  $P$  using a shared node
21: end for
22: return  $P$  ▷  $P$  is a Hamiltonian  $s$ - $t$  path
```

---

**Proof.** Let  $\bar{F}$  be the value of  $F$  at the start of the current iteration of the outside loop, i.e. before  $F'$  is added to it on line 4.  $\bar{F}$  is acyclic, because during the course of the loop, all cycles of  $F$  are subtracted from it. So  $A$  is a union of cycles, formed from the sum of an acyclic flow  $\bar{F}$  and a path-cycle cover  $F'$ , which sends exactly one unit of flow through each vertex.

Consider a topological ordering of nodes based on the flow  $\bar{F}$ , and let  $x$  and  $y$  be the first and last nodes of  $A$ , respectively, in this ordering. As  $A$  always contains at least two nodes,  $x$  and  $y$  are distinct. Since  $x$  and  $y$  participate in some cycle(s) in  $A$ , their in-degrees are at least 1. We now claim that the in-degree of  $x$  in  $A$  is at most 1. Indeed, since all other nodes of  $A$  are later than  $x$  in the topological ordering, it cannot have any flow coming from them in  $\bar{F}$ . So the only incoming flow to  $x$  can be in  $F'$ . But since  $F'$  sends a flow of exactly one unit through each vertex, the in-degree of  $x$  in  $A$  is at most one. A symmetrical argument can be made for  $y$ , showing that its out-degree in  $A$  is at most one. But since  $A$  is a union of cycles, every node's in-degree is equal to its out-degree, and the in-degree of  $y$  is also at most 1. □

**Proof of Lemma 2.3** As the algorithm proceeds, let us construct a forest on the set of nodes  $V$ . Initially, each node is the root of its own tree. We maintain the invariant that  $W$  is the set of tree roots in this forest. For each component  $A$  that the algorithm considers, and the node  $v$  found on line 8, we attach the nodes of  $A$ , except  $v$ , as children of  $v$ . Note that the invariant is maintained, as these nodes are removed from  $W$  on line 12. The set of nodes of each component  $A$  found on

line 6 is always a subset of  $W$ , and thus our construction indeed produces a forest.

We show by induction on the steps of the algorithm that if a node has label  $l$ , then its subtree contains at least  $2^l$  leaves. Thus, since there are  $n$  nodes total, no label can exceed  $\log_2 n$ . At the beginning of the algorithm, all labels are 0, and all trees have one leaf each, so the base case holds. Now consider some iteration in which the label of vertex  $v \in A$  is increased from  $l_v$  to  $l_v + d_v$ . By Claim 2.4, there are nodes  $x, y \in A$  (possibly one of them equal to  $v$ ) with  $d_x = d_y = 1$ . Since  $v$  minimizes  $l_u + d_u$  among all vertices  $u \in A$ , we have that  $l_x + d_x \geq l_v + d_v$  and  $l_y + d_y \geq l_v + d_v$ , and thus  $l_x \geq l_v + d_v - 1$  and  $l_y \geq l_v + d_v - 1$ . Thus, by the induction hypothesis, the trees rooted at  $x$  and  $y$  each have at least  $2^{l_v + d_v - 1}$  leaves. Because we update the forest in such a way that  $v$ 's new tree contains all the leaves of trees previously rooted at  $x$  and  $y$ , this tree now has at least  $2 \cdot 2^{l_v + d_v - 1} = 2^{l_v + d_v}$  leaves.  $\square$

**Lemma 2.5** *At the end of the algorithm's main loop, the flow in  $F$  passing through any node  $v \in W$  is equal to  $2 \log n + 1 - l_v$ , and thus (by Lemma 2.3) is at least  $\log n + 1$ .*

**Proof.** There are  $2 \log n + 1$  iterations, each of which adds one unit of flow through each vertex  $v \in W$ . We now claim that for a vertex  $v \in W$ , the amount of flow removed from it is equal to its label,  $l_v$ . Flow is removed from  $v$  only if  $v$  becomes part of some component  $A$ . Now, if it is ever part of  $A$ , but not chosen as a representative on line 8, then it is removed from  $W$ . Thus, we are only concerned about vertices that are chosen as representatives every time that they are part of  $A$ . Such a vertex has flow  $d_v$  going through it in  $A$ , which is the amount subtracted from  $F$ . But since this is also the amount by which its label increases, the lemma follows.  $\square$

We now show that Algorithm 1 returns a Hamiltonian  $s$ - $t$  path of cost at most  $(2 \log_2 n + 1) \cdot L$ .

**Proof of Theorem 1.1.** At the end of the main loop, all nodes of  $V$  are part of either  $W$  or  $H$  or both. So when all components of  $H$  are incorporated into the path  $P$ , all nodes of  $V$  become part of the path. We bound the cost of all the edges used in the final path by the total cost of all the path-cycle covers found on line 3 of the algorithm. We note that at the end of the algorithm, the cost of the flow  $F + H$  is no more than this total.

We claim that when the  $s$ - $t$  path  $P$  is found on line 17,  $F$  contains flow on every edge between consecutive nodes of  $P$ . First, since  $F$  is acyclic, it has a topological ordering. Suppose we find a flow decomposition of  $F$  into paths. There are at most  $2 \log n + 1$  such paths, and, by Lemma 2.5, each vertex of  $W$  participates in at least  $\log n + 1$ , or more than half, of them. This means that any two vertices  $u, v \in W$  must share a path, say  $P'$ , in this decomposition. In particular, suppose that  $v$  immediately follows  $u$  in the path  $P$ . This means that  $v$  appears later than  $u$  in the topological order, so on  $P'$   $v$  comes after  $u$ . Moreover, we claim that on  $P'$ ,  $v$  will be the immediate successor of  $u$ . If not, suppose that there is a node  $w$  that appears between  $u$  and  $v$  in  $P'$ . But this means that in the topological ordering (and thus in  $P$ ),  $w$  will appear after  $u$  and before  $v$ , which contradicts the fact that they are consecutive in  $P$ . So we conclude that there is an edge with flow in  $F$  between any two consecutive nodes of  $P$ , and thus the path  $P$  costs no more than the flow  $F$ .

Regarding  $H$ , we note that it is a sum of cycles, and thus Eulerian. So it is possible to find a Euler tour of each of its components, using only edges with flow in  $H$ . The subsequent shortcutting can only decrease the cost. Thus, the total cost of cycles found on line 19 is no more than the cost of the flow  $H$ . To describe how these cycles are incorporated into the path  $P$ , we need to show that each of them (or, equivalently, each connected component of  $H$ ) shares exactly one node with  $W$  (and thus with  $P$ ). Note that every component  $A$  added to  $H$  contains only nodes that are in

$W$  at that time. Moreover, when this is done, all but one nodes of  $A$  are expelled from  $W$ . So when several components of  $H$  are connected by the addition of  $A$ , the invariant is maintained that there is only one node per component that is shared with  $W$ . Now, suppose that  $v$  is the vertex shared by the cycle obtained from component  $X$  and the path  $P$ . On line 20, we incorporate the cycle into the path by following the path up to  $v$ , then following the cycle up to the predecessor of  $v$ , then connecting it to the successor of  $v$  on the path. By triangle inequality, the resulting longer path costs no more than the sum of costs of the old path and the cycle.  $\square$

### 3 Integrality gap for relaxed ATSP LP

Consider  $LP(\alpha)$  with  $\frac{1}{2} < \alpha \leq 1$ , and say that it has cost  $L$ . We bound its integrality gap for ATSP. As in the proof of Lemma 2.1, we can apply splitting-off to obtain a feasible solution to  $LP(\alpha)$ , of cost at most  $L$ , on a graph induced by a subset of vertices  $W \subseteq V$ . Let  $x$  be such a solution. Lemma 3.1 below shows how to use  $x$  to find a feasible fractional solution to LP (6) on  $W$ , of cost within a constant factor of  $L$ , namely  $\frac{3}{2\alpha-1}L$ . Since LP (6) has integer optimum, there is an integer solution to LP (6) on  $W$ , and thus a path-cycle cover, of cost at most  $\frac{3}{2\alpha-1}L$ . Then we can proceed as in Section 2, applying Algorithm 1 to bound the cost of the resulting ATSP solution by  $2 \log n + 1$  times the path-cycle cover cost. This shows that  $LP(\alpha)$  has integrality gap at most  $\frac{6 \log n + 3}{2\alpha-1}$ , proving Theorem 1.2.

**Lemma 3.1** *Given a solution  $x$  to  $LP(\alpha)$ , with  $\alpha > 1/2$ , on a subset  $W \subseteq V$ , with cost at most  $L$ , a feasible solution to LP (6) on  $W$  of cost at most  $\frac{3}{2\alpha-1}L$  can be found.*

**Proof.** Multiply  $x$  by  $1/\alpha$ . Now it constitutes a flow  $F$  of  $1/\alpha$  units from  $s$  to  $t$ . Constraints (5), restricted to sets of size 1, imply that each node  $u$  now has at least one unit of flow going through it. Find a flow decomposition of  $F$  into paths and cycles, so that the union of the paths is acyclic. Let  $F = F_p + F_c$ , where  $F_p$  is the sum of flows on the paths in our decomposition, and  $F_c$  is the sum of flows on the cycles.

Choose some  $\gamma$  such that  $\frac{1}{2\alpha} < \gamma < 1$ . For any node  $u$  such that the amount of  $F_p$  flow going through  $u$  is less than  $\gamma$ , shortcut any flow decomposition paths that contain  $u$ , so that there is no more  $F_p$  flow going through  $u$ . Let  $U \subseteq W$  be the set of vertices still participating in the  $F_p$  flow. Then each vertex in  $U$  has at least  $\gamma$  units of  $F_p$  flow going through it, and each vertex in  $W \setminus U$  has at least  $1 - \gamma$  units of  $F_c$  flow going through it.

We find a topological ordering of vertices in  $U$  according to  $F_p$  (which is acyclic), and let  $P$  be an  $s$ - $t$  path that visits the nodes of  $U$  in this topological order. We claim that the cost of  $P$  is within a constant factor of the cost of  $F_p$ . The argument for this is similar to one in the proof of Theorem 1.1. Out of  $1/\alpha$  units of flow going from  $s$  to  $t$  in  $F_p$ , each vertex  $u \in U$  carries  $\gamma$  units, which is more than half of the total amount (as  $\gamma > 1/2\alpha$ ). So for any two such vertices  $u$  and  $v$ , there must be shared flow paths that carry flow of at least  $2\gamma - 1/\alpha$  units. In particular, for every two consecutive nodes  $u, v \in P$ ,  $F_p$  must contain such shared paths in which  $v$  immediately follows  $u$ . So the cost of  $P$  is at most  $\frac{1}{2\gamma-1/\alpha}$  times the cost of  $F_p$ .

We now define  $\tilde{x}$  as a flow equal to one unit of  $s$ - $t$  flow on the path  $P$  plus  $\frac{1}{1-\gamma}$  times the flow  $F_c$ . We claim that  $\tilde{x}$  is a feasible solution to LP (6): there is exactly one unit of flow from  $s$  to  $t$  (as  $F_c$  consists of cycles not containing  $s$  or  $t$ ); there is flow conservation at all nodes except  $s$  and  $t$ ; each vertex in  $U$  (and thus in  $P$ ) has at least one unit of flow going through it; and each vertex

in  $W \setminus U$  has at least one unit of flow going through it (as it had at least  $1 - \gamma$  units of  $F_c$  flow). The cost of this solution is at most

$$\frac{1}{2\gamma - 1/\alpha} \cdot \text{cost}(F_p) + \frac{1}{1 - \gamma} \cdot \text{cost}(F_c) \leq \max\left(\frac{1}{2\gamma - 1/\alpha}, \frac{1}{1 - \gamma}\right) \cdot \frac{1}{\alpha} L.$$

If we set  $\gamma = \frac{1}{3} + \frac{1}{3\alpha}$ , which satisfies  $\frac{1}{2\alpha} < \gamma < 1$ , we see that the cost of  $\tilde{x}$  is at most  $\frac{3}{2\alpha - 1} \cdot L$ .  $\square$

## 4 Relaxed ATSP LP with $\alpha \leq 1/2$

Consider LP (1) with  $\alpha = \frac{1}{k} \leq \frac{1}{2}$  for some integer  $k \geq 2$ . It can be shown that, as a relaxation for the ATSP problem, this LP has unbounded integrality gap. We note that if we multiply this LP by  $k$ , it becomes a relaxation for the  $k$  traveling salesmen problem, i.e. the problem of finding  $k$   $s$ - $t$  paths, of shortest total length, that together cover all nodes of  $G$ . An interesting open question is to determine the integrality gap of this LP as a relaxation for this problem.

Let  $L$  be the cost of the optimal solution to  $\text{LP}(\alpha = \frac{1}{k})$ . We show how to find  $k \cdot \log n$  paths from  $s$  to  $t$ , such that each node of  $G$  appears on at least one path, and the total cost of all these paths is at most  $k \log n \cdot L$ . Let us define a  $k$ -path-cycle cover to be a set of  $k$  disjoint paths from  $s$  to  $t$  and zero or more cycles, which together cover all nodes. Like a path-cycle cover, the minimum-cost  $k$ -path-cycle cover can be found by a combinatorial algorithm.

**Lemma 4.1** *For any subset  $W \subseteq V$  that includes  $s$  and  $t$ , there is a  $k$ -path-cycle cover of  $W$  with total cost at most  $kL$ .*

**Proof.** As in the proof of Lemma 2.1, we apply splitting-off to a solution of  $\text{LP}(\alpha)$  to get a solution to  $\text{LP}(\alpha)$  on a subset  $W \subseteq V$  of vertices, of no greater cost. Now, if we multiply this solution by  $k$ , we get a feasible solution to LP (6) with constraints (7) replaced with

$$x(\delta^+(s)) = x(\delta^-(t)) = k.$$

The cost of this solution is no more than  $kL$ . But this LP has an integer optimum, which, possibly after shortcutting, is exactly a  $k$ -path-cycle cover.  $\square$

**Proof of Theorem 1.3.** We start with  $W = V$  and repeat the following until  $W = \{s, t\}$ :

1. Find a  $k$ -path-cycle cover  $F$  of  $W$  of cost at most  $kL$ , as guaranteed by Lemma 4.1.
2. Remove from  $W$  all nodes (except  $s$  and  $t$ ) that participate in paths of  $F$ .
3. For each cycle  $C$  of  $F$ , choose a representative node  $v \in C$ , and remove from  $W$  all nodes of  $C$  except  $v$ .

Let us say that the procedure terminates after  $T$  iterations. As the size of  $W$  halves in each iteration,  $T$  is at most  $\log n$ . In the last iteration, all elements of  $W$  must have participated in the paths, as otherwise there would be a node  $v$  that remains in  $W$  after this iteration. This implies that the graph  $\bigcup F$  is connected. It also has total cost at most  $kL \log n$ , and, if we add  $kT$  edges from  $t$  to  $s$ , becomes Eulerian. This means that we can construct  $kT \leq k \log n$  paths from  $s$  to  $t$ , covering all nodes of  $V$ , out of edges of  $\bigcup F$ .  $\square$

## 5 Approximation algorithm for Directed Latency

We introduce the following LP relaxation for the directed latency problem. The use of  $x_{uw}$  and  $f_{uw}^v$  variables in an integer programming formulation for directed latency was proposed by Mendez-Diaz et al. [16], who do a computational evaluation of the strength of the LP relaxation that they formulate. However, our LP formulation uses a different set of constraints than the one in [16].

In LP (8), a variable  $x_{uw}$  indicates that node  $u$  appears before node  $w$  on the path. Similarly,  $x_{uvw}$  for three distinct nodes  $u, v, w$  indicates that they appear in this order on the path. For every node  $v \neq s$ , we send one unit of flow from  $s$  to  $v$ , and we call it the  $v$ -flow. Then  $f_{uw}^v$  is the amount of  $v$ -flow going through edge  $(u, w)$ , and  $\ell(v)$  is the latency of node  $v$ . To show that this LP is a relaxation of the directed latency problem, given a solution path  $P$ , we can set  $f_{uw}^v = 1$  whenever the edge  $(u, w)$  is in  $P$  and  $v$  occurs later than  $u$  in  $P$ , and  $f_{uw}^v = 0$  otherwise. So  $f^v$  is one unit of flow from  $s$  to  $v$  along the path  $P$ . Also setting the ordering variables  $x_{uw}$  and  $x_{uvw}$  to 0 or 1 appropriately and setting  $\ell(v)$  to the latency of  $v$  in  $P$ , we get a feasible solution to LP (8) of the same cost as the total latency of  $P$ .

$$\min \sum_{v \neq s} \ell(v) \tag{8}$$

$$\text{s.t. } \ell(v) \geq \sum_{uw} d_{uw} f_{uw}^v \quad \forall v \tag{9}$$

$$\ell(v) \geq [d_{su} + d_{uw} + d_{wv}] x_{uvw} \quad \forall u, w, v : |\{u, w, v\}| = 3 \tag{10}$$

$$\ell(t) \geq \ell(v) \quad \forall v$$

$$x_{uw} = x_{vuw} + x_{uvw} + x_{uwv} \quad \forall u, w, v : |\{u, w, v\}| = 3 \tag{11}$$

$$x_{uw} + x_{wu} = 1 \quad \forall u, w : u \neq w \tag{12}$$

$$x_{su} = x_{ut} = 1 \quad \forall u \notin \{s, t\}$$

$$\sum_w f_{wu}^v = \sum_w f_{uw}^v \quad \forall v, u \notin \{s, v\} \tag{13}$$

$$\sum_w f_{sw}^v = \sum_w f_{wv}^v = 1 \quad \forall v$$

$$f_{us}^v = f_{vu}^v = 0 \quad \forall u, v \tag{14}$$

$$\sum_w f_{uw}^v = x_{uv} \quad \forall v, u \neq v \tag{15}$$

$$f_{uv}^v \leq f_{uv}^t \quad \forall u, w, v \tag{16}$$

$$\sum_{u \notin S, w \in S} f_{uw}^v \geq x_{yv} \quad \forall S \subset V, S \not\ni s, y \in S \tag{17}$$

$$x_{uw}, x_{uvw}, f_{uw}^v \geq 0 \quad \forall u, w, v$$

Constraints (13) are the flow conservation constraints for  $v$ -flow at node  $u$ . Constraints (14) ensure that no  $v$ -flow enters  $s$  or leaves  $v$ . Constraints (15) say that  $v$ -flow passes through  $u$  if and only if  $u$  occurs before  $v$ . Since the  $t$ -flow goes through every vertex, when all the variables are in  $\{0, 1\}$ , it defines an  $s$ - $t$  path. We can think of the  $t$ -flow as the universal flow and Constraints (16)

ensure that every  $v$ -flow follows an edge which has a universal flow on it. Constraints (17), in an integer solution, ensure that if a set  $S$  contains some node  $y$  that comes before  $v$  (i.e.  $x_{yv} = 1$ ), then at least one unit of  $v$ -flow enters  $S$ .

We note that a min-cut subroutine can be used to detect violated constraints of type (17), allowing us to solve LP (8) using the ellipsoid method. Our analysis does not actually use Constraints (16), so we can drop them from the LP. Although without these constraints the corresponding integer program may not be an exact formulation for the directed latency problem, we can still find a solution whose cost is within factor  $O(\log n)$  of this relaxed LP.

**Lemma 5.1** *Given a feasible solution to LP (8) with objective value  $L$ , we can find another solution of value at most  $(1 + \frac{1}{n})L$  in which the ratio of the largest to smallest latency  $\ell(\cdot)$  is at most  $n^2$ .*

**Proof.** Let  $(x, \ell, f)$  be a feasible solution with value  $L$ , with  $\ell(t)$  the largest latency value in this solution. Note that  $L \geq \ell(t)$ . Define a new feasible solution  $(x, \ell', f)$  by  $\ell'(v) = \max\{\ell(v), \ell(t)/n^2\}$ . The total increase in the objective function is at most  $n \cdot \frac{\ell(t)}{n^2} \leq L/n$  as there are  $n$  nodes in total. Thus, the objective value of this new solution is at most  $(1 + 1/n)L$ .  $\square$

Using Lemma 5.1 and scaling the edge lengths (if needed), we can assume that we have a solution  $(x, \ell, f)$  satisfying the following:

**Corollary 5.2** *There is a feasible solution  $(x, \ell, f)$  in which the smallest latency is 1 and the largest latency is at most  $n^2$  and whose cost is at most  $(1 + \frac{1}{n})$  times the optimum LP solution.*

Let  $L^*$  be the value (i.e. total latency) of this solution.

The idea of our algorithm is to construct  $s$ - $v$  paths for several nodes  $v$ , such that together they cover all vertices of  $V$ , and then to “stitch” these paths together to obtain one Hamiltonian path. We use our results for ATSP to construct these paths. For this, we observe that parts of a solution to the latency LP (8) can be transformed to obtain feasible solutions to different instances of LP( $\alpha$ ). For example, we can construct a Hamiltonian  $s$ - $t$  path of total length  $O(\log n) \cdot \ell(t)$  as follows. From a solution to LP (8), take the  $t$ -flow defined by the variables  $f_{uv}^t$ , and notice that it constitutes a feasible solution to LP( $\alpha = 1$ ). In particular, since  $x_{yt} = 1$  for all  $y$ , constraints (17) of LP (8) for  $v = t$  imply that the set constraints (5) of LP (1) are satisfied. The objective function value for LP (1) of this solution is at most  $\ell(t)$ . Thus, by Theorem 1.1, we can find the desired path. Of course, this path is not yet a good solution for the latency problem, as even nodes  $v$  with  $\ell(v) \ll \ell(t)$  can have latency in this path close to  $O(\log n) \cdot \ell(t)$ . Our algorithm constructs several paths of different lengths, incorporating most nodes  $v$  into paths of length  $O(\log n) \cdot \ell(v)$ , and then combines these paths to obtain the final solution.

## 5.1 Constructing the paths

Algorithm 2 finds an approximate solution to the directed latency problem, and we now explain how some of its steps are performed. The algorithm maintains a path  $S$ , initially containing only the source, and gradually adds new parts to it. This is done through operation *append* on lines 9, 10, and 16. To append a path  $P$  to  $S$  means to extend  $S$  by connecting its last node to the first node of  $P$  that does not already appear in  $S$ , and then following until the end of  $P$ . For example, if  $S = abc$  and  $P = bdc$ , the result is  $S = abcdce$ . Step 10 appends a set of paths to  $S$ . This just means sequentially appending all paths in the set, in arbitrary order, to  $S$ .

---

**Algorithm 2** Directed minimum latency

---

- 1: Let  $(x, \ell, f)$  be a solution to LP (8). Let  $S$  be the path  $\{s\}$ .
  - 2: Partition the nodes into  $g = \lfloor \log \ell(t) + 1 \rfloor$  sets  $V_1, \dots, V_g$  with  $v \in V_i$  if  $2^{i-1} \leq \ell(v) < 2^i$ .
  - 3: **for**  $i = 1$  to  $g - 1$  **do**
  - 4:     **for**  $j = 1$  to  $2$  **do**
  - 5:         **if**  $V_i \neq \emptyset$  **then**
  - 6:             Let  $v_i^j = \operatorname{argmax}_{v \in V_i} |\{u \in V_i : x_{uv} \geq \frac{1}{2}\}|$       $\triangleright$  this maximizes the size of  $B_i^j$  below
  - 7:             Let  $A_i^j = \{u \in V : x_{uv_i^j} \geq \frac{2}{3} + \frac{2i-2+j}{24 \log n}\}$
  - 8:             Let  $B_i^j = \{u \in V_i : x_{uv_i^j} \geq \frac{1}{2}\}$       $\triangleright |B_i^j| \geq (|V_i| - 1)/2$
  - 9:             Find an  $s$ - $v_i^j$  path  $P_i^j$ , containing  $A_i^j$ , of cost  $\delta_1 \log n \cdot 2^i$ ; append  $P_i^j$  to  $S$ .
  - 10:            Find  $2 \log n$   $s$ - $v_i^j$  paths  $\mathcal{P}_i^j$ , containing  $B_i^j$ , of total cost  $2 \log n \cdot 2^i$ ; append  $\mathcal{P}_i^j$  to  $S$ .
  - 11:             $V_i = V_i \setminus (A_i^j \cup B_i^j \cup \{v_i^j\})$       $\triangleright$  size of  $V_i$  is at least halved
  - 12:         **end if**
  - 13:     **end for**
  - 14:     Let  $V_{i+1} = V_{i+1} \cup V_i$       $\triangleright$  remaining nodes are carried over to the next set
  - 15: **end for**
  - 16: Construct an  $s$ - $t$  path  $P_g$ , containing  $V_g$ , of cost at most  $(2 \log n + 1) \cdot \ell(t)$ . Append  $P_g$  to  $S$ .
  - 17: Shortcut  $S$  over all repeated nodes. Output  $S$ .
- 

Next we describe how to build paths  $P_i^j$  and  $\mathcal{P}_i^j$  in Steps 9 and 10. We described above how to use Theorem 1.1 to build a Hamiltonian  $s$ - $t$  path  $P$  of length  $(2 \log n + 1) \cdot \ell(t)$ , which is used on line 16 of the algorithm. The idea behind building paths  $P_i^j$  and  $\mathcal{P}_i^j$  with their corresponding length guarantees is similar.

To construct  $P_i^j$ , we do the following. Since each node  $u \in A_i^j$  has  $x_{uv_i^j} \geq 2/3$ , it implies that the amount of  $v_i^j$ -flow that goes through  $u$  is at least  $2/3$ . We apply splitting-off on this flow to nodes outside of  $A_i^j$ , and obtain a total of one unit of  $s$ - $v_i^j$  flow over the nodes in  $A_i^j$ , of cost no larger than  $\ell(v_i^j) \leq 2^i$ . This flow satisfies all the constraints of LP( $\alpha = 2/3$ ), including the set constraints (5), which are implied by the set constraints (17) of the latency LP (8), as  $x_{uv_i^j} \geq 2/3$  for  $u \in A_i^j$ . Thus, using Theorem 1.2, we can find a path from  $s$  to  $v_i^j$ , spanning all the nodes of  $A_i^j$ , whose cost is at most  $\delta_1 \log n \cdot 2^i$  for some constant  $\delta_1$ .

To obtain the set of paths  $\mathcal{P}_i^j$ , we look at the  $v_i^j$ -flow going through each node of  $B_i^j$ , whose amount is at least  $\frac{1}{2}$ . After splitting-off all nodes outside of  $B_i^j$ , we get a feasible solution of cost at most  $\ell(v_i^j) \leq 2^i$  to LP( $\alpha = 1/2$ ). By Theorem 1.3, we can find  $2 \log n$  paths, each going from  $s$  to  $v_i^j$ , which together cover all the nodes of  $B_i^j$ , and whose total cost is at most  $2 \log n \cdot 2^i$ .

## 5.2 Connecting the paths

We now bound the lengths of edges introduced by the append operation in the different cases. For a path  $P$ , let  $\operatorname{app}(P)$  be the length of the edge used for appending  $P$  to the path  $S$  in the algorithm.

**Lemma 5.3** *For any  $i, j$ , and path  $P \in \mathcal{P}_i^j$ ,  $\operatorname{app}(P) \leq 6 \cdot 2^i$ . Also,  $\operatorname{app}(P_g) \leq 6 \cdot 2^g$ .*

**Proof.** Let  $u$  be the last node of the path  $S$  before the append operation,  $v_i^j$  be the last node of  $P$ , and  $w$  be the first node of  $P$  that does not appear in  $S$ . We need to bound  $d_{uw}$ , the distance from  $u$  to  $w$ .

We observe that  $x_{wu} \leq 5/6$ . If  $u = s$ , this is trivial. Otherwise,  $u = v_i^{j'}$  is the endpoint of some path constructed in an earlier iteration. Note that  $j' \leq 2$  and  $i' \leq g - 1 \leq \log \ell(t) \leq 2 \log n$  by our assumption that  $\ell(t) \leq n^2$ , which means that  $\frac{5}{6} \geq \frac{2}{3} + \frac{2i'-2+j'}{24 \log n}$ . So, if we had  $x_{wu} > 5/6$ , then  $w$  would be included in the set  $A_i^{j'}$  and in the path  $P_i^{j'}$ , and thus be already contained in  $S$ , which is a contradiction.

Consequently,  $x_{uw} = 1 - x_{wu} \geq 1/6$ . This means that the amount of  $w$ -flow that goes through  $u$  is at least  $1/6$ . Since this flow has to reach  $w$  after visiting  $u$ , it has to cover a distance of at least  $d_{uw}$ , thus adding at least  $\frac{1}{6} \cdot d_{uw}$  to  $\ell(w)$ , the latency of  $w$ . Thus,  $\ell(w) \geq \frac{1}{6}d_{uw}$ , and  $d_{uw} \leq 6\ell(w)$ . Now, if  $w \in \mathcal{P}_i^j$ , it must be in  $B_i^j$ , which, by definition, means that  $w \in V_i$ , and therefore  $\ell(w) \leq 2^i$ . So  $\text{app}(P) = d_{uw} \leq 6 \cdot 2^i$ . If  $w \in P_g$ , then  $\text{app}(P_g) \leq 6\ell(w) \leq 6\ell(t) \leq 6 \cdot 2^g$ .  $\square$

To bound the cost of appending a path  $P_i^j$  to  $S$ , we need an auxiliary lemma.

**Lemma 5.4** *For any  $\epsilon > 0$ , if  $x_{uw} + x_{vw} \geq 1 + \epsilon$ , then  $\ell(v) \geq \epsilon \cdot d_{uw}$ .*

**Proof.** Using Constraint (11) we have:

$$1 + \epsilon \leq x_{uw} + x_{vw} = (x_{vuw} + x_{uvw} + x_{uvv}) + (x_{uuv} + x_{uvu} + x_{vvu}) = 2x_{uvw} + (x_{vuv} + x_{uvv}) + (x_{uvv} + x_{vvu}).$$

On the other hand,  $(x_{vuv} + x_{uvv}) + (x_{uvv} + x_{vvu}) \leq x_{vw} + x_{vu} = 2 - (x_{uw} + x_{uv}) \leq 1 - \epsilon$ , using again Constraint (11), then Constraint (12), and the assumption of the lemma. Therefore,  $2x_{uvw} \geq (1 + \epsilon) - (1 - \epsilon) = 2\epsilon$ , i.e.  $x_{uvw} \geq \epsilon$ . Then the claim follows using Constraint (10).  $\square$

**Lemma 5.5** *For any  $i$  and  $j$ ,  $\text{app}(P_i^j) \leq 24 \log n \cdot 2^i$ .*

**Proof.** Let  $u$ ,  $v_i^j$ , and  $w$  be as in the proof of Lemma 5.3. To bound  $d_{uw}$ , we consider two cases.

**Case 1:** If  $w \in V_i$ , we apply the same proof as for Lemma 5.3 and conclude that  $\text{app}(P_i^j) \leq 6 \cdot 2^i$ .

**Case 2:** If  $w \notin V_i$ , let  $(i', j')$  be an earlier iteration of the algorithm in which node  $u = v_i^{j'}$  was added to  $S$ . Since  $w \notin S$ , it must be that  $w \notin A_i^{j'}$ , and thus  $x_{wu} < \frac{2}{3} + \frac{2i'-2+j'}{24 \log n}$ . On the other hand, since  $w \in A_i^j$ , it must be that  $x_{wv_i^j} \geq \frac{2}{3} + \frac{2i-2+j}{24 \log n}$ . Because  $2i' + j' \leq 2i + j - 1$ , we have

$$x_{uw} + x_{wv_i^j} = (1 - x_{wu}) + x_{wv_i^j} \geq 1 - \frac{2i' - 2 + j'}{24 \log n} + \frac{2i - 2 + j}{24 \log n} \geq 1 + \frac{1}{24 \log n}.$$

Using Lemma 5.4, we get that  $\text{app}(P_i^j) = d_{uw} \leq 24 \log n \cdot \ell(v_i^j) \leq 24 \log n \cdot 2^i$ .  $\square$

**Lemma 5.6** *Suppose that a node  $v$  is first added to the path  $S$  in iteration  $k$  of the outer loop of the algorithm. Then the latency of  $v$  in  $S$  is at most  $\delta_2 \log n \cdot 2^k$ , for some constant  $\delta_2 > 0$ .*

**Proof.** Let  $\text{len}(P)$  denote the length of a path  $P$ . The latency of node  $v$  on  $S$  is at most:

$$\begin{aligned} & \sum_{i=1}^k \sum_{j=1}^2 \left[ \text{len}(P_i^j) + \sum_{P \in \mathcal{P}_i^j} \text{len}(P) + \text{app}(P_i^j) + \sum_{P \in \mathcal{P}_i^j} \text{app}(P) \right] \\ & \leq \sum_{i=1}^k \sum_{j=1}^2 [\delta_1 \log n \cdot 2^i + 2 \log n \cdot 2^i + 24 \log n \cdot 2^i + 2 \log n \cdot 6 \cdot 2^i] \leq \delta_2 \log n \cdot 2^k \quad \square \end{aligned}$$

Suppose that  $n_i$  is the number of nodes that are originally placed into the set  $V_i$ . Since a node  $v$  is originally placed in  $V_i$  if  $\ell(v) \geq 2^{i-1}$ , the value of the LP solution  $L^*$  can be bounded by:

$$L^* = \sum_v \ell(v) \geq \sum_{i=1}^g n_i 2^{i-1}. \quad (18)$$

Let  $n'_i$  denote the size of  $V_i$  at the beginning of iteration  $i$  of the outer loop. Note that  $n'_i$  may be larger than  $n_i$  since some nodes may have been moved to  $V_i$  in Step 14 of the previous iteration.

**Claim 5.7** *For any  $i$ , the size of the set  $V_i$  at the end of iteration  $i$  is at most  $n'_i/4$ .*

**Proof.** Consider the iteration  $(i, j = 1)$ . Note that the vertex  $v_i^j$  is chosen precisely to maximize the number of nodes  $u$  in  $V_i$  with  $x_{uv_i^j} \geq 1/2$ , which is the size of the set  $B_i^j$ . If we imagine a directed graph  $H$  on the set of vertices  $V_i$ , in which an edge  $(u, w)$  exists whenever  $x_{uw} \geq 1/2$ , then  $v_i^j$  is the vertex with highest in-degree in this graph. Now, from Constraint (12), it's not hard to see that some vertex in  $H$  will have in-degree at least  $(n' - 1)/2$ . So the number of nodes removed from  $V_i$  in step 11 of the algorithm is at least  $|B_i^j \cup \{v_i^j\}| \geq n'/2$ , and size of  $V_i$  decreases at least by a factor of two. Similarly, at least half of the remaining nodes of  $V_i$  are removed in the iteration  $j = 2$ , so overall the size of  $V_i$  decreases at least by a factor of four.  $\square$

We now show that the total latency of the final solution  $S$  is at most  $O(\log n) \cdot L^*$ .

**Proof of Theorem 1.4.** From Claim 5.7, it follows that at most a  $1/4$  fraction of the  $n'_i$  nodes that are in  $V_i$  at the beginning of iteration  $i$  are moved to the set  $V_{i+1}$  at the end of this iteration. Thus, for any  $1 < i \leq g$ ,  $n'_i \leq n_i + n'_{i-1}/4$ . This implies that  $n'_i \leq \sum_{h=1}^i n_h/4^{i-h}$ .

Now we claim that the total latency of the solution  $S$  is at most  $\sum_{i=1}^g n'_i \cdot \delta_2 \log n \cdot 2^i$ . This is because at most  $n'_i$  nodes are added to  $S$  in iteration  $i$ , and each such node has latency at most  $\delta_2 \log n \cdot 2^i$  (using Lemma 5.6). Therefore, the total latency of the solution is at most:

$$\begin{aligned} \sum_{i=1}^g n'_i \cdot \delta_2 \log n \cdot 2^i &\leq \sum_{i=1}^g \delta_2 \log n \cdot 2^i \cdot \sum_{h=1}^i \frac{n_h}{4^{i-h}} = \delta_2 \log n \sum_{i=1}^g \sum_{h=1}^i 2^{h-i} \cdot 2^h n_h \\ &\leq \delta_2 \log n \sum_{h=1}^g (1 + \frac{1}{2} + \frac{1}{4} + \dots) \cdot 2^h n_h \leq O(\log n) \cdot L^*, \end{aligned}$$

using the bound on  $n'_i$ , re-ordering the summation, and using inequality (18). Combined with Corollary 5.2, this proves the theorem.  $\square$

## 6 Concluding Remarks

In this paper we present a new algorithm which shows that the integrality gap of ATSP is  $O(\log n)$ . We also show how to use this algorithm to upper bound integrality gap of even more relaxed LPs for ATSP. Then we obtain an algorithm for the directed latency problem whose solution cost is within  $O(\log n)$  of an LP for this problem. This algorithm can be easily extended to the more general setting in which every node of the graph comes with a weight  $c(v)$  and the goal is to find a Hamiltonian  $s$ - $t$  path to minimize the total *weighted* latency, where the weighted latency of a node  $v_i$  is equal to  $c(v) \cdot \sum_{j=1}^{i-1} d_{v_j v_{j+1}}$ . This requires changing the objective function of LP (8) to

$\sum_{v \neq s} c(v)\ell(v)$  and changing the definition of  $v_i^j$  on line 6 of Algorithm 2 to maximize the total weight, instead of the number, of vertices in  $B_i^j$ .

Obtaining a better approximation ratio for each of ATSP and directed latency remains a challenging problem as they are closely related to the classical ATSP. Perhaps the most important open question in this subject is to close the gap between the upper and lower bounds for ATSP. If there is an  $\alpha$ -approximation for ATSP, with  $\alpha$  smaller than  $O(\log n)$ , then using the results of [7] it would imply an  $O(\alpha)$ -approximation for ATSP. However, even in that case, it is not clear if one could obtain a better approximation algorithm for directed latency.

The relaxed LP( $\alpha$ ) for ATSP with  $\alpha \leq 1/2$  (studied in Section 4) is closely related to the problem of finding  $k$   $s$ - $t$  paths covering all nodes, of minimum total cost. Theorem 1.3 can be seen as a bicriteria approximation for it, and an interesting open problem is to improve this result.

## References

- [1] A. Archer and D.P. Williamson. *Faster Approximation Algorithms for the Minimum Latency Problem*. In Proc. of ACM-SIAM Symposium on Discrete Algorithms (SODA) 2003, pp 88-96.
- [2] A. Blum, P. Chalasani, D. Coppersmith, W.R. Pulleyblank, P. Raghavan and M. Sudan. *The minimum latency problem*. In Proc. of the 26th ACM Symposium on the Theory of Computing (STOC), 163–171, 1994.
- [3] M. Charikar, M. Goemans and H. Karloff. *On the integrality ratio for asymmetric TSP*. In Proc. of the 45th IEEE Symposium on Foundations of Computer Science (FOCS), 101–107, 2004.
- [4] K. Chaudhuri, B. Godfrey, S. Rao and K. Talwar. *Paths, trees, and minimum latency tours*. In Proc. of the 44th IEEE Symposium on Foundations of Computer Science (FOCS), 36–45, 2003.
- [5] C. Chekuri and M. Pal. *An  $O(\log n)$  approximation ratio for the asymmetric traveling salesman path problem*. Theory of Computing, 3:197-209, 2007.
- [6] N. Christofides. *Worst-case analysis of a new heuristic for the traveling salesman problem*. Technical report, Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, PA, 1976.
- [7] U. Feige and M. Singh. *Improved approximation ratios for traveling salesperson tours and paths in directed graphs*. In Proc. of APPROX-RANDOM, 104–118, 2007.
- [8] A. Frank. *On connectivity properties of Eulerian digraphs*. Annals of Discrete Mathematics, 41:179-194, 1989.
- [9] A. Frieze, G. Galbiati and F. Maffioli. *On the worst-case performance of some algorithms for the asymmetric traveling salesman problem*. Networks, 12:23–39, 1982.
- [10] M. Goemans and J. Kleinberg. *An improved approximation ratio for the minimum latency problem*. Mathematical Programming 82:111-124, 1998.

- [11] G. Gutin and A. P. Punnen (Eds.). *Traveling Salesman Problem and Its Variations*. Springer, Berlin, 2002.
- [12] B. Jackson. *Some remarks on arc-connectivity, vertex splitting, and orientation in digraphs*. Journal of Graph Theory, 12(3):429-436, 1988.
- [13] H. Kaplan, M. Lewenstein, N. Shafir, and M. Sviridenko. *Approximation algorithms for asymmetric TSP by decomposing directed regular multigraphs*. J. of the ACM, 52(4):602-626, 2005.
- [14] F. Lam and A. Newman. *Traveling salesman path problems*. Mathematical Programming, 113(1):39-59, 2008.
- [15] E. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. Shmoys (Eds.). *The Traveling Salesman Problem: A guided tour of combinatorial optimization*. John Wiley & Sons Ltd., 1985.
- [16] I. Mendez-Diaz, P. Zabala, A. Lucena. *A new formulation for the Traveling Deliveryman Problem*. Discrete Applied Mathematics 156:3223-3237, 2008.
- [17] E. Minieka. *The deliveryman problem on a tree network*. Annals of Operations Research, 18:261-266, 1989.
- [18] V. Nagarajan and R. Ravi. *Poly-logarithmic approximation algorithms for directed vehicle routing problems*. In Proc. of APPROX-RANDOM, 257-270, 2007.
- [19] V. Nagarajan and R. Ravi. *The directed minimum latency problem*. In Proc. of APPROX-RANDOM, 193-206, 2008.
- [20] C.H. Papadimitriou, G. Papageorgiou, N. Papakostantinou, F. Afrati, and S. Cosmadakis. *The complexity of the traveling repairman problem*. Informatique Theorique et Applications, 20:79-87, 1986.
- [21] C.H. Papadimitriou and S. Vempala. *On the approximability of the traveling salesman problem*. In Proc. of the 32nd ACM Symposium on the Theory of Computing (STOC), 136-133, 2000.
- [22] D.B. Shmoys and D.P. Williamson. *Analyzing the Held-Karp TSP bound: a monotonicity property with applications*. Information Processing Letters, 35:281-285, 1990.
- [23] D.P. Williamson. *Analysis of the Held-Karp heuristic for the traveling salesman problem*. M.S. Thesis, Massachusetts Institute of Technology, 1990.
- [24] L.A. Wolsey. *Heuristic analysis, linear programming and branch and bound*. Mathematical Programming Study, 13:121-134, 1980.