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

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 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* (ATSPP), 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 ATSPP had ratio $O(\log n)$ [7,9] until the very recent result that improves it to $O(\log n/\log\log n)$ [3,9]. However, 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 $\epsilon > 0$ [23].

We present a new algorithm for the ATSPP problem that has approximation ratio of $O(\log n)$, but whose analysis also bounds the integrality gap of the standard LP relaxation of ATSPP by the same factor. This solves an open problem posed in [7]. 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.

1 Introduction

Let $G = (V, E)$ be a complete directed graph on a set of n nodes and let $d : E \to \mathbb{R}^+$ be a cost function satisfying the directed triangle inequality $d_{uw} \le d_{uv} + d_{vw}$ for all $u, v, w \in V$. However, d is not necessarily symmetric: it may be that $d_{uv} \ne d_{vu}$ for some nodes $u, v \in V$. In the metric Asymmetric Traveling Salesman Path Problem (ATSPP), we are also given two distinct nodes $s, t \in V$. The goal is to find a path $s = v_1, v_2, \ldots, v_n = t$ that visits all nodes in V while minimizing the sum $\sum_{j=1}^{n-1} d_{v_j v_{j+1}}$. ATSPP can be used to model scenarios such as minimizing the total cost of travel for a person trying to visit a set of cities on the way from a starting point to a destination. 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 ATSPP is the directed latency problem. On the same input, the goal is to find a path $s = v_1, v_2, \ldots, v_n = t$ 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* [21] and the *traveling repairman problem* [1].

1.1 Related work

Both ATSPP 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 [13, 19]. 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. $d_{uw} \le d_{uv} + d_{vw}$). Though metric TSP is still NP-hard, the well-known algorithm of Christofides [8] has an approximation ratio of $3/2$. Later the analysis in [25, 27] 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 $4/3$. Furthermore, for all $\epsilon > 0$, approximating TSP within a factor of $220/219 - \epsilon$ is NP-hard [24]. Christofides' heuristic was adapted to the problem of finding the cheapest Hamiltonian path in a metric graph with an approximation guarantee of $3/2$ if at most one endpoint is specified or $5/3$ if both

*Supported by NSERC and iCORE scholarships
†Supported by NSERC and an Alberta Ingenuity New Faculty award
‡Supported by Alberta Ingenuity

endpoints are given [14].

In contrast to TSP, no constant-factor approximation for its asymmetric version is known. The current best approximation for ATSP is the very recent result of Asadpour et al. [3], which gives an $O(\log n/\log\log n)$-approximation algorithm. It also upper-bounds the integrality gap of the asymmetric Held-Karp LP relaxation by the same factor. Previous algorithms guarantee a solution of cost within $O(\log n)$ factor of optimum [9,11,16,17]. The algorithm of Frieze et al. [11] is shown to upper-bound the Held-Karp integrality gap by $\log_2 n$ in [26], and a different proof that bounds the integrality gap of a slightly weaker LP is obtained in [22]. The best known lower bound on the Held-Karp integrality gap is essentially 2 [5], and tightening these bounds remains an important open problem. ATSP is NP-hard to approximate within $^{117}/_{116} - \epsilon$ [24].

The path version of the problem, ATSPP, 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 [18], which was subsequently improved to $O(\log n)$ by Chekuri and Pal [7]. Feige and Singh [9] improved upon this guarantee by a constant factor and also showed that the approximability of ATSP and ATSPP are within a constant factor of each other, i.e. an α-approximation for one implies an $O(\alpha)$-approximation for the other. Combined with the result of [3], this implies an $O(\log n/\log\log n)$ approximation for ATSPP. However, none of these algorithms bound the integrality gap of the LP relaxation for ATSPP. This integrality gap is considered by Nagarajan and Ravi [23], who showed that it is at most $O(\sqrt{n})$.

The metric minimum latency problem is NP-hard for both the undirected and directed versions since an exact algorithm for either of these could be used to efficiently solve the Hamiltonian Path problem. The first constant-factor approximation for minimum latency on undirected graphs was developed by Blum et al. [4]. This was subsequently improved in a series of papers from 144 to 21.55 [12], then to 7.18 [2] and ultimately to 3.59 [6]. Blum et al. [4] also observed 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 [23] gave an $O((\rho+\log n)\, n^{\epsilon}\, \epsilon^{-3})$ approximation algorithm that runs in time $n^{O(1/\epsilon)}$, where ρ is the integrality gap of an LP relaxation for ATSPP. Using their $O(\sqrt{n})$ upper bound on ρ, they obtained a guarantee of $O(n^{1/2+\epsilon})$, 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 ATSPP and the directed latency problem. The

$$
\begin{aligned}
(1.1)\quad & \min \quad \sum_{e\in E} d_e x_e \\
(1.2)\quad & \text{s.t.} \quad x(\delta^+(u)) = x(\delta^-(u)) && \forall u \in V\setminus\{s,t\} \\
(1.3)\quad & \qquad x(\delta^+(s)) = x(\delta^-(t)) = 1 \\
(1.4)\quad & \qquad x(\delta^-(s)) = x(\delta^+(t)) = 0 \\
(1.5)\quad & \qquad x(\delta^-(S)) \geq \alpha && \forall S \subset V, S \neq \emptyset, s \notin S \\
& \qquad x_e \geq 0 && \forall e \in E
\end{aligned}
$$

natural LP relaxation for ATSPP is (1.1) with $\alpha = 1$, where $\delta^+(\cdot)$ 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 LP(α), as we study it for different values of α in constraints (1.5). We begin by proving that the integrality gap of LP($\alpha = 1$) is $O(\log n)$.

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

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 α with $\frac{1}{2} < \alpha \leq 1$. This captures the LP of [23], which has $\alpha = \frac{2}{3}$, and is also used in our algorithm for the directed latency problem.

THEOREM 1.2. *If L is the cost of a feasible solution to LP (1.1) 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 [23], imply a polylogarithmic approximation algorithm for the directed latency problem which runs in quasi-polynomial time, as well as a polynomial-time $O(n^{\epsilon})$-approximation. However, that approach relies on guessing a large number of intermediate vertices of the path, and thus does not yield an algorithm that has both a polynomial running time and a polylogarithmic approximation guarantee. So, to obtain a polynomial-time approximation, we use a different approach. For that we consider LP(α) for values of

[1] All logarithms in this paper are base 2.

α that include $\alpha \leq \frac{1}{2}$. If we allow $\alpha \leq \frac{1}{2}$ then LP(α), as a relaxation of ATSPP, can be shown to have an unbounded integrality gap. However, we obtain the following theorem.

THEOREM 1.3. *If L is the cost of a feasible solution to LP (1.1) 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 LP(α), we study a particular LP relaxation for the directed latency problem. We improve upon the $O(n^{1/2+\epsilon})$-approximation of [23] substantially by proving the following:

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 (3.8), which is also a lower bound on the integer optimum.*

We note that this seems to be the first time that a bound is placed on the integrality gap of any LP-relaxation for the minimum latency problem, even in the undirected case.

We consider the ATSPP problem in Section 2, where we prove Theorem 1.1. Section 3 contains the algorithm for directed latency and the proof of Theorem 1.4. Proofs of Theorems 1.2 and 1.3 are omitted from this version of the paper.

2 Integrality gap of ATSPP

We show that LP relaxation (1.1) of ATSPP 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. [11], analyzed by Williamson [26] 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 of the cycles, 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 solution, the upper bound of $\log n$ on the integrality gap is obtained. In our algorithm for ATSPP, the analogue of a cycle cover is a path-cycle cover (also used in [18]), 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.

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 [18]. In the proof of Lemma 2.1 below, we make use of the following splitting-off theorem, as also done in [22], 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 2.1. (FRANK [10] AND JACKSON [15]) *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, i.e. ones in which the weighted out-degree of every vertex is equal to its weighted in-degree, since weighted edges can be replaced by multiple parallel edges, producing an unweighted Eulerian multigraph.

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

Proof. 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 (1.2)-(1.4) of LP($\alpha = 1$) it follows that this is a weighted Eulerian graph. Constraints (1.5) 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 2.1, 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 2.1, 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 α. This ensures that x' satisfies constraints (1.5) for all sets $S \subset W$ with $S \neq \emptyset$ and $s \notin S$, and is a feasible solution to LP(α) 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 LP($\alpha = 1$) constraints (1.5) 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 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:

$$\begin{aligned}
(2.6)\quad \min \quad & \sum_{e \in E} d_e x_e \\
\text{s.t.} \quad & x(\delta^+(u)) = x(\delta^-(u)) \geq 1 \quad \forall u \in W \setminus \{s,t\} \\
(2.7)\qquad & x(\delta^+(s)) = x(\delta^-(t)) = 1 \\
& x(\delta^-(s)) = x(\delta^+(t)) = 0 \\
& x_e \geq 0 \qquad \forall e \in E
\end{aligned}$$

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. □

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 [23]. 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 2.2. *During the course of the algorithm, no label l_v exceeds the value $\log n$.*

The idea of the proof is, as the algorithm proceeds, to maintain a forest on the set of nodes V, such that the number of leaves in a subtree rooted at any node $v \in V$ is at least 2^{l_v}. The lemma then follows because the total number of leaves is at most n. We first prove an auxiliary claim.

CLAIM 2.1. *In each component A found by Algorithm 1 on line 6, there are two distinct nodes x and y such that $d_x = d_y = 1$.*

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 2.2. 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 2.1, 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

Algorithm 1 Asymmetric Traveling Salesman Path

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 $C_1 ... C_k$ and paths $P_1 ... P_h$, 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

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. □

LEMMA 2.3. *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 2.2) 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. □

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. This is similar to an argument used in [23]. 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 2.3, 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 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 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. □

3 Algorithm for directed latency

We introduce LP relaxation (3.8) for the directed latency problem. The use of x_{uw} and f^v_{uw} variables in an integer programming formulation for directed latency was proposed by Mendez-Diaz et al. [20], who do a computational evaluation of the strength of its LP relaxation. However, our LP formulation uses a different set of constraints than the one in [20].

In LP (3.8), 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^v_{uw} 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 $f^v_{uw} = 1$ whenever the edge (u, w) is in P and v occurs later than u in P, and $f^v_{uw} = 0$ 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 (3.8) of the same cost as the total latency of P.

Constraints (3.12) are the flow conservation constraints for v-flow at node u. Constraints (3.13) ensure that no v-flow enters s or leaves v. Constraints (3.14) 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 (3.15) ensure that every v-flow follows an edge which has a universal flow on it. Constraints (3.16), 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 (3.16), allowing us to solve LP (3.8) using the ellipsoid method. Our analysis does not actually use Constraints (3.15), 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.

$$
\begin{aligned}
(3.8)\quad & \min \sum_{v \neq s} \ell(v) \\
& \text{s.t. } \ell(v) \geq \sum_{uw} d_{uw} f^v_{uw} && \forall v \\
(3.9)\quad & \ell(v) \geq [d_{su} + d_{uw} + d_{wv}]\, x_{uwv} \\
& && \forall u, w, v : |\{u, w, v\}| = 3 \\
& \ell(t) \geq \ell(v) && \forall v \\
(3.10)\quad & x_{uw} = x_{vuw} + x_{uvw} + x_{uwv} \\
& && \forall u, w, v : |\{u, w, v\}| = 3 \\
(3.11)\quad & x_{uw} + x_{wu} = 1 && \forall u, w : u \neq w \\
& x_{su} = x_{ut} = 1 && \forall u \notin \{s, t\} \\
(3.12)\quad & \sum_w f^v_{wu} = \sum_w f^v_{uw} && \forall v, \forall u \notin \{s, v\} \\
& \sum_w f^v_{sw} = \sum_w f^v_{wv} = 1 && \forall v \\
(3.13)\quad & f^v_{us} = f^v_{vu} = 0 && \forall u, v \\
(3.14)\quad & \sum_w f^v_{uw} = x_{uv} && \forall v, u \neq v \\
(3.15)\quad & f^v_{uw} \leq f^t_{uw} && \forall u, w, v \\
(3.16)\quad & \sum_{u \notin S, w \in S} f^v_{uw} \geq x_{yv} && \forall S \subset V \setminus \{s\}, y \in S \\
& x_{uw}, x_{uwv}, f^v_{uw} \geq 0 && \forall u, w, v
\end{aligned}
$$

Lemma 3.1. *Given a feasible solution to LP (3.8) 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 $\ell(\,)$ is at most n^2.*

Proof. Let (x, ℓ, 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, ℓ', 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 $(1 + 1/n)L$. □

Using Lemma 3.1 and scaling the edge lengths (if needed), we can assume that we have a solution (x, ℓ, f) satisfying the following:

COROLLARY 3.1. *There is a feasible solution (x, ℓ, 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 ATSPP to construct these paths. For this, we observe that parts of a solution to the latency LP (3.8) can be transformed to obtain feasible solutions to different instances of LP(α). 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 (3.8), take the t-flow defined by the variables f^t_{uw}, and notice that it constitutes a feasible solution to LP($\alpha = 1$). In particular, since $x_{yt} = 1$ for all y, constraints (3.16) of LP (3.8) for $v = t$ imply that the set constraints (1.5) of LP (1.1) are satisfied. The objective function value for LP (1.1) 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.

3.1 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 $S = sabc$ and $P = sbdce$, the result is $S = sabcdce$. Step 10 appends a set of paths to S. This just means sequentially appending all paths in the set, in arbitrary order, to S.

Next we describe how to build paths P^j_i and $\mathcal{P}^j_i$ 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^j_i and $\mathcal{P}^j_i$ with their corresponding length guarantees is similar.

To construct P^j_i, we do the following. Since each node $u \in A^j_i$ has $x_{uv^j_i} \geq 2/3$, the amount of v^j_i-flow that goes through u is at least $2/3$. We apply splitting-off on this flow to nodes outside of A^j_i, and obtain a total of one unit of s-v^j_i flow over the nodes in A^j_i, of cost no larger than $\ell(v^j_i) \leq 2^i$. This flow satisfies all the constraints of LP($\alpha = 2/3$), including the set constraints (1.5), which are implied by the set constraints (3.16) of the latency LP (3.8), as $x_{uv^j_i} \geq 2/3$ for $u \in A^j_i$. Thus, using Theorem 1.2, we can find a path from s to v^j_i, spanning all the nodes of A^j_i, whose cost is at most $\delta_1 \log n \cdot 2^i$ for some constant δ_1.

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

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

LEMMA 3.2. *For any i, j, and path $P \in \mathcal{P}^j_i$, $app(P) \leq 6 \cdot 2^i$. Also, $app(P_g) \leq 6 \cdot 2^g$.*

Proof. Let u be the last node of the path S before the append operation, v^j_i 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, $u = v^{j'}_{i'}$ 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 $A^{j'}_{i'}$ and in the path $P^{j'}_{i'}$, 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 $w \in \mathcal{P}^j_i$, it must be in B^j_i,

Algorithm 2 Directed Latency

Let (x, ℓ, f) be a solution to LP (3.8). Let S be the path $\{s\}$.
Partition the nodes into $g = \lfloor \log \ell(t) + 1 \rfloor$ sets $V_1, \ldots, V_g$ with $v \in V_i$ if $2^{i-1} \leq \ell(v) < 2^i$.
for $i = 1$ to $g - 1$ **do**
  for $j = 1$ to 2 **do**
    if $V_i \neq \emptyset$ **then**
      Let $v_i^j = \text{argmax}_{v \in V_i} |\{u \in V_i : x_{uv} \geq \frac{1}{2}\}|$ ▷ this maximizes the size of B_i^j below
      Let $A_i^j = \{u \in V : x_{uv_i^j} \geq \frac{2}{3} + \frac{2i-2+j}{24 \log n}\}$
      Let $B_i^j = \{u \in V_i : x_{uv_i^j} \geq \frac{1}{2}\}$ ▷ $|B_i^j| \geq (|V_i| - 1)/2$
      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.
      Find $2 \log n$ s-v_i^j paths $\mathcal{P}_i^j$, containing B_i^j, of total cost at most $2 \log n \cdot 2^i$; append $\mathcal{P}_i^j$ to S.
      $V_i = V_i \setminus (A_i^j \cup B_i^j \cup \{v_i^j\})$ ▷ size of V_i is at least halved
    end if
  end for
  Let $V_{i+1} = V_{i+1} \cup V_i$ ▷ remaining nodes are carried over to the next set
end for
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.
Shortcut S over the later copies of repeated nodes. Output S.

which, by definition, means that $w \in V_i$, and therefore $\ell(w) \leq 2^i$. So $app(P) = d_{uw} \leq 6 \cdot 2^i$. If $w \in P_g$, then $app(P_g) \leq 6\ell(w) \leq 6\ell(t) \leq 6 \cdot 2^g$. □

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

LEMMA 3.3. *For any $\epsilon > 0$, if $x_{uw} + x_{wv} \geq 1 + \epsilon$, then $\ell(v) \geq \epsilon \cdot d_{uw}$.*

Proof. Using Constraint (3.10) we have:

$$\begin{aligned} 1 + \epsilon &\leq x_{uw} + x_{wv} \\ &= (x_{vuw} + x_{uvw} + x_{uwv}) + (x_{uwv} + x_{wuv} + x_{wvu}) \\ &= 2x_{uwv} + (x_{vuw} + x_{uvw}) + (x_{wuv} + x_{wvu}). \end{aligned}$$

On the other hand, $(x_{vuw} + x_{uvw}) + (x_{wuv} + x_{wvu}) \leq x_{vw} + x_{wu} = 2 - (x_{uw} + x_{wv}) \leq 1 - \epsilon$, using again Constraint (3.10), then Constraint (3.11), and the assumption of the lemma. Therefore, $2x_{uwv} \geq (1+\epsilon) - (1-\epsilon) = 2\epsilon$, i.e. $x_{uwv} \geq \epsilon$. Then the claim follows using Constraint (3.9). □

LEMMA 3.4. *For any i and j, $app(P_i^j) \leq 24 \log n \cdot 2^i$.*

Proof. Let u, v_i^j, and w be as in the proof of Lemma 3.2. To bound d_{uw}, we consider two cases.

Case 1: If $w \in V_i$, we apply the same proof as for Lemma 3.2 and conclude that $app(P_i^j) \leq 6 \cdot 2^i$.

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

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

Using Lemma 3.3, we get that $app(P_i^j) = d_{uw} \leq 24 \log n \cdot \ell(v_i^j) \leq 24 \log n \cdot 2^i$. □

LEMMA 3.5. *Suppose that a node v is first added to 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 $len(P)$ denote the length of a path P. The latency of node v on S is at most:

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

□

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:

$$L^* \;=\; \sum_v \ell(v) \;\geq\; \sum_{i=1}^{g} n_i \, 2^{i-1}. \tag{3.17}$$

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 3.1. *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 (3.11), 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. □

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 3.1, 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 3.5). Therefore, the total latency of the solution is at most:

$$\begin{aligned}
\sum_{i=1}^{g} n'_i \cdot \delta_2 \log n \cdot 2^i \;&\leq\; \sum_{i=1}^{g} \delta_2 \log n \cdot 2^i \cdot \sum_{h=1}^{i} \frac{n_h}{4^{i-h}} \\
&=\; \delta_2 \log n \sum_{i=1}^{g} \sum_{h=1}^{i} 2^{h-i} \cdot 2^h \, n_h \\
&\leq\; \delta_2 \log n \sum_{h=1}^{g} 2^h \, n_h \sum_{i=0}^{\infty} \frac{1}{2^i} \\
&\leq\; O(\log n) \cdot L^*,
\end{aligned}$$

using the bound on n'_i, re-ordering the summation, and using inequality (3.17). Combined with Corollary 3.1, this proves the theorem. □

4 Concluding remarks

In this paper we present a new algorithm for ATSPP which shows that the integrality gap of its standard LP relaxation is $O(\log n)$. We also show how to use this algorithm to upper-bound the integrality gaps of even more relaxed LPs for ATSPP. 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 (3.8) 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.

We note that our approximation guarantee for directed latency is of the form $O(\gamma + \log n)$, where γ is the integrality gap of ATSPP. So an improvement of the bound on γ would not immediately lead to an improvement for directed latency. The relaxed LP(α) for ATSPP with $\alpha \leq 1/2$ 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.

Acknowledgements

This work was partly done while the second author was visiting Microsoft Research New England; he thanks MSR for hosting him. We also thank the anonymous referees for helpful comments.

References

[1] F. N. Afrati, S. S. Cosmadakis, C. H. Papadimitriou, G. Papageorgiou, and N. Papakostantinou. The complexity of the travelling repairman problem. *Informatique Theorique et Applications*, 20(1):79–87, 1986.

[2] A. Archer, A. Levin, and D. P. Williamson. A faster, better approximation algorithm for the minimum latency problem. *SIAM J. Comput.*, 37(5):1472–1498, 2008.

[3] A. Asadpour, M. X. Goemans, A. Madry, S. Oveis Gharan, and A. Saberi. An $O(\log n/\log\log n)$-approximation algorithm for the asymmetric traveling salesman problem. In *Proc. 21st ACM Symp. on Discrete Algorithms*, 2010.

[4] A. Blum, P. Chalasani, D. Coppersmith, B. Pulleyblank, P. Raghavan, and M. Sudan. The minimum latency problem. In *Proc. 26th ACM Symp. on Theory of Computing*, 1994.

[5] M. Charikar, M. X. Goemans, and H. Karloff. On the integrality ratio for asymmetric TSP. In *Proc. 45th IEEE Symp. on Foundations of Computer Science*, 2004.

[6] K. Chaudhuri, B. Godfrey, S. Rao, and K. Talwar. Paths, trees, and minimum latency tours. In *Proc. 44th IEEE Symp. on Foundations of Computer Science*, 2003.

[7] C. Chekuri and M. Pal. An $O(\log n)$ approximation ratio for the asymmetric traveling salesman path problem. *Theory of Computing*, 3(1):197–209, 2007.

[8] 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.

[9] U. Feige and M. Singh. Improved approximation ratios for traveling salesperson tours and paths in directed graphs. In *Proc. 10th APPROX*, 2007.

[10] A. Frank. On connectivity properties of Eulerian digraphs. *Ann. Discrete Math.*, 41:179–194, 1989.

[11] 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.

[12] M. Goemans and J. Kleinberg. An improved approximation ratio for the minimum latency problem. *Math. Program.*, 82:111–124, 1998.

[13] G. Gutin and A. P. Punnen, editors. *Traveling Salesman Problem and Its Variations*. Springer, Berlin, 2002.

[14] J. A. Hoogeveen. Some paths are more difficult than cycles. *Oper. Res. Lett.*, 10:291–295, 1991.

[15] B. Jackson. Some remarks on arc-connectivity, vertex splitting, and orientation in digraphs. *Journal of Graph Theory*, 12(3):429–436, 1988.

[16] H. Kaplan, M. Lewenstein, N. Shafrir, and M. Sviridenko. Approximation algorithms for asymmetric TSP by decomposing directed regular multigraphs. *J. ACM*, 52(4):602–626, 2005.

[17] J. Kleinberg and D. P. Williamson. Unpublished note, 1998.

[18] F. Lam and A. Newman. Traveling salesman path problems. *Math. Program.*, 113(1):39–59, 2008.

[19] E. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. Shmoys, editors. *The Traveling Salesman Problem: A guided tour of combinatorial optimization*. John Wiley & Sons Ltd., 1985.

[20] I. Mendez-Diaz, P. Zabala, and A. Lucena. A new formulation for the traveling deliveryman problem. *Discrete Applied Mathematics*, 156(17):3223–3237, 2008.

[21] E. Minieka. The deliveryman problem on a tree network. *Ann. Oper. Res.*, 18:261–266, 1989.

[22] V. Nagarajan and R. Ravi. Poly-logarithmic approximation algorithms for directed vehicle routing problems. In *Proc. 10th APPROX*, pages 257–270, 2007.

[23] V. Nagarajan and R. Ravi. The directed minimum latency problem. In *Proc. 11th APPROX*, pages 193–206, 2008.

[24] C. H. Papadimitriou and S. Vempala. On the approximability of the traveling salesman problem. *Combinatorica*, 26(1):101–120, 2006.

[25] D. Shmoys and D. P. Williamson. Analyzing the Held-Karp TSP bound: a monotonicity property with application. *Inf. Process. Lett.*, 35(6):281–285, 1990.

[26] D. P. Williamson. Analysis of the Held-Karp heuristic for the traveling salesman problem. M.S. Thesis, MIT, 1990.

[27] L. A. Wolsey. Heuristic analysis, linear programming and branch and bound. *Mathematical Programming Study*, 13:121–134, 1980.