A Constant-Factor Approximation Algorithm for the Asymmetric Traveling Salesman Problem[*]

Ola Svensson[†]
ola.svensson@epfl.ch

Jakub Tarnawski[‡]
jakub.tarnawski@gmail.com

László A. Végh[§]
l.vegh@lse.ac.uk

September 15, 2020

Abstract

We give a constant-factor approximation algorithm for the asymmetric traveling salesman problem (ATSP). Our approximation guarantee is analyzed with respect to the standard LP relaxation, and thus our result confirms the conjectured constant integrality gap of that relaxation.

The main idea of our approach is a reduction to Subtour Partition Cover, an easier problem obtained by significantly relaxing the general connectivity requirements into local connectivity conditions. We first show that any algorithm for Subtour Partition Cover can be turned into an algorithm for ATSP while only losing a small constant factor in the performance guarantee. Next, we present a reduction from general ATSP instances to structured instances, on which we then solve Subtour Partition Cover, yielding our constant-factor approximation algorithm for ATSP.

Keywords: approximation algorithms, asymmetric traveling salesman problem, combinatorial optimization, linear programming

[*]This paper is the joint journal version of the conference publications [Sve15] and [STV18a].

[†]École Polytechnique Fédérale de Lausanne. Supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement 335288–OptApprox) and the Swiss National Science Foundation project 200021-184656 "Randomness in Problem Instances and Randomized Algorithms".

[‡]École Polytechnique Fédérale de Lausanne. Supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement 335288–OptApprox).

[§]London School of Economics and Political Science. Supported by EPSRC First Grant EP/M02797X/1 and the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement 757481–ScaleOpt).

Contents

1 Introduction

The traveling salesman problem—to find the shortest tour visiting n given cities—is one of the best-known NP-hard optimization problems.

Without any assumptions on the distances, a simple reduction from the problem of deciding whether a graph is Hamiltonian shows that it is NP-hard to approximate the shortest tour to within any factor. Therefore it is common to relax the problem by allowing the tour to visit cities more than once. This is equivalent to assuming that the distances satisfy the triangle inequality: the distance from city i to k is no larger than the distance from i to j plus the distance from j to k. All results mentioned and proved in this paper refer to this setting.

If we also assume the distances to be symmetric, then Christofides' classic algorithm from 1976 [Chr76], also discovered independently by Serdyukov [Ser78, vBS20], is guaranteed to find a tour of length at most $3/2$ times the optimum. Improving this approximation guarantee is a notorious open question in approximation algorithms. There has been a flurry of recent progress in the special case of unweighted graphs [GSS11, MS16, Muc12, SV14]. However, even though the standard linear programming (LP) relaxation is conjectured to approximate the optimum within a factor of $4/3$, it remains an elusive problem to improve upon the Christofides–Serdyukov algorithm.

If we do not restrict ourselves to symmetric distances (undirected graphs), we obtain the more general *asymmetric* traveling salesman problem (ATSP). Compared to the symmetric setting, the gap in our understanding is much larger, and the current algorithmic techniques have failed to give *any* constant approximation guarantee. This is intriguing especially since the standard LP relaxation, also known as the *Held–Karp lower bound*, is conjectured to approximate the optimum to within a small constant. In fact, it is only known that its integrality gap[1] is at least 2 [CGK06]. We also note that the best known inapproximability bound for ATSP is 75/74 [KLS13].

One can easily show that a multiset of edges forms a feasible solution to ATSP if and only if it is connected and Eulerian. The first approximation algorithm for ATSP was given by Frieze, Galbiati and Maffioli [FGM82], achieving an approximation guarantee of $\log_2(n)$. Their elegant "repeated cycle cover" approach maintains an Eulerian edge multiset throughout, but it initially relaxes connectivity, and enforces connectivity gradually. This approach was refined in several papers [Blä08, KLSS05, FS07], but there was no *superconstant* improvement in the approximation guarantee until the more recent 2010 $O(\log n \,/\, \log\log n)$-approximation algorithm by Asadpour et al. [AGM+17]. They introduced a new and influential approach to ATSP based on relaxing the Eulerian degree constraints but maintaining connectivity throughout. The key idea is establishing a connection to the graph-theoretic concept of thin spanning trees. This has further led to improved algorithms for special cases of ATSP, such as graphs of bounded genus [GS11]. Moreover, Anari and Oveis Gharan recently exploited this connection to significantly improve the best known upper bound on the integrality gap of the standard LP relaxation to $O(\text{poly}\log\log n)$ [AG15]. This implies an efficient

[1]Recall that the integrality gap is defined as the maximum ratio between the optimum values of the exact (integer) formulation and of its relaxation.

algorithm for estimating the optimal value of a tour within a factor $O(\text{poly} \log \log n)$ but, as their arguments are non-constructive, no approximation algorithm for finding a tour of matching guarantee.

In this paper, we follow an approach more akin to the one by Frieze, Galbiati and Maffioli [FGM82]. Namely, we maintain the Eulerian degree constraints but relax the connectivity requirements, by introducing a problem called *Subset Partition Cover*. We prove our main theorem using this auxiliary problem.

Theorem 1.1. *There is a polynomial-time algorithm for ATSP that returns a tour of value at most* 506 *times the Held–Karp lower bound.*

This paper is a joint version of the two conference publications [Sve15] and [STV18a]. The paper [Sve15] introduced the relaxed problem *Local-Connectivity ATSP* and used it to obtain a constant-factor approximation algorithm for unweighted digraphs, and more generally for *node-weighted graphs*, i.e., graphs whose weight function can be written as $w(u, v) = f(u) + f(v)$ for some $f : V \to \mathbb{R}_+$.[2] The Subset Partition Cover problem described in this paper is a more refined version of Local-Connectivity ATSP; the terminology has been changed in order to emphasize the differences between the problems.

The publication [STV18a] proved Theorem 1.1 with an approximation ratio of 5500. The significant improvement in the guarantee presented here is due to using the more refined Subset Partition Cover problem along with some more efficient reductions.

We remark that we can obtain a tighter upper bound of 319 for the integrality gap of the Held–Karp relaxation, and that our results also imply a constant-factor approximation algorithm for the Asymmetric Traveling Salesman *Path* Problem via black-box reductions, given by Feige and Singh for the approximation guarantee [FS07] and recently by Köhne, Traub and Vygen [KTV19] for the integrality gap—see Section 11.

In a recent development, Traub and Vygen [TV20] have improved the approximation guarantee to $22 + \varepsilon$, and the integrality gap to 22. Their results are attained by improving and simplifying the techniques in this paper. They use a simpler chain of reductions by avoiding the use of irreducible instances inherent in our argument, as well as refining other parts of the reduction. See the conclusions (Section 12) for further discussion and open problems.

1.1 Brief overview of approach and outline of paper

It will be convenient to define ATSP in terms of its *graphic formulation*:

Definition 1.2. The input for ATSP is a pair (G, w), where G is a strongly connected directed graph (digraph) and w is a nonnegative weight function defined on the edges. The objective is to find a closed walk of minimum weight that visits every vertex at least once.

[2] In [Sve15], the definition is slightly different: $w(u, v) = g(u)$ for every $(u, v) \in E$ for a function $g : V \to \mathbb{R}_+$. The two definitions are equivalent: by assigning $g(u) = 2f(u)$, the weight of any tour is equal for the weights $w(u, v) = g(u)$ and for the weights $w(u, v) = f(u) + f(v)$.

Another standard definition of ATSP asks for a minimum-weight Hamiltonian cycle, that is, a closed walk that visits every vertex exactly once. If the weight function satisifies the triangle inequality, these two forms are equivalent: a tour that visits every vertex at least once can be shortcut to a tour visiting every vertex exactly once, without increasing the weight of the tour. Throughout the paper, we use the graphic formulation as in Definition 1.2.

Without loss of generality, one could assume that G is a complete digraph. However, for our reductions, it will be important that G may not be complete. We also remark that a closed walk that visits every vertex at least once is equivalent to an Eulerian multiset of edges that connects the graph. (An edge set of a digraph is Eulerian if the in-degree of each vertex equals its out-degree.)

The first main step of our argument is introducing the problem *Subtour Partition Cover* in Section 3. The main technical contribution of Part I is a reduction which shows that if one can solve Subtour Partition Cover on some class of graphs, then one can obtain a constant-factor approximation for ATSP on that class of graphs (cf. Theorem 5.1). In [Sve15], a constant-factor approximation algorithm was obtained for ATSP on node-weighted graphs by solving Subtour Partition Cover (more precisely, the earlier variant Local-Connectivity ATSP) for this class. Subsequently, [STV18b] solved Local-Connectivity ATSP for graphs with two different edge-weights. This required a rather difficult technical argument, and it appears to be very challenging to solve Subtour Partition Cover directly on general graphs.

In this paper, we follow a different approach. Before applying the reduction (Theorem 5.1), we remain in the realm of ATSP and use a series of natural reductions to gradually simplify the structure of instances that we are dealing with. The first of these reductions (in Section 2.2) crucially uses the laminar structure arising from the Held–Karp relaxation and its dual linear program (see Theorems 2.2 and 2.4). All further reductions are described in Part II. The most structured instances, for which we apply the reduction to Subtour Partition Cover and on which we then solve Subtour Partition Cover in Part III, are called *vertebrate pairs*.

The outline of the paper is as follows. In Section 2 we present preliminaries and introduce notation used throughout the paper. We introduce the standard Held–Karp relaxation in Section 2.1. Section 2.2 is devoted to our first reduction. There, we show that we can focus on *laminarly-weighted ATSP instances*: there is a laminar family $\mathcal{L}$ of vertex sets and a nonnegative vector $(y_S)_{S\in\mathcal{L}}$ such that any edge e has $w(e) = \sum_{S\in\mathcal{L}:\, e\in\delta(S)} y_S$. See the left part of Figure 1 for an example. Note that the special case when the laminar family consists only of singletons roughly corresponds to node-weighted instances. We call laminarly-weighted ATSP instances where $\mathcal{L} \subseteq \{\{v\} : v \in V\}$ *singleton instances*.[3]

Next, in Part I we define the Subtour Partition Cover problem, where the connectivity requirements are relaxed in comparison to ATSP, and reduce the task of solving ATSP (with a constant-factor approximation) to that problem. We also solve Subtour Partition Cover on singleton instances (Theorem 4.1), thus illustrating the power of the reduction as well as developing a tool necessary later in Part II.

[3]Every singleton instance is a node-weighted instance, but not vice versa; see the discussion following Theorem 2.5.

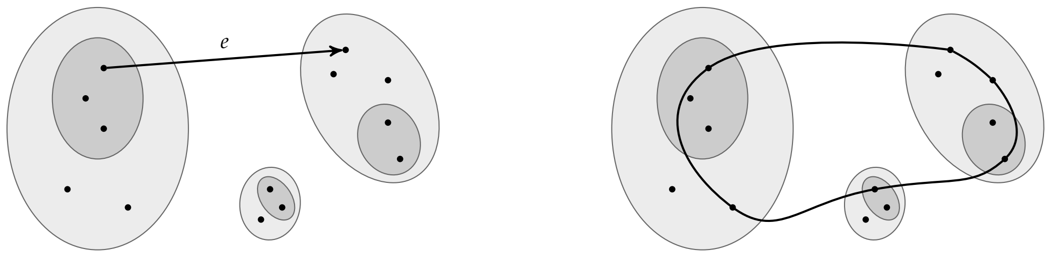


Figure 1: On the left we give an example of a laminarly-weighted ATSP instance. The sets of the laminar family are shown in gray. We depict a single edge e that crosses three sets in the laminar family, say S_1, S_2, S_3, and so $w(e) = y_{S_1} + y_{S_2} + y_{S_3}$. On the right, we give an example of a vertebrate pair. Notice that the backbone (depicted as the cycle) crosses all non-singleton sets of the laminar family, though it may not visit all the vertices.

In Part II we turn our attention back to ATSP and, starting from laminarly-weighted instances, show that we can obtain very structured ATSP instances called vertebrate pairs by only increasing the approximation guarantee by a constant factor. A vertebrate pair consists of a laminar instance and a subtour B, called the backbone, that crosses every non-singleton set of $\mathcal{L}$. An example is depicted on the right part of Figure 1.

Finally, in Part III we give an algorithm for Subtour Partition Cover on such instances. By the aforementioned reductions, solving Subtour Partition Cover for vertebrate pairs is sufficient for obtaining a constant-factor approximation algorithm for general ATSP. We combine all the ingredients and calculate the obtained ratio in Section 11. We discuss future research directions in Section 12.

2 Preliminaries

For a directed graph G, we let $V(G)$ and $E(G)$ denote the set of vertices and edges, respectively. All graphs in the paper will be directed; we will use the term 'graph' to refer to a directed graph. We use simply V and E whenever the graph is clear from the context. By an edge set $F \subseteq E$, we always mean an *edge multiset*: the same edge can be present in multiple copies. We will refer to the union of two edge (multi)sets as a multiset (adding up the multiplicities of every edge).

For vertex sets $S, T \subseteq V$ we let $\delta(S,T) = \{(u,v) \in E : u \in S \setminus T, v \in T \setminus S\}$. For a set $S \subseteq V$ we let $\delta^+(S) = \delta(S, V \setminus S)$ denote the set of outgoing edges, and we let $\delta^-(S) = \delta(V \setminus S, S)$ denote the set of incoming edges. Further, let $\delta(S) = \delta^-(S) \cup \delta^+(S)$ be all boundary edges and $E(S) = \{(u,v) \in E : u, v \in S\}$ be interior edges of a vertex set S. For a vertex $v \in V$ we define $\delta^+(v) = \delta^+(\{v\})$ and $\delta^-(v) = \delta^-(\{v\})$. For an edge (multi)set $F \subseteq E$, we use $\delta_F(S,T) = \delta(S,T) \cap F$, $\delta^+_F(S) = \delta^+(S) \cap F$, etc. We let $V(F)$ denote the set of vertices incident to at least one edge in F, and $\mathbb{1}_F$ denote the indicator vector of F, which has a coordinate for each edge e with value equal to the multiplicity of e in F.

For a vertex set $U \subsetneq V$, we let $G[U] = (U, E(U))$ denote the subgraph induced by U. That is, $G[U]$ is the subgraph of G whose vertex set is U and whose edge set consists of all edges in $E(G)$ with both endpoints in U. We also let G/U denote the graph obtained by contracting the vertex set U, i.e., by replacing the vertices in U by a single new vertex u and redirecting every edge with one endpoint in U to point from/to the new vertex u. This may create parallel edges in G/U. We keep all parallel copies; thus, every edge in G/U will have a unique preimage in G.

For a set $S \subsetneq V$ we let S_{in} and S_{out} be those vertices of S that have an incoming edge from outside of S and those that have an outgoing edge to outside of S, respectively. That is,

$$S_{\text{in}} = \{v \in S : \delta^-(S) \cap \delta^-(v) \neq \emptyset\} \quad \text{and} \quad S_{\text{out}} = \{v \in S : \delta^+(S) \cap \delta^+(v) \neq \emptyset\}.$$

We let $\mathbb{R}_+$ denote the set of nonnegative real numbers. The *support* of a function/vector $f : X \to \mathbb{R}_+$ is the subset $\{x \in X : f(x) > 0\}$. For a subset $Y \subseteq X$, we also use $f(Y) = \sum_{x \in Y} f(x)$.

When talking about graphs, we shall slightly abuse notation and sometimes write $w(G)$ instead of $w(E)$ and $f(G)$ instead of $f(V)$ when it is clear from the context that w and f are functions on the edges and vertices.

Finally, a closed walk will be called a subtour:

Definition 2.1. We call $F \subseteq E$ a *subtour* if F is Eulerian (we have $|\delta_F^+(v)| = |\delta_F^-(v)|$ for every $v \in V$) and the graph $(V(F), F)$ is connected. By convention, $F = \emptyset$ is a subtour. A *tour* is a subtour F with $V(F) = V$.

Therefore a subtour is an Eulerian multiset of edges that form a single connected component (or an empty set), and a tour is an Eulerian multiset of edges that connects the graph.

For any Eulerian multiset F of edges, we refer to the connected components of $(V(F), F)$ as *subtours in F*. We often refer to F as a collection of subtours. Note that if T is a subtour in F, then $T \neq \emptyset$.

We say that a subtour T intersects another subtour T' if we have $V(T) \cap V(T') \neq \emptyset$.

2.1 Held–Karp Relaxation

Given an edge-weighted digraph (G, w), the Held–Karp relaxation has a variable $x(e) \geqslant 0$ for every edge $e \in E$. The intended solution is that $x(e)$ should equal the number of times e is used in the solution. The linear programming relaxation LP(G, w) is now defined as follows:

$$\begin{array}{llll} \text{minimize} & \displaystyle\sum_{e \in E} w(e)x(e) & & \\ \text{subject to} & x(\delta^+(v)) = x(\delta^-(v)) & \text{for } v \in V, & \qquad (\text{LP}(G,w)) \\ & x(\delta(S)) \geqslant 2 & \text{for } \emptyset \neq S \subsetneq V, & \\ & x \geqslant 0. & & \end{array}$$

The optimum value of this LP is called the *Held–Karp lower bound*. The first set of constraints says that the in-degree should equal the out-degree for each vertex, i.e., the solution should be Eulerian. We call a non-negative vector x satisfying these constraints a *circulation*. The second set of constraints enforces that the solution is connected. These are sometimes referred to as subtour elimination constraints. Notice that the Eulerian property implies $x(\delta^-(S)) = x(\delta^+(S))$ for every set $S \subseteq V$, and therefore these constraints are equivalent to $x(\delta^+(S)) \geqslant 1$ for all $\emptyset \neq S \subsetneq V$, which appear more frequently in the literature. We use the above formulation as it enables some simplifications in the presentation.

We say that a set $S \subseteq V$ is *tight* with respect to a solution x to $\mathrm{LP}(G, w)$ if $x(\delta(S)) = 2$, that is, $x(\delta^-(S)) = x(\delta^+(S)) = 1$.

Let us now formulate the dual linear program $\mathrm{DUAL}(G, w)$. We associate variables $(\alpha_v)_{v\in V}$ and $(y_S)_{\emptyset\neq S\subset V}$ with the first and second set of constraints of $\mathrm{LP}(G, w)$, respectively.

$$\begin{array}{lll}
\text{maximize} & \displaystyle\sum_{\emptyset\neq S\subsetneq V} 2 \cdot y_S & \\
\text{subject to} & \displaystyle\sum_{S:(u,v)\in\delta(S)} y_S + \alpha_u - \alpha_v \leqslant w(u, v) & \text{for } (u, v) \in E, \\
& y \geqslant 0. &
\end{array} \qquad (\mathrm{DUAL}(G, w))$$

For singleton sets $\{u\}$, we will also use the notation $y_u = y_{\{u\}}$.

The Held–Karp relaxation has exponentially many constraints, but it can be solved in polynomial time using the ellipsoid method with a separation oracle. Moreover, an optimal solution to $\mathrm{DUAL}(G, w)$ can also be found in polynomial time. We now briefly explain how the ellipsoid method can be applied. Let P be the feasible region. Given a point $x \in \mathbb{R}^E$, we can decide whether $x \in P$ by checking the Eulerian constraints and computing the minimum cut value. If x is not feasible, this yields a separating inequality that is one of the inequalities of the system $\mathrm{LP}(G, w)$. We have a *well-described polyhedron* as in Definition 6.2.2 in [GLS12]: P is an m-dimensional polyhedron defined by inequalities of encoding length at most $2m + 3$. Theorems 6.4.9 and 6.5.14-15 in [GLS12] show that optimal primal and dual solutions can be found using a polynomial number of oracle calls. Since the encoding length is $O(m)$, and separation can be done in strongly polynomial time, the overall running time is strongly polynomial. We note that an optimal primal solution can also be found by formulating an equivalent compact (polynomial-size) linear program [Art82, Car96]; but such a reduction would not directly yield a dual optimal solution.

The ellipsoid method provides an optimal solution to $\mathrm{DUAL}(G, w)$ with a polynomial-size support. We will need a dual optimal solution with a "nice" support, as stated next. Recall that a family $\mathcal{L} \subseteq 2^V$ of vertex subsets is *laminar* if for any $A, B \in \mathcal{L}$ we have either $A \subseteq B$ or $B \subseteq A$ or $A \cap B = \emptyset$.

Lemma 2.2. *For every edge-weighted digraph (G, w) there exists an optimal solution (α, y) to* $\mathrm{DUAL}(G, w)$ *such that the support of y is a laminar family of vertex subsets. Moreover, such a solution can be computed in polynomial time.*

We note that the existence of a laminar solution was also used previously in [VY99]. For finding one in polynomial time we invoke a result by Karzanov [Kar96].

Proof. We start by showing the existence of a laminar optimal solution using a standard uncrossing argument (see e.g. [CFN85] for an early application of this technique to the Held–Karp relaxation of the symmetric traveling salesman problem). Select (α, y) to be an optimal solution to DUAL(G, w) minimizing $\sum_S |S| y_S$. That is, among all dual solutions that maximize the dual objective $2\sum_S y_S$, we select one that minimizes $\sum_S |S| y_S$. We claim that the support $\mathcal{L} = \{S : y_S > 0\}$ is a laminar family. Suppose not, i.e., that there are sets $A, B \in \mathcal{L}$ such that $A \cap B, A \setminus B, B \setminus A \neq \emptyset$. Then we can obtain a new dual solution $(\alpha, \hat{y})$, where $\hat{y}$ is defined, for $\varepsilon = \min(y_A, y_B) > 0$, as

$$\hat{y}_S = \begin{cases} y_S - \varepsilon & \text{if } S = A \text{ or } S = B, \\ y_S + \varepsilon & \text{if } S = A \setminus B \text{ or } S = B \setminus A, \\ y_S & \text{otherwise.} \end{cases}$$

That $(\alpha, \hat{y})$ remains a feasible solution follows since $\hat{y}$ remains non-negative (by the selection of ε) and since for any edge e we have

$$\mathbb{1}_{e \in \delta(A)} + \mathbb{1}_{e \in \delta(B)} \geqslant \mathbb{1}_{e \in \delta(A \setminus B)} + \mathbb{1}_{e \in \delta(B \setminus A)} .$$

Therefore $\sum_{S: e \in \delta(S)} \hat{y}_S \leqslant \sum_{S: e \in \delta(S)} y_S$ and so the constraint corresponding to edge e remains satisfied. Further, we clearly have $2\sum_S \hat{y}_S = 2\sum_S y_S$. In other words, $(\alpha, \hat{y})$ is an optimal dual solution. However,

$$\sum_S |S|(y_S - \hat{y}_S) = (|A| + |B| - |A \setminus B| - |B \setminus A|)\varepsilon > 0 ,$$

which contradicts that (α, y) was selected to be an optimal dual solution minimizing $\sum_S |S| y_S$. Therefore, there can be no such sets A and B in $\mathcal{L}$, and so it is a laminar family.

To find a laminar optimal solution in polynomial time, we start with an arbitrary dual optimal solution. As noted above, one can be computed in polynomial time. Now we apply the above uncrossing operation to obtain a laminar optimal solution. A result by Karzanov [Kar96, Theorem 2] shows that if we carefully select the sequence of pairs A, B to uncross, this can be performed in polynomial time (although for an arbitrary sequence, the number of uncrossing steps may not be polynomially bounded). □

2.2 Laminarly-Weighted ATSP and Singleton Instances

In this section we show that without loss of generality (i.e., without any loss in the approximation factor) we can focus on instances whose weights come from a sparse and highly structured family of sets. Below we define the crucial notion of laminarly-weighted instances that we will work with throughout the paper.

Definition 2.3. A tuple $\mathcal{I} = (G, \mathcal{L}, x, y)$ is called a *laminarly-weighted ATSP instance* if G is a strongly connected digraph, $\mathcal{L}$ is a laminar family of vertex subsets, x is a feasible solution to LP$(G, \mathbf{0})$, and $y : \mathcal{L} \to \mathbb{R}_+$. We further require that $x_e > 0$ for every $e \in E$ and that every set $S \in \mathcal{L}$ be tight with respect to x, i.e., that $x(\delta^+(S)) = x(\delta^-(S)) = 1$. We define the *induced weight function* $w_{\mathcal{I}} : E \to \mathbb{R}_+$ as

$$w_{\mathcal{I}}(e) = \sum_{S \in \mathcal{L}:\, e \in \delta(S)} y_S \qquad \text{for every } e \in E.$$

Here, **0** denotes the zero weight function.

Given an instance $\mathcal{I}$ as in the definition, the vectors x and y have the following important property. Define a dual solution $(\bar{\alpha}, \bar{y})$ by setting $\bar{\alpha}_u = 0$ for all $u \in V$, and $\bar{y}_S = y_S$ if $S \in \mathcal{L}$ and $\bar{y}_S = 0$ otherwise. Then complementary slackness implies that for the induced weight function $w_{\mathcal{I}}$, the vector x is an optimal solution to LP$(G, w_{\mathcal{I}})$ and $(\bar{\alpha}, \bar{y})$ is an optimal solution to DUAL$(G, w_{\mathcal{I}})$.

Our first main insight is that ATSP with arbitrary weights can be reduced to the laminarly-weighted ATSP problem.

Theorem 2.4. *Assume we have a polynomial-time algorithm that finds a solution of weight at most α times the Held–Karp lower bound for every laminarly-weighted ATSP instance. Then there is a polynomial-time algorithm for the general ATSP problem that finds a solution of weight at most α times the Held–Karp lower bound.*

Proof. Consider an arbitrary edge-weighted strongly connected digraph (G, w). Let x be an optimal solution to LP(G, w) and let (α, y) be an optimal solution to DUAL(G, w) as guaranteed by Lemma 2.2, that is, y has a laminar support $\mathcal{L}$. We now define a pair (G', w') as

$$V(G') = V(G)\,, \quad E(G') = \{e \in E(G) : x(e) > 0\}\,, \quad \text{and} \quad w'(u,v) = w(u,v) - \alpha_u + \alpha_v\,.$$

We claim that $\mathcal{I} = (G', \mathcal{L}, x, y)$ is a laminarly-weighted ATSP instance whose induced weight function $w_{\mathcal{I}}$ equals w'. To see this, recall that x is a primal optimal solution and that (α, y) is a dual optimal solution (for (G, w)). Therefore complementary slackness implies that every set in $\mathcal{L}$ is tight with respect to x and that for every edge $(u, v) \in E(G')$, the weight $w'(u,v) = w(u,v) - \alpha_u + \alpha_v$ equals the sum of y_S-values for the sets S crossed by (u, v). Finally, we have $x_e > 0$ for every $e \in E(G')$ by definition. So $\mathcal{I}$ satisfies all the properties of Definition 2.3, i.e., it is a laminarly-weighted instance.

We now argue that an α-approximate solution for $\mathcal{I}$ with respect to the Held–Karp relaxation LP(G', w') implies an α-approximate solution for the original instance (G, w) with respect to LP(G, w). To this end, we make the following observation:

Claim. For any circulation $x \in \mathbb{R}_+^{E(G')}$, we have $\sum_{e \in E(G')} w(e)x(e) = \sum_{e \in E(G')} w'(e)x(e)$.

Therefore the Held–Karp lower bound is the same for (G, w) and for (G', w'), and any solution for (G', w') is a solution of the same weight for (G, w). $\square$

In the rest of the paper we work exclusively with laminarly-weighted ATSP instances $\mathcal{I} = (G, \mathcal{L}, x, y)$. We will refer to them as simply *instances*.

Definition 2.5. We say that an instance $\mathcal{I} = (G, \mathcal{L}, x, y)$ is a *singleton instance* if all sets in $\mathcal{L}$ are singletons.

Such instances will play an important role in our algorithm. In particular, note that for singleton instances, the weight function $w_{\mathcal{I}}$ is induced by nodes ($w(u, v) = y_u + y_v$ for all $(u, v) \in E$) just like in a node-weighted instance. The difference between singleton and node-weighted instances is that singleton instances are those *laminarly-weighted* instances whose weight function is induced by nodes after having performed the reduction of Theorem 2.4. A node-weighted instance does not necessarily give rise to a singleton instance: for singleton instances we will also require that $x(\delta^+(v)) = x(\delta^-(v)) = 1$ for every node v with $y_v > 0$.

Recall that $w_{\mathcal{I}}(F)$ is the induced weight of an edge multiset $F \subseteq E$ in the instance $\mathcal{I}$. We will omit the subscript and use simply $w(F)$ whenever $\mathcal{I}$ is clear from the context.

Definition 2.6. For an instance $\mathcal{I} = (G, \mathcal{L}, x, y)$ and a set $S \subseteq V$ we define

$$\text{value}_{\mathcal{I}}(S) = 2 \cdot \sum_{R \in \mathcal{L}:\ R \subsetneq S} y_R$$

to be the fractional dual value associated with the sets strictly inside S.

Again, we will omit the subscript whenever clear from the context. We also use $\text{value}(\mathcal{I}) = \text{value}_{\mathcal{I}}(V)$; note that this equals the Held–Karp lower bound of the instance. Indeed, as noted above, y can be extended to an optimal dual solution to DUAL(G, w), and hence the optimum value for DUAL(G, w) equals $2 \cdot \sum_{S \in \mathcal{L}} y_S$, which is equal to the primal optimum value $\sum_{e \in E} w(e)x(e)$ for LP(G, w) by strong duality.

Part I

Reducing ATSP to Subtour Partition Cover

In this part we define the Subtour Partition Cover problem and reduce the task of solving ATSP to that problem. This reduction will be used to solve general instances in Part III. Here, we illustrate its power by giving a constant-factor approximation algorithm for singleton instances.

Let us begin with some intuition. It is illustrative to consider the following "naive" algorithm:

1. Select a random cycle cover C using the Held–Karp relaxation.

   It is well known that one can sample such a cycle cover C of expected weight equal to the optimal value value$(\mathcal{I})$ of the Held–Karp relaxation.

2. While there exist more than one component, add the lightest cycle (i.e., the cycle of smallest weight) that decreases the number of components.

It is clear that the above algorithm always returns a solution to ATSP: we start with an Eulerian graph, and the graph stays Eulerian during the execution of the while-loop, which does not terminate until the graph is connected. This gives a tour. However, what is its weight? First, as remarked above, we have that the expected weight of the cycle cover is value$(\mathcal{I})$. So if C contains $k = |C|$ cycles, we would expect that a cycle in C has weight value$(\mathcal{I})/k$ (at least on average). Moreover, the number of cycles added in Step 2 is at most $k - 1$ since each cycle decreases the number of components by at least one. Thus, if each cycle in Step 2 has weight at most the average weight value$(\mathcal{I})/k$ of a cycle in C, we obtain a 2-approximate tour of weight at most value$(\mathcal{I}) + \frac{k-1}{k}\,$value$(\mathcal{I}) \leqslant 2\,$value$(\mathcal{I})$.

Unfortunately, it seems hard to find a cycle cover C such that we can always connect it with light cycles. Instead, what we can do is to first select a cycle cover C, then add light cycles that decrease the number of components as long as possible. When there are no more light cycles to add, the vertices are partitioned into connected components $V_1, \dots, V_k$. In order to make progress from this point, we would like to find a "light" Eulerian set F of edges that crosses the cuts $\{(V_i, \bar{V}_i) \mid i = 1, 2, \dots, k\}$. We could then hope to add F to our solution and continue from there. It turns out that the meaning of "light" in this context is crucial. For our arguments to work, we need that F is selected so that the edges in each component have weight at most α times what the linear programming solution "pays" for the vertices in that component. This is the intuition behind the definitions of Subtour Partition Cover (formerly Local-Connectivity ATSP) and "light" algorithms for that problem. We also need to be very careful as to how we add edges from light cycles and how to use the light algorithm for Subtour Partition Cover. In Section 5, our algorithm will iteratively solve Subtour Partition Cover and, in each iteration, it will add a carefully chosen subset of the found edges, together with light cycles.

We remark that in Subtour Partition Cover we have relaxed the global connectivity properties of ATSP into local connectivity conditions that only say that we need to find an Eulerian set of edges that crosses at most $n = |V|$ cuts defined by a partitioning of the vertices. In spite of that, we are able to leverage the intuition above to obtain our main technical result of this part (Theorem 5.1).

Its proof is based on generalizing and, as alluded to above, deviating from the above intuition in several ways. First, we start with a carefully chosen collection of subtours which generalizes the role of the cycle cover C in Step 1 above. Second, both the iterative use of the light algorithm for Subtour Partition Cover and the way we add light cycles are done in a careful and dependent manner so as to be able to bound the total weight of the returned solution.

3 Subtour Partition Cover

In this section we define the Subtour Partition Cover problem, which is obtained from ATSP by relaxing the connectivity requirements. Consider a laminarly-weighted instance $\mathcal{I} = (G, \mathcal{L}, x, y)$. For notational convenience we extend the vector y to all singletons so that $y_v = 0$ if $\{v\} \notin \mathcal{L}$. Let $\mathrm{lb}_{\mathcal{I}} : V \to \mathbb{R}$ be the *lower bound function* defined by $\mathrm{lb}_{\mathcal{I}}(v) = 2y_v$. We simplify notation and write lb instead of $\mathrm{lb}_{\mathcal{I}}$ if $\mathcal{I}$ is clear from the context. Note that $\mathrm{lb}(V)$ is at most the Held–Karp lower bound value($\mathcal{I}$), with equality only for singleton instances. For an edge set F, we use the simplified notation $\mathrm{lb}(F) = \mathrm{lb}(V(F))$ to denote the total lower bound of the vertices incident to F.

Perhaps the main difficulty of ATSP is to satisfy the connectivity requirement, i.e., to select an Eulerian subset F of edges that satisfies all subtour elimination constraints. In *Subtour Partition Cover*, this condition is relaxed and we only require that the subtour elimination constraints be satisfied for some disjoint sets.

Subtour Partition Cover

Given: An instance $\mathcal{I} = (G, \mathcal{L}, x, y)$, a subtour B in G, and a partition $(V_1, V_2, \ldots, V_k)$ of $V \setminus V(B)$ such that the graph induced by V_i is strongly connected for $i = 1, \ldots, k$.

Find: A collection F of subtours of E such that $|\delta^+_F(V_i)| \geqslant 1$ for $i = 1, 2, \ldots, k$.

An example of an instance and a solution to Subtour Partition Cover can be found in Figure 3 on page 19. In that figure, the vertices $V \setminus V(B)$ are partitioned into six sets depicted in gray (the right-most set is $V(B)$) and the blue (solid) subtours show a solution. Note that the subtours are not required to connect the whole graph; they are only required to cross the boundaries defined by the partitioning of $V \setminus V(B)$.

Definition 3.1. We say that an algorithm for Subtour Partition Cover is (α, β)-*light* for an instance $\mathcal{I}$ and subtour B if, for any input partition of strongly connected subsets, the collection F of subtours satisfies

– $w_{\mathcal{I}}(T) \leqslant \alpha \operatorname{lb}(T)$ for every subtour T in F with $V(T) \cap V(B) = \emptyset$, and

– $w_{\mathcal{I}}(F_B) \leqslant \beta$, where $F_B \subseteq F$ is the collection of subtours in F that intersect B.[4]

We use the (α, β)-light terminology to avoid any ambiguities with the concept of approximation algorithms.

If we let c be the scaling factor such that $c \operatorname{lb}(V) = \text{value}(\mathcal{I})$, then an α-approximation algorithm for ATSP with respect to the Held–Karp relaxation is trivially an $(\alpha \cdot c, 0)$-light algorithm for Subtour Partition Cover with $B = \emptyset$: output the same tour F as the algorithm for ATSP. However, Subtour Partition Cover seems like a significantly easier problem than ATSP, as the set of subtours F only needs to cross k cuts formed by a partitioning of the vertices $V \setminus V(B)$. We substantiate this intuition by proving, in Section 4, that there exists a simple $(2, 0)$-light algorithm for Subtour Partition Cover on singleton instances with $B = \emptyset$. Perhaps more surprisingly, in Section 5 we show that an (α, β)-light algorithm for Subtour Partition Cover for an instance $\mathcal{I}$ with subtour B can be turned into an approximation algorithm for ATSP that always returns a tour of cost at most $(9 + \varepsilon)\alpha \operatorname{lb}(V \setminus V(B)) + \beta + w(B)$ for any $\varepsilon > 0$.

The main difference between the Subtour Partition Cover problem and the Local-Connectivity ATSP problem introduced in [Sve15] is the introduction of the subtour B and the more general definition of lightness. While this flexibility is unnecessary for singleton instances with $B = \emptyset$ (which are closely related to the node-weighted instances considered in that paper), it will be useful in the general case: in Section 10 we give an algorithm for Subtour Partition Cover that, in turn, implies the constant-factor approximation algorithm for general instances.

Remark 3.2. Our generic reduction from ATSP to Subtour Partition Cover (Theorem 5.1) is robust with respect to the definition of lb and there are many possibilities to define such a lower bound. Another natural example is $\operatorname{lb}(v) = \sum_{e \in \delta^+(v)} x_e w(e)$. In [Sve15], Local-Connectivity ATSP was defined with this lb function, and with $B = \emptyset$; in this case we can set $\beta = 0$. In fact, in order to get a constant bound on the integrality gap

[4]Recall that we say that a subtour T intersects another subtour B if they visit a common vertex, i.e., $V(T) \cap V(B) \neq \emptyset$. Hence, $F_B = \{T \text{ subtour in F} : V(T) \cap V(B) \neq \emptyset\}$.

of the Held–Karp relaxation, our results say that it is enough to find an $(O(1), 0)$-light algorithm for Subtour Partition Cover with respect to some nonnegative lb that only needs to satisfy that $\mathrm{lb}(V)$ is at most the value $\mathrm{value}(\mathcal{I})$ of the optimal solution to the LP. Even more generally, if $\mathrm{lb}(V)$ is at most the value of an optimal tour (rather than the LP value) then our methods would give a similar approximation guarantee (but not with respect to the Held–Karp relaxation).

A variant of Subtour Partition Cover was used in [STV18b] to obtain a constant factor approximation guarantee for ATSP with two different edge weights; a key idea of that paper is the careful choice of the lb function.

4 Subtour Partition Cover for Singleton Instances

We give a simple $(2, 0)$-light algorithm for Subtour Partition Cover for the special case of singleton instances, that is, when $\mathcal{L}$ is a singleton family, and for $B = \emptyset$.

The proof is based on finding an integral circulation that sends flow across the cuts $\{(V_i, \bar{V}_i) : i = 1, 2, \ldots, k\}$ and, in addition, satisfies that the outgoing flow of each vertex $v \in V$ with $y_v > 0$ is at most 2, which in turn, by the assumptions on the instance, implies a $(2, 0)$-light algorithm.

Theorem 4.1. *There exists a polynomial-time algorithm for Subtour Partition Cover that is* $(2, 0)$-*light for singleton instances with* $B = \emptyset$.

Proof. Let $\mathcal{I} = (G, \mathcal{L}, x, y), B, (V_1, V_2, \ldots, V_k)$ be an instance of Subtour Partition Cover where $\mathcal{I}$ is a singleton instance and $B = \emptyset$. Let also $w = w_{\mathcal{I}}$ denote the induced weight function. We prove the theorem by giving a polynomial-time algorithm that finds a collection F of subtours satisfying

$$|\delta^+_F(V_i)| \geqslant 1 \text{ for } i = 1, \ldots, k \quad \text{and} \quad |\delta^+_F(v)| \leqslant 2 \text{ for } v \in V \text{ with } y_v > 0. \tag{4.1}$$

The first condition means that F crosses every cut $(V_i, \bar{V}_i)$, thus the algorithm indeed solves the Subtour Partition Cover problem. We show that the second condition implies $(2, 0)$-lightness. Since $B = \emptyset$, we need to show that $w(T) \leqslant 2\,\mathrm{lb}(T)$ for every subtour T in F. Since $\mathcal{I}$ is a singleton instance, $w(u, v) = y_u + y_v$ for all $(u, v) \in E$ (recall the convention $y_u = 0$ if $\{u\} \notin \mathcal{L}$). Consider now any subtour T in F. We have

$$w(T) = \sum_{e \in T} w(e) = \sum_{v \in V(T)} |\delta_F(v)| y_v = 2 \sum_{v \in V(T)} |\delta^+_F(v)| y_v \leqslant 4 \sum_{v \in V(T)} y_v = 2\,\mathrm{lb}(T).$$

We proceed by describing a polynomial-time algorithm for finding an Eulerian set F satisfying (4.1). The set F will be obtained by rounding the circulation x to integrality while maintaining that it crosses each cut $(V_i, \bar{V}_i)$. For each cut $(V_i, \bar{V}_i)$ we introduce a new auxiliary vertex a_i to represent it. In lieu of requiring a flow of at least 1 through V_i, we will require such a flow through a_i. To show that such a (fractional) circulation exists, we modify x by redirecting an arbitrary flow of value 1 that passes through V_i to instead pass through a_i.

First, we transform G into a new graph G' and x into a new circulation x' by performing the following for each $i = 1, \ldots, k$ (see also Figure 2):

- Select a subset of incoming edges $X_i^- \subseteq \delta^-(V_i)$ with $x(X_i^-) = 1$. This is possible since $x(\delta^-(V_i)) \geqslant 1$.[5]
- Consider a cycle decomposition of x and follow the incoming edges in X_i^- in the decomposition. Select $X_i^+ \subseteq \delta^+(V_i)$ to be the set of outgoing edges on which these cycles first leave V_i after entering on an edge in X_i^-, such that $x(X_i^+) = 1$.[6] We define a flow x_i to be the x-flow on these cycle segments connecting the heads of edges in X_i^- and the tails of edges in X_i^+.
- We introduce a new auxiliary vertex a_i and redirect all edges in X_i^- to point to a_i, and those in X_i^+ to point from a_i. We subtract the flow x_i from x.

Note that x' is a circulation, and that it satisfies the following conditions:

- $x'(\delta^+(v)) \leqslant 1$ for all $v \in V$ with $y_v > 0$,
- $x'(\delta^+(a_i)) = 1$ for all $i = 1, \dots, k$.

Here, the first condition holds since all sets in $\mathcal{L}$ are tight and thus $x(\delta^+(v)) = 1$ whenever $y_v > 0$; the rest is by construction. As the vertex-degree bounds are integral, we can also, in polynomial time, find an *integral* circulation z' that satisfies these two conditions (see e.g. Chapter 11 in [Sch03]).

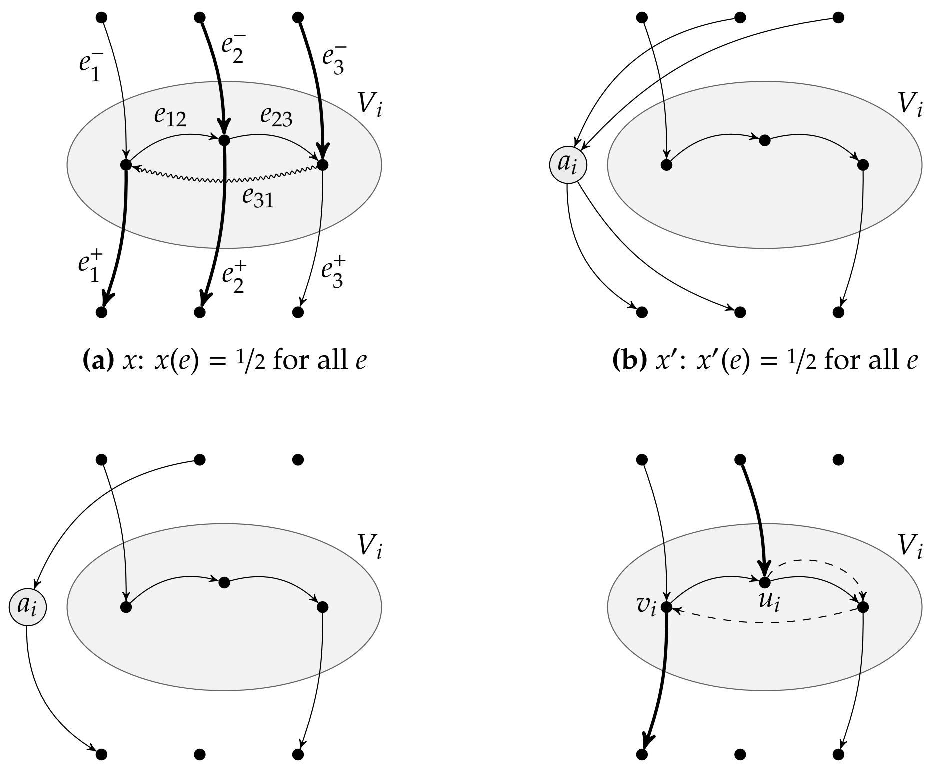
Next, we map z' from G' to a flow z in G in the natural way: by reversing the redirection of the edges incident to the auxiliary vertices a_i while retaining their flow. Now, the flow z so obtained may not be a circulation. Specifically, since the in- and out-degree of a_i were exactly 1 in z', in each component V_i there is a pair of vertices u_i, v_i that are the head and tail, respectively, of the mapped-back edges adjacent to a_i. These are the only vertices whose in-degree in z may differ from their out-degree. (They differ unless $u_i = v_i$.) To repair this, for each $i = 1, \dots, k$ we route a path P_i from u_i to v_i in V_i; this is always possible as we assumed that V_i is strongly connected (by the definition of Subtour Partition Cover).

We obtain our final solution F from z by taking every edge $e \in E$ with multiplicity z_e and adding the paths P_i. Note that F is Eulerian, i.e, a collection of subtours. To see that F satisfies (4.1), note that F crosses every cut $(V_i, \bar{V}_i)$ (since the edges redirected from a_i are boundary edges of V_i). Moreover, by the first property above and the fact that paths P_i are vertex-disjoint (being inside disjoint subsets V_i), we have $|\delta_F^+(v)| \leqslant 2$ for each $v \in V$ with $y_v > 0$. This concludes the proof of Theorem 4.1.

□

[5]To obtain exactly 1, we might need to break an edge up into two copies, dividing its x-value between them appropriately, and include one copy in X_i^- but not the other; we omit this for simplicity of notation, and assume there is such an edge set with exactly $x(X_i^-) = 1$.

[6]Again, to obtain exactly 1, we proceed as in the above footnote.



(a) x: $x(e) = 1/2$ for all e **(b)** x': $x'(e) = 1/2$ for all e

(c) z' (integral) **(d)** F

Figure 2: A depiction of the proof of Theorem 4.1. The neighborhood of a component V_i is shown.
(a) shows x, with $x(e) = 1/2$ on every shown edge. We select $X_i^- = \{e_2^-, e_3^-\}$ (thick incoming edges). Suppose that the cycle decomposition of x has a cycle containing e_1^-, e_{12}, e_{23}, e_3^+, a cycle containing e_2^-, e_2^+, and a cycle containing e_3^-, e_{31}, e_1^+. Thus we have $X_i^+ = \{e_1^+, e_2^+\}$ (thick outgoing edges), and the flow x_i (wiggly) puts value $1/2$ on e_{31}.
(b) shows x', with $x'(e) = 1/2$ on every shown edge. We redirect e_2^-, e_3^- to point to a_i and e_1^+, e_2^+ to point from a_i. We also subtract x_i, removing e_{31}.
(c) shows z', which is integral. Note that $z'(\delta^-(a_i)) = z'(\delta^+(a_i)) = 1$.
(d) shows the final solution F. The thick edges, which are redirected from the edges incident to a_i in z', guarantee that F crosses V_i. The path P_i is dashed.

5 From Local to Global Connectivity

In this section, we reduce the task of approximating ATSP to that of solving Subtour Partition Cover. To simplify the notation, for a subtour B, we let

$$\mathrm{lb}_{\mathcal{I}}(\bar{B}) = \mathrm{lb}_{\mathcal{I}}(V \setminus V(B)) = 2 \sum_{v \in V \setminus V(B)} y_v. \tag{5.1}$$

The main theorem can be stated as follows.

Theorem 5.1. *Let $\mathcal{A}$ be an algorithm for Subtour Partition Cover. For any instance $\mathcal{I} = (G, \mathcal{L}, x, y)$ and subtour B, if $\mathcal{A}$ is (α, β)-light for $\mathcal{I}$ and B, then there exists a tour of G of weight at most $5\alpha\,\mathrm{lb}_{\mathcal{I}}(\bar{B}) + \beta + w_{\mathcal{I}}(B)$. Moreover, for any $\varepsilon > 0$, a tour of weight at most $9(1+\varepsilon)\alpha\,\mathrm{lb}_{\mathcal{I}}(\bar{B}) + \beta + w_{\mathcal{I}}(B)$ can be found in time polynomial in the number $n = |V|$ of vertices, in $1/\varepsilon$, and in the running time of $\mathcal{A}$.*

Using Theorem 4.1, we immediately obtain a constant-factor approximation for ATSP on singleton instances.

Corollary 5.2. *For any $\varepsilon > 0$, there exists a polynomial-time $(18+\varepsilon)$-approximation algorithm for ATSP on singleton instances.*

In the sequel, we will use $\alpha_S = 18 + \varepsilon$ for the approximation ratio for ATSP on singleton instances to make the dependence on this factor transparent.

Throughout this section we let $\mathcal{I} = (G, \mathcal{L}, x, y)$, B and $\mathcal{A}$ be fixed as in the statement of the theorem; we let $w = w_{\mathcal{I}}$ throughout. The proof of the theorem is by giving an algorithm that uses $\mathcal{A}$ as a subroutine. We first give the non-polynomial-time algorithm (with the better guarantee) in Section 5.1, followed by Section 5.2 where we modify the arguments so that we also efficiently find a tour (with a slightly worse guarantee).

5.1 Existence of a Good Tour

The idea of the algorithm is to start with a collection of subtours and then iteratively merge/connect them into a single tour that visits all vertices by adding additional (cheap) subtours. We remark that since we will only add Eulerian subsets of edges, the algorithm always maintains a collection of subtours. So the state of the algorithm is described by a collection of subtours T^*.

Initialization. For the rest of the section, we assume that $V(B) \neq V$. Otherwise, B itself is a tour and the algorithm simply returns B. The algorithm starts by selecting non-empty subtours $T_1^*, T_2^*, \ldots, T_k^*$ such that

I1: $B, T_1^*, T_2^*, \ldots, T_k^*$ are disjoint subtours;

I2: $w(T_i^*) \leqslant 2\alpha\,\mathrm{lb}(T_i^*)$ for $i = 1, 2, \ldots, k$;

I3: the tuple $\langle \mathrm{lb}(T_1^*), \mathrm{lb}(T_2^*), \ldots, \mathrm{lb}(T_k^*)\rangle$ is lexicographically maximal.

As the lexicographic order is maximized, the subtours are ordered so that $\mathrm{lb}(T_1^*) \geqslant \mathrm{lb}(T_2^*) \geqslant \cdots \geqslant \mathrm{lb}(T_k^*)$. The set T^* is initialized as $T^* = T_0^* \cup T_1^* \cup T_2^* \cup \cdots \cup T_k^*$, where we let $T_0^* = B$.

During the execution of the algorithm we will also use the following concept. For a subtour T of G, let $\mathrm{ind}(T)$ be the smallest index of a subtour in $T_0^*, \ldots, T_k^*$ that it intersects (or ∞ if it intersects none). That is,

$$\mathrm{ind}(T) = \min\{i : V(T_i^*) \cap V(T) \neq \emptyset\}.$$

Moreover, an important quantity will be $\mathrm{lb}(T^*_{\mathrm{ind}(T)})$, with the convention that $\mathrm{lb}(T^*_\infty) = 0$.

Remark 5.3. The main difference in the polynomial-time algorithm is the initialization, as we do not know how to find an initialization satisfying I1-I3 in polynomial time. Indeed, it is consistent with our knowledge that 2α (even 2) is an upper bound on the integrality gap and, in that case, such an algorithm would always find a tour for singleton instances with $B = \emptyset$.

Remark 5.4. For intuition, let us mention that the reason to maximize the lexicographic order (subject to I1-I2) is that we will use the following properties to bound the weight of the final tour:

1. Let T be a subtour with $\mathrm{ind}(T) = i > 0$ and $w(T) \leqslant 2\alpha\,\mathrm{lb}(T)$. Then $\mathrm{lb}(T) \leqslant \mathrm{lb}(T^*_i)$.
2. For any disjoint subtours $T_1, T_2, \ldots, T_\ell$ with $\mathrm{ind}(T_j) = i > 0$ and $w(T_j) \leqslant \alpha\,\mathrm{lb}(T_j)$ for $j = 1, \ldots, \ell$, we have

$$\sum_{j=1}^{\ell} \mathrm{lb}(T_j) \leqslant 2\,\mathrm{lb}(T^*_i).$$

These claims will be used to bound the weight of the subtours added in the merge procedure (see below). Their proofs are easy and can be found in the analysis (see the proofs of Claim 5.10 and Claim 5.11).

Merge procedure. After the initialization, T^* contains a collection of subtours that do not necessarily form a tour. The goal of the "merge procedure" is to form a tour of the entire graph, connecting these subtours by adding additional (cheap) subtours. We will do so while maintaining the invariant that $T^*_0 = B$ is a disjoint subtour in T^* until the very last step when a tour is formed. We emphasize that even though T^* changes throughout the procedure, the index $\mathrm{ind}(T)$ will always be defined with respect to the original subtours $T^*_0, \ldots, T^*_k$.

Specifically, the procedure repeats the following until T^* is a tour (that visits all the vertices). Let $T^*_0, T_1, \ldots, T_\ell$ be the collection of subtours in T^* (recall that $T^*_0 = B$). As they are disjoint, $T_1, T_2, \ldots, T_\ell$ naturally partition the vertex set $V \setminus V(B)$ into $V(T_1), V(T_2), \ldots, V(T_\ell)$ plus singleton sets for the remaining vertices in $V \setminus (V(B) \cup V(T_1) \cup \ldots \cup V(T_\ell))$. Let C be this partitioning of $V \setminus V(B)$. By construction, each nonsingleton set in C corresponds to a subtour and thus, for each $V' \in C$, the subgraph induced by V' is strongly connected. We can therefore use $\mathcal{A}$ to find a collection F of subtours such that

(i) $|\delta^+_F(V')| \geqslant 1$ for all $V' \in C$,

(ii) $w_{\mathcal{I}}(T) \leqslant \alpha\,\mathrm{lb}(T)$ for every subtour T in F disjoint from B, and

(iii) $w(F_B) \leqslant \beta$, where $F_B \subseteq F$ is the collection of subtours in F that intersect B.

Note that $\mathcal{A}$ is guaranteed to find such a collection F since it is assumed to be an (α, β)-light algorithm for Subtour Partition Cover on $(\mathcal{I}, B)$. Furthermore, we may assume that a subtour T in F does not only visit a subset $V(T)$ of the vertices $V(T')$ visited by a subtour T' in T^*. Indeed, such a subtour can safely be removed from

F, yielding a new (smaller) collection of subtours that satisfies the above conditions. Having selected F, we now proceed to explain the "update phase".

U1: Let $X = \emptyset$.

U2: Select a subtour T in $T^* \cup F \cup X$ that *maximizes* $\mathrm{ind}(T)$. Let $j = \mathrm{ind}(T)$.

U3: If there exists a cycle C of weight $w(C) \leqslant \alpha\, \mathrm{lb}(T_j^*)$ that connects T to other vertices, i.e., $V(T) \cap V(C) \neq \emptyset$ and $V(C) \nsubseteq V(T)$, then add C to X and repeat from Step U2.

U4: Otherwise, update T^* by adding the "new" edges in T, i.e., $T^* \leftarrow T^* \cup (T \cap F) \cup (T \cap X)$.

Some comments about the update of T^* are in order. We emphasize that we do *not* add all edges of $F \cup X$ to T^*. Instead, we only add those new edges that belong to the component T selected in the final iteration of the update phase. Among other things, this ensures the invariant that B is a subtour in T^* until the very end. Indeed, the iteration where we add a subtour intersecting $B = T_0^*$ must be the last iteration. This is because, in that case, the selected T that maximizes $\mathrm{ind}(T)$ must satisfy $\mathrm{ind}(T) = 0$, which in turn implies that $T^* \cup F \cup X$ is a single tour T that visits all vertices.

Finally, let us remark that the update maintains that T^* is a collection of subtours (i.e., an Eulerian multiset of edges). As T is a subtour in $T^* \cup F \cup X$, and F and X themselves are collections of subtours, we have that T^* remains a collection of subtours after the update. This finishes the description of the merging procedure and the algorithm (see also the example below).

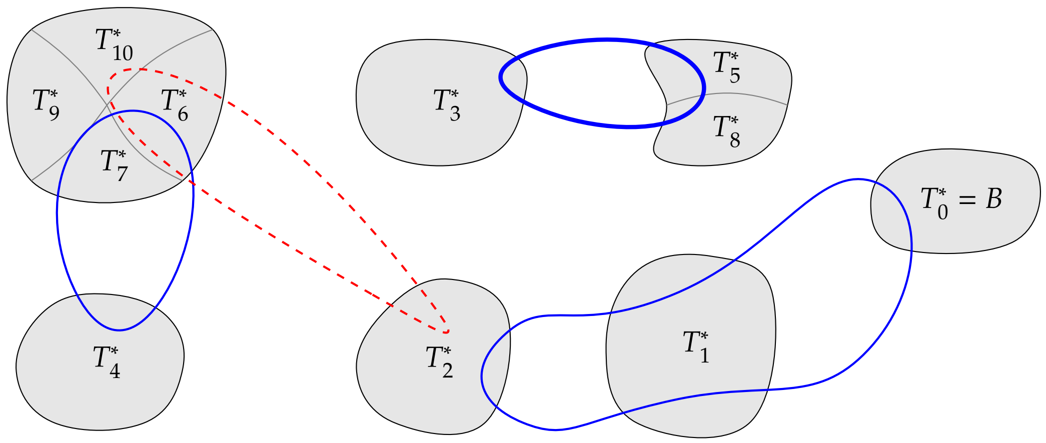


Figure 3: An illustration of the merge procedure. The gray areas depict the subtours in T^*. Blue (solid) cycles depict F and the red (dashed) cycle depicts X after one iteration of the update phase. The thick cycle represents the edges that this merge procedure would add to T^*.

Example 5.5. In Figure 3, we have that, at the start of a merging step, T^* consists of 7 subtours containing $\{T_6^*, T_7^*, T_9^*, T_{10}^*\}$, $\{T_3^*\}$, $\{T_5^*, T_8^*\}$, $\{T_4^*\}$, $\{T_2^*\}$, and $\{T_1^*\}$. The blue (solid) cycles depict the subtours of F. First, we set $X = \emptyset$ and the algorithm selects the subtour T in $T^* \cup F \cup X$ that maximizes $\mathrm{ind}(T)$. In this example, it would be the leftmost of the three subtours in $T^* \cup F$, with $\operatorname{ind}(T) = 4$. The algorithm now tries to connect this component to another component by adding a cycle with weight at most $\alpha \operatorname{lb}(T_4^*)$. The red (dashed) cycle corresponds to such a cycle and its edge set is added to X. In the next iteration, the algorithm considers the two subtours in $T^* \cup F \cup X$. The one that maximizes $\operatorname{ind}(T)$ contains T_3^*, T_5^*, and T_8^*. Now suppose that there is no cycle of weight at most $\alpha \operatorname{lb}(T_3^*)$ that connects this component to another component. Then the set T^* is updated by adding those subtours (edges) of $F \cup X$ that belong to this component (depicted by the thick cycle).

5.1.1 Analysis

Termination. We show that the algorithm terminates by arguing that the update phase decreases the number of connected components and the merge procedure is therefore repeated at most $k \leqslant n$ times.

Lemma 5.6. *The update phase terminates in polynomial time and decreases the number of connected components in (V, T^*).*

Proof. First, observe that each single step of the update phase can be implemented in polynomial time. The only nontrivial part is Step U3, which can be implemented as follows: for each edge $(u, v) \in \delta^+(V(T))$ consider the cycle consisting of (u, v) and a shortest path from v to u. Moreover, the entire update phase terminates in polynomial time, because each time the if-condition of Step U3 is satisfied, we add a cycle to X that decreases the number of connected components in $(V, T^* \cup F \cup X)$. The if-condition of Step U3 can therefore be satisfied at most $k \leqslant n$ times.

We proceed by proving that, at termination, the update phase decreases the number of connected components in (V, T^*). Recall that, once the algorithm reaches Step U4, it has selected a subtour T in $T^* \cup F \cup X$. We claim that T visits vertices in at least two components of (V, T^*). This is because the subtours in F satisfy $|\delta_F^+(V')| \geqslant 1$ for all $V' \in \mathcal{C}$, where $\mathcal{C}$ denotes the connected components of $(V \setminus V(B), T^* \setminus B)$. Moreover, as already noted, T intersects B only in the last iteration, when T forms a tour. Therefore, when the algorithm updates T^* by adding the edges $(F \cup X) \cap T$, it decreases the number of components in (V, T^*) by at least one. □

Performance Guarantee. We split our analysis of the performance guarantee into two parts. Note that when one execution of the merge procedure terminates (Step U4), we add the edge set $(F \cap T) \cup (X \cap T)$ to our solution. We will analyze the contribution of these two sets ($F \cap T$ and $X \cap T$) separately. More formally, suppose that the algorithm performs R repetitions of the merge procedure. Let $T_1, T_2, \ldots, T_R$, $F_1, F_2, \ldots, F_R$, and $X_1, X_2, \ldots, X_R$ denote the selected subtour T, the edge set F, and the edge set X, respectively, at the end of each repetition. To simplify notation, we denote the edges added to T^* in the r-th repetition by $\tilde{F}_r = F_r \cap T_r$ and $\tilde{X}_r = X_r \cap T_r$.

With this notation, we proceed to bound the total weight of the solution by

$$\underbrace{w\left(\bigcup_{r=1}^{R} \tilde{F}_r\right)}_{\leqslant 2\alpha\,\mathrm{lb}(\bar{B})+\beta \text{ by Lemma 5.8}} + \underbrace{w\left(\bigcup_{r=1}^{R} \tilde{X}_r\right)}_{\leqslant \alpha\,\mathrm{lb}(\bar{B}) \text{ by Lemma 5.7}} + w(B) + \sum_{i=1}^{k} w(T_i^*)$$
$$\leqslant 5\alpha\,\mathrm{lb}(\bar{B}) + \beta + w(B),$$

as claimed in Theorem 5.1. Here we used that $\sum_{i=1}^{k} w(T_i^*) \leqslant 2\alpha\,\mathrm{lb}(\bar{B})$, which is implied by the selection of $T_1^*, T_2^*, \ldots, T_k^*$ (I1-I2). It remains to prove Lemmas 5.7 and 5.8.

Lemma 5.7. *We have* $w\left(\bigcup_{r=1}^{R} \tilde{X}_r\right) \leqslant \alpha\,\mathrm{lb}(\bar{B})$.

Proof. Note that $\tilde{X}_r$ consists of a subset of the cycles added to X_r in Step U3 of the update phase: specifically, of those cycles that were contained in the subtour T_r selected in Step U2 in the last iteration of the update phase during the r-th repetition of the merge procedure. We can therefore decompose $\bigcup_{r=1}^{R} \tilde{X}_r$ into cycles $C_1, C_2, \ldots, C_c$, indexed in the order they were added by the algorithm. We assume that all these cycles have strictly positive weight, as 0-weight cycles do not affect $w\left(\bigcup_{r=1}^{R} \tilde{X}_r\right)$. At the time a cycle C_i (with $w(C_i) > 0$) was selected in Step U3 of the update phase, it satisfied the following two properties:

(i) it connected the subtour T with $\mathrm{ind}(T) = j > 0$ selected in Step U2 with at least one other subtour T' such that $\mathrm{ind}(T') < \mathrm{ind}(T)$; and

(ii) it had weight $w(C_i) \leqslant \alpha\,\mathrm{lb}(T_j^*)$.

In this case, we say that C_i is marked by j. Note that $1 \leqslant j \leqslant k$, since $\alpha\,\mathrm{lb}(T_j^*) \geqslant w(C_i) > 0$ and by convention $\mathrm{lb}(T_\infty^*) = 0$.

We claim that at most one cycle in $C_1, C_2, \ldots, C_c$ is marked by each of the numbers $\{1, 2, \ldots, k\}$. To see this, consider the first cycle C_i marked by j (if any). By *(i)* above, when C_i was added, it connected two subtours T and T' such that $\mathrm{ind}(T') < \mathrm{ind}(T) = j$. As the algorithm only adds edges, T and T' will remain connected throughout the execution of the algorithm. Therefore, by the definition of ind and by the fact that $\mathrm{ind}(T') < \mathrm{ind}(T)$, we have that a subtour T'' selected in Step U2 later in the algorithm always has $\mathrm{ind}(T'') \neq j$. Hence, no other cycle will be marked by j.

The bound now follows since at most one cycle with positive weight is marked by j, and such a cycle has weight at most $\alpha\,\mathrm{lb}(T_j^*)$. Moreover, we have $\sum_{j=1}^{k} \alpha\,\mathrm{lb}(T_j^*) \leqslant \alpha\,\mathrm{lb}(\bar{B})$, which is again implied by the selection of $T_1^*, T_2^*, \ldots, T_k^*$ (I1-I2). □

We complete the analysis of the performance guarantee with the following lemma. We remark that this is the only part of the proof that relies on the initial subtours $T_1^*, \ldots, T_k^*$ maximizing the lexicographic order (I3).

Lemma 5.8. *We have* $w\left(\bigcup_{r=1}^{R} \tilde{F}_r\right) \leqslant 2\alpha\,\mathrm{lb}(\bar{B}) + \beta$.

Proof. Consider the r-th repetition of the merge procedure. Partition the collection of subtours $\tilde{F}_r$ into

$$\tilde{F}_r^i = \{T \text{ subtour in } \tilde{F}_r : \mathrm{ind}(T) = i\} \quad \text{for } i \in \{0, 1 \ldots, k, \infty\}.$$

That is, $\tilde{F}_r^i$ contains those subtours in $\tilde{F}_r$ that intersect T_i^* and do not intersect any of the subtours $T_0^* = B, T_1^*, T_2^*, \ldots, T_{i-1}^*$ (or intersect none if $i = \infty$). The total weight $w(\tilde{F}_r)$ of $\tilde{F}_r$ thus equals

$$w(\tilde{F}_r^0) + w(\tilde{F}_r^\infty) + \sum_{i=1}^{k} w(\tilde{F}_r^i).$$

We bound the weight of $\tilde{F}_r$ by considering these terms separately. Let us start with $\tilde{F}_r^0$.

Claim 5.9. The set $\tilde{F}_r^0$ can be non-empty only for $r = R$, and $w(\tilde{F}_R^0) \leqslant \beta$.

Proof. The first claim follows by the invariant that B is a subtour in T^* until the very last iteration of the merge procedure. Indeed, if $\tilde{F}_r^0 \neq \emptyset$, then the algorithm must terminate after the r-th merge procedure: the subtour T selected in Step U2 must visit all vertices. For the second part, we have that every $T \in \tilde{F}_R^0$ must intersect $T_0^* = B$. Therefore, property (iii) of the edge set $\tilde{F}_R$ returned by $\mathcal{A}$ asserts that $w(\tilde{F}_R^0) \leqslant \beta$. □

For $i > 0$, we start by two simple claims that follow since each subtour T in $\tilde{F}_r$ satisfies $w(T) \leqslant \alpha\,\mathrm{lb}(T)$ (by property (ii) of $\mathcal{A}$) and the choice of $T_1^*, \ldots, T_k^*$ to maximize the lexicographic order I3 subject to I1-I2.

Claim 5.10. For $i > 0$ and $T \in \tilde{F}_r^i$ we have $\mathrm{lb}(T) \leqslant \mathrm{lb}(T_i^*)$.

Proof. The inequality $\mathrm{lb}(T) > \mathrm{lb}(T_i^*)$ together with the fact that $w(T) \leqslant \alpha\,\mathrm{lb}(T) \leqslant 2\alpha\,\mathrm{lb}(T)$ would contradict that $T_1^*, \ldots, T_k^*$ was chosen to maximize the lexicographic order I3. Indeed, in that case, a an initialization satisfying I1-I2 of higher lexicographic order would be $T_1^*, \ldots, T_{i-1}^*, T$. □

Together with $\mathrm{lb}(T_\infty^*) = 0$, this claim implies that $w(\tilde{F}_r^\infty) = 0$. We now present a more general claim that also applies to $\tilde{F}_r^i$ with $1 \leqslant i \leqslant k$.

Claim 5.11. If $i > 0$, then we have $\mathrm{lb}(\tilde{F}_r^i) \leqslant 2\,\mathrm{lb}(T_i^*)$.

Proof. Suppose towards a contradiction that $\mathrm{lb}(\tilde{F}_r^i) > 2\,\mathrm{lb}(T_i^*)$. Let $T_1, T_2, \ldots, T_\ell$ be the subtours in $\tilde{F}_r^i$ and define T to be the subtour obtained by taking the union of the subtours T_i^* and $T_1, \ldots, T_\ell$. Consider the initialization $T_1^*, \ldots, T_{i-1}^*, T$. By construction these subtours are disjoint and disjoint from B since $i > 0$. Thus I1 is satisfied. Moreover, we have $\mathrm{lb}(T) > \mathrm{lb}(T_i^*)$ and therefore this initialization is lexicographically larger than $T_1^*, \ldots, T_k^*$. This is a contradiction if T also satisfies I2, i.e., if $w(T) \leqslant 2\alpha\,\mathrm{lb}(T)$.

Therefore, we must have $w(T) > 2\alpha\,\mathrm{lb}(T)$. By the facts that $w(T_j) \leqslant \alpha\,\mathrm{lb}(T_j)$ (by property (ii) of $\mathcal{A}$) and that $w(T_i^*) \leqslant 2\alpha\,\mathrm{lb}(T_i^*)$ (by I2),

$$w(T) = w(T_i^*) + \sum_{j=1}^{\ell} w(T_j) \leqslant 2\alpha\,\mathrm{lb}(T_i^*) + \sum_{j=1}^{\ell} \alpha\,\mathrm{lb}(T_j) \quad \text{and} \quad \mathrm{lb}(T) \geqslant \sum_{j=1}^{\ell} \mathrm{lb}(T_j).$$

Together with $w(T) > 2\alpha \operatorname{lb}(T)$, we see that

$$2\alpha \sum_{j=1}^{\ell} \operatorname{lb}(T_j) < 2\alpha \operatorname{lb}(T_i^*) + \alpha \sum_{j=1}^{\ell} \operatorname{lb}(T_j).$$

From here, we can conclude that $\operatorname{lb}(\tilde{F}_r^i) = \sum_{j=1}^{\ell} \operatorname{lb}(T_j) \leqslant 2 \operatorname{lb}(T_i^*)$. □

Using the above claims, we can write $w\left(\bigcup_{r=1}^{R} \tilde{F}_r\right)$ as

$$\begin{aligned}\sum_{r=1}^{R}\left(w(\tilde{F}_r^0) + w(\tilde{F}_r^\infty) + \sum_{i=1}^{k} w(\tilde{F}_r^i)\right) &\leqslant \beta + \sum_{r=1}^{R}\sum_{i=1}^{k} w(\tilde{F}_r^i)\\ &\leqslant \beta + \alpha \sum_{r=1}^{R}\sum_{i=1}^{k} \operatorname{lb}(\tilde{F}_r^i)\\ &= \beta + \alpha \sum_{i=1}^{k} \sum_{r\colon \tilde{F}_r^i \neq \emptyset} \operatorname{lb}(\tilde{F}_r^i)\\ &\leqslant \beta + 2\alpha \sum_{i=1}^{k} \sum_{r\colon \tilde{F}_r^i \neq \emptyset} \operatorname{lb}(T_i^*).\end{aligned}$$

We complete the proof of the lemma by using Claim 5.10 to prove that $\tilde{F}_r^i$ is non-empty for at most one repetition r of the merge procedure. Suppose towards a contradiction that there exist $1 \leqslant r_0 < r_1 \leqslant R$ such that both $\tilde{F}_{r_0}^i \neq \emptyset$ and $\tilde{F}_{r_1}^i \neq \emptyset$. In the r_0-th repetition of the merge procedure, T_i^* was contained in the subtour T_{r_0} (selected in Step U2) since otherwise no edges incident to T_i^* would have been added to T^*. Therefore $j = \operatorname{ind}(T_{r_0}) \leqslant i$. Now consider a subtour T in $\tilde{F}_{r_1}^i$. First, recall that we have assumed that T, being a subtour in F_{r_1}, does not only visit a subset $V(T)$ of vertices $V(T')$ visited by a subtour T' in T^*. In particular, since T_{r_0} is a subset of some subtour T' in T^* during the r_1-th repetition, we have $V(T) \nsubseteq V(T_{r_0})$. Second, by Claim 5.10, we have $w(T) \leqslant \alpha \operatorname{lb}(T) \leqslant \alpha \operatorname{lb}(T_i^*)$.

In short, T is a subtour that connects T_{r_0} to another component and it has weight at most $\alpha \operatorname{lb}(T_i^*) \leqslant \alpha \operatorname{lb}(T_j^*)$, where $j = \operatorname{ind}(T_{r_0}) \leqslant i$. As T is Eulerian, it can be decomposed into cycles. One of these cycles, say C, connects T_{r_0} to another component and

$$w(C) \leqslant w(T) \leqslant \alpha \operatorname{lb}(T_j^*). \tag{5.2}$$

In other words, there exists a cycle C that, in the r_0-th repetition of the merge procedure, satisfied the if-condition of Step U3, which contradicts the fact that C was not added during the r_0-th repetition. □

5.2 Polynomial-Time Algorithm

In this section we describe how to modify the arguments in Section 5.1 to obtain an algorithm that runs in time polynomial in the number n of vertices, in $1/\varepsilon$, and in the running time of $\mathcal{A}$.

By Lemma 5.6, the update phase can be implemented in time polynomial in n. Therefore, the merge procedure described in Section 5.1 runs in time polynomial in n and in the running time of $\mathcal{A}$. The problem is the initialization: as mentioned in Remark 5.3, it seems difficult to give a polynomial-time algorithm for finding subtours $T_1^*, \ldots, T_k^*$ that satisfy I1 and I2 together with the third condition I3 that we should maximize the lexicographic order of

$$\langle \mathrm{lb}(T_1^*), \mathrm{lb}(T_2^*), \ldots, \mathrm{lb}(T_k^*) \rangle.$$

We overcome this obstacle by first identifying the properties that we actually use from selecting the subtours as above. We then show that we can obtain an initialization that satisfies these properties in polynomial time. Our initialization will still satisfy I1 and a relaxed variant of I2 that we now describe. To simplify notation, define

$$\overline{\mathrm{lb}}(T) = \mathrm{lb}(T) + \varepsilon \cdot \frac{|V(T)|}{n} \cdot \mathrm{lb}(\bar{B})$$

for a subtour T. Note that $\overline{\mathrm{lb}}(T)$ is a slightly increased version of $\mathrm{lb}(T)$. This increase is used to lower-bound the progress in Lemma 5.13. Also note that $\overline{\mathrm{lb}}$, like lb, is additive over vertex-disjoint subtours. Finally, we reprise the convention that $\overline{\mathrm{lb}}(T_\infty^*) = \mathrm{lb}(T_\infty^*) = 0$.

Our initializations will be collections $T_1^*, \ldots, T_k^*$ of subtours satisfying

I1: $B, T_1^*, T_2^*, \ldots, T_k^*$ are disjoint subtours;

I2′: $w(T_i^*) \leqslant 3\alpha\, \overline{\mathrm{lb}}(T_i^*)$ for $i = 1, 2, \ldots, k$.

While we do not maximize the lexicographic order, we assume that the subtours are indexed so that $\overline{\mathrm{lb}}(T_1^*) \geqslant \overline{\mathrm{lb}}(T_2^*) \geqslant \ldots \geqslant \overline{\mathrm{lb}}(T_k^*)$. Note that the difference between I2′ and I2 is that here we require $w(T_i^*) \leqslant 3\alpha\, \overline{\mathrm{lb}}(T_i^*)$ instead of $w(T_i^*) \leqslant 2\alpha\, \mathrm{lb}(T_i^*)$. The reason why we use a factor of 3 instead of 2 is that it leads to a better constant when balancing the parameters and, as previously mentioned, we use $\overline{\mathrm{lb}}$ instead of lb to lower-bound the progress in Lemma 5.13.

The main change to our initialization to achieve polynomial running time is that we do *not* maximize the lexicographic order (I3). As mentioned in the analysis in Section 5.1, the only way we use that the initialization maximizes the lexicographic order is for the proof of Lemma 5.8. In particular, this is used in the proofs of Claims 5.10 and 5.11. Instead of maximizing the lexicographic order, our polynomial-time algorithm will ensure a relaxed variant of those claims (formalized in the lemma below: see Condition (5.3)). The claimed polynomial-time algorithm is then obtained by first proving that a slight modification of the merge procedure returns a tour of value at most $9(1+\varepsilon)\alpha\, \mathrm{lb}(\bar{B}) + \beta + w(B)$ if Condition (5.3) holds, and then showing that an initialization satisfying this condition (and I1,I2′) can be found in time polynomial in n, $1/\varepsilon$, and in the running time of $\mathcal{A}$. We start by describing the modification to the merge procedure.

Modified merge procedure. The only modification to the merge procedure in Section 5.1 is that we change the update phase by relaxing the condition of the if-statement in Step U3 from $w(C) \leqslant \alpha\,\mathrm{lb}(T_j^*)$ to $w(C) \leqslant 3\alpha\,\overline{\mathrm{lb}}(T_j^*)$, where $j = \mathrm{ind}(T)$ and T is the subtour selected in Step U2. In other words, Step U3 is replaced by:

U3′: If there exists a cycle C of weight $w(C) \leqslant 3\alpha\,\overline{\mathrm{lb}}(T_j^*)$ that connects T to other vertices, i.e., $V(T) \cap V(C) \neq \emptyset$ and $V(C) \nsubseteq V(T)$, then add C to X and repeat from Step U2.

Clearly the modified merge procedure still runs in time polynomial in n and in the running time of $\mathcal{A}$. Moreover, we show that if Condition (5.3) holds then the returned tour will have the desired weight. Recall from Section 5.1 that $\tilde{F}_r$ denotes the subset of F and $\tilde{X}_r$ denotes the subset of X that were added in the r-th repetition of the (modified) merge procedure. Furthermore, we define (as in the previous section) $\tilde{F}_r^i = \{T \text{ subtour in } \tilde{F}_r : \mathrm{ind}(T) = i\}$.

Lemma 5.12. *Suppose that the algorithm is initialized with subtours $T_1^*, T_2^*, \ldots, T_k^*$ satisfying I1 and I2′. If in each repetition r of the modified merge procedure we add a subset $\tilde{F}_r$ such that*

$$\mathrm{lb}(\tilde{F}_r^i) \leqslant 3\,\overline{\mathrm{lb}}(T_i^*) \qquad \text{for all } i \in \{1, 2, \ldots, k, \infty\}, \tag{5.3}$$

then the returned tour has weight at most $9(1+\varepsilon)\alpha\,\mathrm{lb}(\bar{B}) + \beta + w(B)$.

Let us comment on the above statement before giving its proof. The bound (5.3) is a relaxation of the bound of Claim 5.11 from $\mathrm{lb}(\tilde{F}_r^i) \leqslant 2\,\mathrm{lb}(T_i^*)$ to $\mathrm{lb}(\tilde{F}_r^i) \leqslant 3\,\overline{\mathrm{lb}}(T_i^*)$; and it also implies a relaxed version of Claim 5.10: from $\mathrm{lb}(T) \leqslant \mathrm{lb}(T_i^*)$ to $\mathrm{lb}(T) \leqslant 3\,\overline{\mathrm{lb}}(T_i^*)$ (for every T in $\tilde{F}_r^i$). It is because of this relaxed bound that we modified the if-condition of the update phase (by relaxing it by the same amount); this will be apparent in the proof.

Proof. As in the analysis of the performance guarantee in Section 5.1, we can write the weight of the returned tour as

$$w\left(\bigcup_{r=1}^{R} \tilde{F}_r\right) + w\left(\bigcup_{r=1}^{R} \tilde{X}_r\right) + w(B) + \sum_{i=1}^{k} w(T_i^*).$$

To bound $w\left(\bigcup_{r=1}^{R} \tilde{X}_r\right)$, we observe that the proof of Lemma 5.7 generalizes verbatim except that the weight of a cycle marked by i is now bounded by $3\alpha\,\overline{\mathrm{lb}}(T_i^*)$ instead of by $\alpha\,\mathrm{lb}(T_i^*)$ (because of the relaxation of the bound in the if-condition in Step U3′). Hence

$$w\left(\bigcup_{r=1}^{R} \tilde{X}_r\right) \leqslant \sum_{i=1}^{k} 3\alpha\,\overline{\mathrm{lb}}(T_i^*).$$

We proceed to bound $w\left(\bigcup_{r=1}^{R} \tilde{F}_r\right)$. Using the same arguments as in the proof of Lemma 5.8, we get

$$w\left(\bigcup_{r=1}^{R} \tilde{F}_r\right) = \sum_{r=1}^{R}\left(w(\tilde{F}_r^0) + w(\tilde{F}_r^\infty) + \sum_{i=1}^{k} w(\tilde{F}_r^i)\right) \leqslant \beta + \sum_{r=1}^{R}\sum_{i=1}^{k} w(\tilde{F}_r^i)$$

$$\leqslant \beta + \alpha \sum_{i=1}^{k} \sum_{r:\, \tilde{F}_r^i \neq \emptyset} \mathrm{lb}(\tilde{F}_r^i) \leqslant \beta + \alpha \sum_{i=1}^{k} \sum_{r:\, \tilde{F}_r^i \neq \emptyset} 3\, \overline{\mathrm{lb}}(T_i^*)$$

where, for the first inequality, we used that Claim 5.9: $\sum_{r=1}^{R} w(\tilde{F}_r^0) \leqslant \beta$ generalizes verbatim from the non-constructive analysis and we have $w(\tilde{F}_r^\infty) \leqslant \alpha\, \mathrm{lb}(\tilde{F}_r^\infty) \leqslant 3\alpha\, \overline{\mathrm{lb}}(T_\infty^*) = 0$ by the assumption of the lemma; similarly, the last inequality is by the assumption of the lemma.

Now using that the subtours are indexed so that $\overline{\mathrm{lb}}(T_1^*) \geqslant \overline{\mathrm{lb}}(T_2^*) \geqslant \ldots \geqslant \overline{\mathrm{lb}}(T_k^*)$ we apply exactly the same arguments as in the end of the proof of Lemma 5.8 to prove that $\tilde{F}_r^i$ is non-empty for at most one repetition r of the merge procedure. The only difference is that (5.2) becomes

$$w(C) \leqslant w(T) \leqslant 3\alpha\, \overline{\mathrm{lb}}(T_j^*)$$

(because (5.3) can be seen as a relaxed version of Claim 5.10). However, as we also updated the bound in the if-condition, the argument that C would satisfy the if-condition of Step U3′ is still valid. Hence, we conclude that $\tilde{F}_r^i$ is non-empty in at most one repetition and therefore

$$w\left(\bigcup_{r=1}^{R} \tilde{F}_r\right) \leqslant \beta + \alpha \sum_{i=1}^{k} \sum_{r:\, \tilde{F}_r^i \neq \emptyset} 3\, \overline{\mathrm{lb}}(T_i^*) \leqslant \beta + \sum_{i=1}^{k} 3\alpha\, \overline{\mathrm{lb}}(T_i^*)\,.$$

By the above bounds and since the initialization $T_1^*, T_2^*, \ldots, T_k^*$ satisfies I1 and I2′, the weight of the returned tour is

$$\begin{aligned}
w\left(\bigcup_{r=1}^{R} \tilde{F}_r\right) &+ w\left(\bigcup_{r=1}^{R} \tilde{X}_r\right) + w(B) + \sum_{i=1}^{k} w(T_i^*) \\
&\leqslant \beta + \sum_{i=1}^{k} 3\alpha\, \overline{\mathrm{lb}}(T_i^*) + \sum_{i=1}^{k} 3\alpha\, \overline{\mathrm{lb}}(T_i^*) + w(B) + \sum_{i=1}^{k} w(T_i^*) \\
&\leqslant 9\alpha \sum_{i=1}^{k} \overline{\mathrm{lb}}(T_i^*) + \beta + w(B) \\
&\leqslant 9(1+\varepsilon)\alpha\, \mathrm{lb}(\bar{B}) + \beta + w(B).
\end{aligned}$$

□

Finding a good initialization in polynomial time. By the above Lemma 5.12, it is sufficient to find an initialization such that I1, I2′ are satisfied and Condition (5.3) holds during the execution of the modified merge procedure. However, how can we do it in polynomial time? We proceed as follows. First, we select the trivial *empty* initialization that consists of no subtours. Then we run the modified merge procedure and, in each repetition, we verify that Condition (5.3) holds. Note that this condition is easy to verify in time polynomial in n. If it holds until we return a tour, then we know by Lemma 5.12 that the tour has weight at most $9(1+\varepsilon)\alpha\, \mathrm{lb}(\bar{B}) + \beta + w(B)$. If it does not hold during some repetition, then we will restart the algorithm with a new initialization that we find using the following lemma. We continue in this manner until the merge procedure executes without violating Condition (5.3) and therefore returns a tour of weight at most $9(1+\varepsilon)\alpha\,\mathrm{lb}(\bar{B}) + \beta + w(B)$.

Lemma 5.13. *Suppose that some repetition of the (modified) merge procedure violates Condition* (5.3) *when run starting from an initialization $T^*_1, T^*_2, \dots, T^*_k$ satisfying I1 and I2′. Then we can, in time polynomial in n, find a new initialization $T'_1, T'_2, \dots, T'_{k'}$ such that I1, I2′ are satisfied and*

$$\sum_{j=1}^{k'} \overline{\mathrm{lb}}(T'_j)^2 - \sum_{j=1}^{k} \overline{\mathrm{lb}}(T^*_j)^2 \geqslant \frac{\varepsilon^2}{3n^2}\,\mathrm{lb}(\bar{B})^2. \tag{5.4}$$

Note that the above lemma implies that we will reinitialize (in polynomial time) at most $3n^2(1+\varepsilon)^2/\varepsilon^2$ times, because any initialization $T^*_1, \dots, T^*_k$ has $\sum_{i=1}^{k} \overline{\mathrm{lb}}(T^*_i)^2 \leqslant \left((1+\varepsilon)\,\mathrm{lb}(\bar{B})\right)^2$. As each execution of the merge procedure takes time polynomial in n and in the running time of $\mathcal{A}$, we can therefore find a tour of weight at most $9(1+\varepsilon)\alpha\,\mathrm{lb}(\bar{B}) + \beta + w(B)$ in the time claimed in Theorem 5.1, i.e., polynomial in n, $1/\varepsilon$, and the running time of $\mathcal{A}$. It remains to prove the lemma.

Proof. Suppose the r-th repetition of the merge procedure violates Condition (5.3), that is, there is an $i \in \{1, 2, \dots, k, \infty\}$ such that

$$\mathrm{lb}(\tilde{F}^i_r) > 3\,\overline{\mathrm{lb}}(T^*_i)\,.$$

Suppose first $i = \infty$. Then $\tilde{F}^\infty_r \neq \emptyset$. Let T be a subtour in $\tilde{F}^\infty_r$. We have $w(T) \leqslant \alpha\,\mathrm{lb}(T) \leqslant \alpha\,\overline{\mathrm{lb}}(T)$ by property (ii) of $\mathcal{A}$ and T is disjoint from $T^*_1, \dots, T^*_k$ and B by the definition of $\tilde{F}^\infty_r$. We can therefore compute (in polynomial time) a new initialization $T'_1 = T^*_1, T'_2 = T^*_2, \dots, T'_k = T^*_k, T'_{k+1} = T$ with $k' = k+1$ such that I1, I2′ are satisfied and

$$\sum_{j=1}^{k'} \overline{\mathrm{lb}}(T'_j)^2 - \sum_{j=1}^{k} \overline{\mathrm{lb}}(T^*_j)^2 = \overline{\mathrm{lb}}(T'_{k+1})^2 = \overline{\mathrm{lb}}(T)^2 \geqslant \frac{\varepsilon^2}{n^2}\,\mathrm{lb}(\bar{B})^2\,,$$

where the last inequality is by the definition of $\overline{\mathrm{lb}}$ and $|V(T)| \geqslant 1$.

We now consider the case when $\mathrm{lb}(\tilde{F}^i_r) > 3\,\overline{\mathrm{lb}}(T^*_i)$ for an $i \in \{1, \dots, k\}$. Let $I \subseteq \{1, 2, \dots, k\}$ be the indices of those subtours of $T^*_1, T^*_2, \dots, T^*_k$ that intersect subtours in $\tilde{F}^i_r$. Note that, by definition, we have $i \in I$ and $j \geqslant i$ for all $j \in I$. We construct a new initialization as follows:

– Sort $I \setminus \{i\} = \{t_1, \dots, t_{|I|-1}\}$ so that

$$\frac{\overline{\mathrm{lb}}(T^*_{t_j} \setminus \tilde{F}^i_r)}{\overline{\mathrm{lb}}(T^*_{t_j} \cap \tilde{F}^i_r)} \geqslant \frac{\overline{\mathrm{lb}}(T^*_{t_{j+1}} \setminus \tilde{F}^i_r)}{\overline{\mathrm{lb}}(T^*_{t_{j+1}} \cap \tilde{F}^i_r)}\,,$$

where for a subtour T we simplify notation by writing $\overline{\mathrm{lb}}(T \setminus \tilde{F}^i_r)$ and $\overline{\mathrm{lb}}(T \cap \tilde{F}^i_r)$ for $\overline{\mathrm{lb}}(V(T) \setminus V(\tilde{F}^i_r))$ and $\overline{\mathrm{lb}}(V(T) \cap V(\tilde{F}^i_r))$, respectively.

– Let S be the minimal (possibly empty) prefix of indices $t_1, t_2, \dots, t_s$ such that

$$\sum_{j \in S} \overline{\mathrm{lb}}(T^*_j \setminus \tilde{F}^i_r) \geqslant \frac{1}{3} \sum_{j \in I \setminus \{i\}} \overline{\mathrm{lb}}(T^*_j \setminus \tilde{F}^i_r) - \overline{\mathrm{lb}}(T^*_i)\,.$$

– Define T^* to be the subtour obtained by taking T^*_i, the union of all the subtours in $\tilde{F}^i_r$, and all the subtours $\{T^*_j\}_{j \in S}$. Note that this is a single (i.e., connected) subtour since every subtour in $\tilde{F}^i_r$ intersects T^*_i, and every subtour T^*_j with $j \in S \subseteq I$ intersects a subtour in $\tilde{F}^i_r$.

– Reinitialize with subtours T^* and $\{T^*_j\}_{j \notin I}$.

All the above steps can be computed in polynomial time. Moreover, the new initialization still satisfies I1 by the definition of I and since T^* does not intersect B. We now use the way S was selected to prove that I2′ still holds and that the "potential" function has increased as stated in (5.4). As we will calculate below, the increase of the potential function is simply because we required that $\sum_{j \in S} \overline{\mathrm{lb}}(T^*_j \setminus \tilde{F}^i_r) \geqslant \frac{1}{3} \sum_{j \in I \setminus \{i\}} \overline{\mathrm{lb}}(T^*_j \setminus \tilde{F}^i_r) - \overline{\mathrm{lb}}(T^*_i)$. That I2′ holds, i.e., that $w(T^*) \leqslant 3\alpha\, \overline{\mathrm{lb}}(T^*)$, follows since we selected S to be the minimal prefix with respect to the prescribed ordering, which prefers subtours that have small overlap with $\tilde{F}^i_r$ and therefore contribute significantly to $\overline{\mathrm{lb}}(T^*)$. We now formalize this intuition.

Claim 5.14. We have $w(T^*) \leqslant 3\alpha\, \overline{\mathrm{lb}}(T^*)$.

Proof. As S is selected to be a minimal prefix and every $j \in I$ satisfies $\overline{\mathrm{lb}}(T^*_j) \leqslant \overline{\mathrm{lb}}(T^*_i)$, we claim that

$$\sum_{j \in S} \overline{\mathrm{lb}}(T^*_j \setminus \tilde{F}^i_r) \leqslant \frac{1}{3} \sum_{j \in I \setminus \{i\}} \overline{\mathrm{lb}}(T^*_j \setminus \tilde{F}^i_r)\,. \tag{5.5}$$

This is trivially true if $S = \emptyset$; otherwise, since the prefix $S \setminus \{t_s\}$ was not chosen, we had

$$\sum_{j \in S \setminus \{t_s\}} \overline{\mathrm{lb}}(T^*_j \setminus \tilde{F}^i_r) < \frac{1}{3} \sum_{j \in I \setminus \{i\}} \overline{\mathrm{lb}}(T^*_j \setminus \tilde{F}^i_r) - \overline{\mathrm{lb}}(T^*_i)$$

and thus indeed

$$\sum_{j \in S} \overline{\mathrm{lb}}(T^*_j \setminus \tilde{F}^i_r) < \frac{1}{3} \sum_{j \in I \setminus \{i\}} \overline{\mathrm{lb}}(T^*_j \setminus \tilde{F}^i_r) \underbrace{- \overline{\mathrm{lb}}(T^*_i) + \overline{\mathrm{lb}}(T^*_{t_s} \setminus \tilde{F}^i_r)}_{\leqslant 0}\,.$$

Moreover, by the sorting of the indices in $I \setminus \{i\}$ we must then also have

$$\sum_{j \in S} \overline{\mathrm{lb}}(T^*_j \cap \tilde{F}^i_r) \leqslant \frac{1}{3} \sum_{j \in I \setminus \{i\}} \overline{\mathrm{lb}}(T^*_j \cap \tilde{F}^i_r)\,, \tag{5.6}$$

which is again trivially true if $S = \emptyset$; otherwise we write

$$\begin{aligned}
\frac{2}{3}\sum_{j\in S}\overline{\mathrm{lb}}(T_j^*\cap\tilde{F}_r^i)\cdot\frac{\overline{\mathrm{lb}}(T_{t_s}^*\setminus\tilde{F}_r^i)}{\overline{\mathrm{lb}}(T_{t_s}^*\cap\tilde{F}_r^i)} &\leqslant \frac{2}{3}\sum_{j\in S}\overline{\mathrm{lb}}(T_j^*\cap\tilde{F}_r^i)\cdot\frac{\overline{\mathrm{lb}}(T_j^*\setminus\tilde{F}_r^i)}{\overline{\mathrm{lb}}(T_j^*\cap\tilde{F}_r^i)} = \frac{2}{3}\sum_{j\in S}\overline{\mathrm{lb}}(T_j^*\setminus\tilde{F}_r^i)\\
&\leqslant \frac{1}{3}\sum_{j\in I\setminus\{i\}\setminus S}\overline{\mathrm{lb}}(T_j^*\setminus\tilde{F}_r^i) = \frac{1}{3}\sum_{j\in I\setminus\{i\}\setminus S}\overline{\mathrm{lb}}(T_j^*\cap\tilde{F}_r^i)\cdot\frac{\overline{\mathrm{lb}}(T_j^*\setminus\tilde{F}_r^i)}{\overline{\mathrm{lb}}(T_j^*\cap\tilde{F}_r^i)}\\
&\leqslant \frac{1}{3}\sum_{j\in I\setminus\{i\}\setminus S}\overline{\mathrm{lb}}(T_j^*\cap\tilde{F}_r^i)\cdot\frac{\overline{\mathrm{lb}}(T_{t_s}^*\setminus\tilde{F}_r^i)}{\overline{\mathrm{lb}}(T_{t_s}^*\cap\tilde{F}_r^i)}\,,
\end{aligned}$$

where the middle inequality is by subtracting $\frac{1}{3}\sum_{j\in S}\overline{\mathrm{lb}}(T_j^*\setminus\tilde{F}_r^i)$ from both sides of (5.5), and the remaining two are due to our sorting of indices. Next, we divide both sides by $\frac{\overline{\mathrm{lb}}(T_{t_s}^*\setminus\tilde{F}_r^i)}{\overline{\mathrm{lb}}(T_{t_s}^*\cap\tilde{F}_r^i)}$ (which is nonzero by minimality of S) and add $\frac{1}{3}\sum_{j\in S}\overline{\mathrm{lb}}(T_j^*\cap\tilde{F}_r^i)$ to both sides to obtain (5.6).

By (5.6) we have

$$\begin{aligned}
\sum_{j\in S}w(T_j^*) \leqslant 3\alpha\sum_{j\in S}\overline{\mathrm{lb}}(T_j^*) &= 3\alpha\sum_{j\in S}\overline{\mathrm{lb}}(T_j^*\setminus\tilde{F}_r^i) + 3\alpha\sum_{j\in S}\overline{\mathrm{lb}}(T_j^*\cap\tilde{F}_r^i)\\
&\leqslant 3\alpha\sum_{j\in S}\overline{\mathrm{lb}}(T_j^*\setminus\tilde{F}_r^i) + \alpha\sum_{j\in I\setminus\{i\}}\overline{\mathrm{lb}}(T_j^*\cap\tilde{F}_r^i)\\
&\leqslant 3\alpha\sum_{j\in S}\overline{\mathrm{lb}}(T_j^*\setminus\tilde{F}_r^i) + \alpha\,\overline{\mathrm{lb}}(\tilde{F}_r^i)\,,
\end{aligned}$$

where the first inequality holds because $T_1^*,\dots,T_k^*$ satisfy I2′ and the last inequality holds because they are disjoint (I1). By property (ii) of $\mathcal{A}$ and by the assumption that $\mathrm{lb}(\tilde{F}_r^i) > 3\,\overline{\mathrm{lb}}(T_i^*)$ we also have respectively that

$$w(\tilde{F}_r^i) \leqslant \alpha\,\mathrm{lb}(\tilde{F}_r^i) \leqslant \alpha\,\overline{\mathrm{lb}}(\tilde{F}_r^i) \qquad \text{and} \qquad w(T_i^*) \leqslant 3\alpha\,\overline{\mathrm{lb}}(T_i^*) < \alpha\,\overline{\mathrm{lb}}(\tilde{F}_r^i)\,.$$

These inequalities imply the claim since

$$\begin{aligned}
w(T^*) &= w(\tilde{F}_r^i) + w(T_i^*) + \sum_{j\in S}w(T_j^*)\\
&< \alpha\,\overline{\mathrm{lb}}(\tilde{F}_r^i) + \alpha\,\overline{\mathrm{lb}}(\tilde{F}_r^i) + 3\alpha\sum_{j\in S}\overline{\mathrm{lb}}(T_j^*\setminus\tilde{F}_r^i) + \alpha\,\overline{\mathrm{lb}}(\tilde{F}_r^i)\\
&= 3\alpha\,\overline{\mathrm{lb}}(\tilde{F}_r^i) + 3\alpha\sum_{j\in S}\overline{\mathrm{lb}}(T_j^*\setminus\tilde{F}_r^i)\\
&\leqslant 3\alpha\,\overline{\mathrm{lb}}(\tilde{F}_r^i) + 3\alpha\sum_{j\in S}\overline{\mathrm{lb}}(T_j^*\setminus\tilde{F}_r^i) + 3\alpha\,\overline{\mathrm{lb}}(T_i^*\setminus\tilde{F}_r^i)\\
&= 3\alpha\,\overline{\mathrm{lb}}(T^*)\,.
\end{aligned}$$

□

It remains to verify the increase of the "potential" function as stated in (5.4). By the definition of the new initialization, the increase is

$$\overline{\mathrm{lb}}(T^*)^2 - \sum_{j\in I} \overline{\mathrm{lb}}(T^*_j)^2 .$$

Let us concentrate on the first term:

$$\begin{aligned}\overline{\mathrm{lb}}(T^*)^2 &= \left(\overline{\mathrm{lb}}(\tilde{F}^i_r) + \overline{\mathrm{lb}}(T^*_i \setminus \tilde{F}^i_r) + \sum_{j\in S} \overline{\mathrm{lb}}(T^*_j \setminus \tilde{F}^i_r)\right)^2 \\ &\geqslant \overline{\mathrm{lb}}(\tilde{F}^i_r)\left(\overline{\mathrm{lb}}(\tilde{F}^i_r) + \overline{\mathrm{lb}}(T^*_i \setminus \tilde{F}^i_r) + \sum_{j\in S} \overline{\mathrm{lb}}(T^*_j \setminus \tilde{F}^i_r)\right).\end{aligned}$$

By the selection of S, the expression inside the parenthesis is at least

$$\begin{aligned}&\overline{\mathrm{lb}}(\tilde{F}^i_r) + \overline{\mathrm{lb}}(T^*_i \setminus \tilde{F}^i_r) + \frac{1}{3}\sum_{j\in I\setminus\{i\}} \overline{\mathrm{lb}}(T^*_j \setminus \tilde{F}^i_r) - \overline{\mathrm{lb}}(T^*_i) \\ &\qquad\geqslant \overline{\mathrm{lb}}(\tilde{F}^i_r) + \frac{1}{3}\sum_{j\in I} \overline{\mathrm{lb}}(T^*_j \setminus \tilde{F}^i_r) - \overline{\mathrm{lb}}(T^*_i).\end{aligned}$$

Using $\overline{\mathrm{lb}}(T^*_i) < \overline{\mathrm{lb}}(\tilde{F}^i_r)/3$, we can further lower-bound this expression by

$$\frac{1}{3}\,\overline{\mathrm{lb}}(\tilde{F}^i_r) + \frac{1}{3}\left(\overline{\mathrm{lb}}(\tilde{F}^i_r) + \sum_{j\in I} \overline{\mathrm{lb}}(T^*_j \setminus \tilde{F}^i_r)\right) \geqslant \frac{1}{3}\,\overline{\mathrm{lb}}(\tilde{F}^i_r) + \frac{1}{3}\sum_{j\in I} \overline{\mathrm{lb}}(T^*_j).$$

Finally, as $\overline{\mathrm{lb}}(T^*_j) \leqslant \overline{\mathrm{lb}}(T^*_i)$ for all $j \in I$, we have

$$\begin{aligned}\overline{\mathrm{lb}}(T^*)^2 - \sum_{j\in I} \overline{\mathrm{lb}}(T^*_j)^2 &\geqslant \overline{\mathrm{lb}}(T^*)^2 - \overline{\mathrm{lb}}(T^*_i)\sum_{j\in I} \overline{\mathrm{lb}}(T^*_j) \\ &\geqslant \overline{\mathrm{lb}}(\tilde{F}^i_r)\left(\frac{1}{3}\,\overline{\mathrm{lb}}(\tilde{F}^i_r) + \frac{1}{3}\sum_{j\in I} \overline{\mathrm{lb}}(T^*_j)\right) - \overline{\mathrm{lb}}(T^*_i)\sum_{j\in I} \overline{\mathrm{lb}}(T^*_j) \\ &= \frac{1}{3}\,\overline{\mathrm{lb}}(\tilde{F}^i_r)^2 + \underbrace{\left(\frac{\overline{\mathrm{lb}}(\tilde{F}^i_r)}{3} - \overline{\mathrm{lb}}(T^*_i)\right)}_{>0}\sum_{j\in I} \overline{\mathrm{lb}}(T^*_j) \\ &\geqslant \frac{1}{3}\left(\varepsilon\frac{\mathrm{lb}(\bar{B})}{n}\right)^2 = \frac{\varepsilon^2}{3n^2}\,\mathrm{lb}(\bar{B})^2 ,\end{aligned}$$

which completes the proof of Lemma 5.13. □

Part II

Obtaining structured instances

In the previous part we have reduced the problem of approximating ATSP to that of designing algorithms for Subtour Partition Cover. Our approach for dealing with general instances is to first simplify their structure and then to solve Subtour Partition Cover on the resulting structured instances. In this part, we show that we can obtain very structured instances by only increasing the approximation guarantee by a constant factor. In Part III we then solve Subtour Partition Cover on those instances.

The outline of this part is as follows. We begin by exploring the structure of sets in the laminar family $\mathcal{L}$: in Section 6 we study paths inside sets $S \in \mathcal{L}$ and, in Section 7, we introduce analogues of the classic graph-theoretic operations of contracting and inducing on such a set. These operations naturally give rise to a recursive algorithm that, intuitively, works as long as the contraction of some set $S \in \mathcal{L}$ results in a significant decrease in the value of the LP relaxation. In Section 8 we formally analyze this recursive algorithm and reduce the task of approximating ATSP to that of approximating ATSP on *irreducible* instances: those where no set $S \in \mathcal{L}$ brings about a significant decrease of the LP value if contracted.

Informally, every set $S \in \mathcal{L}$ in an irreducible instance has two vertices $u, v \in S$ such that the shortest path from u to v crosses a large (weighted) fraction of the sets $R \in \mathcal{L} : R \subsetneq S$ (otherwise contracting S into a single vertex, endowed with a node-weight equal to the weight of the shortest path, would lead to a decrease in the LP value). This insight, together with the approximation algorithm for singleton instances in Part I (Theorem 5.2), allows us to construct a low-weight subtour B that does not necessarily visit every vertex, but crosses every non-singleton set of $\mathcal{L}$. See the right part of Figure 1 for an example. We refer to B as a *backbone*, and to the ATSP instance and the backbone together as a *vertebrate pair*. This reduction allows us to further assume that our input is such a vertebrate pair; it is presented in Section 9.

In each of the above stages, we prove a theorem of the form: if there is a constant-factor approximation for ATSP on more structured instances, then there is a constant-factor approximation for ATSP on less structured instances. For example, an algorithm for irreducible instances implies an algorithm for laminarly-weighted instances. One can also think of making a stronger and stronger assumption on the instance without loss of generality, making it increasingly resemble a singleton instance.

6 Paths in Tight Sets

An instance $\mathcal{I} = (G, \mathcal{L}, x, y)$ will be fixed throughout this section. We say that a path P *crosses a set S k times* if $|P \cap \delta(S)| = k$. We say that a path P *traverses* a set S if both endpoints of P are in $V \setminus S$ and P crosses S at least twice (once entering and once leaving). We now exhibit properties of paths traversing tight sets. In particular, we show that the strongly connected components of a tight set S enjoy a nice path-like structure as depicted in Figure 4.

Recall from Section 2 that a set S of vertices is tight if $x(\delta(S)) = 2$. Moreover, S_{in} and

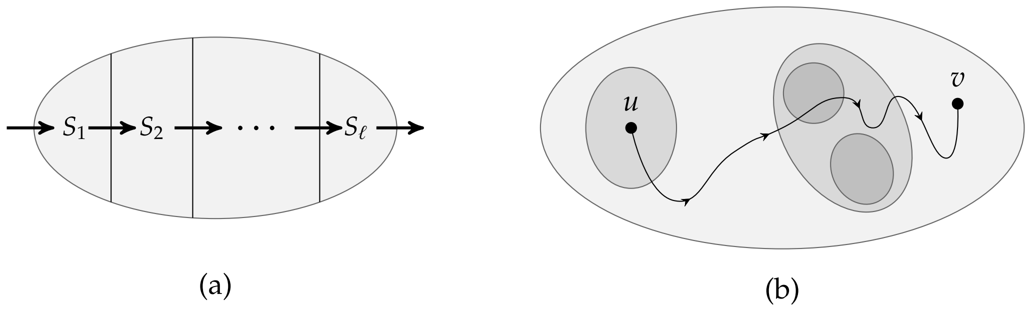


Figure 4: **(a)** The structure of a tight set S with strongly connected components $S_1, \dots, S_\ell$. Every path traversing S enters at a vertex in $S_{\text{in}} \subseteq S_1$, then visits all strongly connected components, which form a "path" structure, before it exits from a vertex in $S_{\text{out}} \subseteq S_\ell$.
(b) The structure of the path P from $u \in S_{\text{in}}$ to $v \in S_{\text{out}}$ for a tight set S as given by Lemma 6.2. The path crosses the set that contains u but not v once and it crosses the sets of $\mathcal{L}$ that are disjoint from $\{u, v\}$ at most twice.

S_{out} denote those vertices of S that have an incoming edge from outside of S and those that have an outgoing edge to outside of S, respectively.

Lemma 6.1. *For a tight set $S \subsetneq V$ we have the following properties:*

(a) *Every path from a vertex $u \in S_{in}$ to a vertex $v \in S_{out}$ (and thus every path traversing S) visits every strongly connected component of S.*

(b) *For every $u \in S_{in}$ and $v \in S$ there is a path from u to v inside S. The same holds for every $u \in S$ and $v \in S_{out}$.*

Proof. We remark that *(a)* can also be seen to follow from the "τ-narrow cut" structure as introduced in [AKS15] by setting $\tau = 0$. We give a different proof that we find simpler for our setting.

Let $S_1, S_2, \dots, S_\ell$ be the vertex sets of the strongly connected components of S, indexed using a topological ordering. We thus have that each subgraph $G[S_i]$ is strongly connected and that there is no edge from a vertex in S_i to a vertex in S_j if $i > j$.

By the above, we must have $\delta^-(S_1) \subseteq \delta^-(S)$. Moreover, since $x(\delta^-(S_1)) \geqslant 1 = x(\delta^-(S))$, we can further conclude that $\delta^-(S_1) = \delta^-(S)$ (recall that all edges have positive x-value) and $x(\delta^-(S_1)) = x(\delta^+(S_1)) = 1$ (i.e., S_1 is a tight set).

Similarly, we can show by induction on $k \geqslant 2$ that

$$\delta^-(S_k) = \delta^+(S_{k-1}) \quad \text{and} \quad x(\delta^-(S_k)) = x(\delta^+(S_k)) = 1.$$

To see this, note that $\delta^-(S_k) \subseteq \delta^-(S) \cup \bigcup_{i<k} \delta^+(S_i)$. However, $\delta^-(S) = \delta^-(S_1)$ (which is disjoint from $\delta^-(S_k)$), and for $i < k-1$, the induction hypothesis gives that $\delta^+(S_i) = \delta^-(S_{i+1})$ (which is also disjoint from $\delta^-(S_k)$). The only term left in the union is $i = k-1$ and so $\delta^-(S_k) \subseteq \delta^+(S_{k-1})$. Moreover, $1 \leqslant x(\delta^-(S_k)) \leqslant x(\delta^+(S_{k-1})) = 1$, which implies the statement for k.

Finally, we have that $\delta^+(S_\ell) = \delta^+(S)$. To recap, all incoming edges of S are into S_1, the set of outgoing edges of every component is the set of incoming edges of the next one, and all outgoing edges of S are from S_ℓ. This shows *(a)*, i.e., that every path traversing S needs to enter through S_1, exit through S_ℓ, and pass through every component on the way.

Finally, *(b)* follows because $S_{\text{in}} \subseteq S_1$ (similarly $S_{\text{out}} \subseteq S_\ell$), each two consecutive components are connected by an edge, and each component is strongly connected. □

Lemma 6.2. *Let $S \subsetneq V$ be a non-empty set such that $\mathcal{L} \cup \{S\}$ is a laminar family. Suppose $u, v \in S$ are two vertices such that there is a path from u to v inside S. Then we can in polynomial time find a path P from u to v inside S that crosses every set in $\mathcal{L}$ at most twice. Thus, the path satisfies $w(P) \leqslant \sum_{R\in\mathcal{L}:\, R\subsetneq S} 2 \cdot y_R = \text{value}(S)$.*

In addition, if $u \in S_{in}$ or $v \in S_{out}$, then P crosses every tight set $R \in \mathcal{L}$, $R \subsetneq S$ at most $2 - |R \cap \{u, v\}|$ times. Thus, it satisfies $w(P) \leqslant \sum_{R\in\mathcal{L}:\, R\subsetneq S} (2 - |R \cap \{u, v\}|) \cdot y_R$.

Proof. Since $\mathcal{L} \cup \{S\}$ is a laminar family, any path inside S only crosses those sets $R \in \mathcal{L}$ that have $R \subsetneq S$. Now, to prove both statements, it is enough to find, in polynomial time, a path P inside S that for each $R \in \mathcal{L}$ with $R \subsetneq S$ satisfies

$$|P \cap \delta(R)| \leqslant \begin{cases} 2 & \text{if } |R \cap \{u, v\}| = 0, \\ 1 & \text{if } |R \cap \{u, v\}| = 1, \\ 2 & \text{if } |R \cap \{u, v\}| = 2, \\ 0 & \text{if } |R \cap \{u, v\}| = 2 \text{ and } u \in S_{\text{in}} \text{ or } v \in S_{\text{out}}. \end{cases}$$

The algorithm for finding P starts with any path P from u to v inside S. Such a path is guaranteed to exist by the assumptions of the lemma and can be easily found in polynomial time. Now, while P does not satisfy the above conditions, select a set $R \in \mathcal{L}$ of *maximum* cardinality that violates one of the above conditions. We remark that the selected set R is tight since $R \in \mathcal{L}$. Therefore Lemma 6.1*(b)* implies that there is a path from any $u' \in R$ to any $v' \in R$ inside R if $u' \in R_{\text{in}}$ or $v' \in R_{\text{out}}$. Using this, the algorithm now modifies P depending on which of the above conditions is violated:

Case 1: $|R \cap \{u, v\}| = 0$. Let u' be the first vertex visited by P in R and let v' be the last. Then $u' \in R_{\text{in}}$ and $v' \in R_{\text{out}}$, which implies by Lemma 6.1*(b)* that there is a path Q from u' to v' inside R. We update P by letting Q replace the segment of P from u' to v'. This ensures that the set R is no longer violated, since the path P now only enters and exits R once.

Case 2: $|R \cap \{u, v\}| = 1$. This case is similar to the previous one. Suppose that $u \in R$ and $v \notin R$ (the other case is analogous). Let v' be the last vertex visited by P in R. Then $v' \in R_{\text{out}}$, and again by Lemma 6.1*(b)* there is a path Q from u to v' inside R. We update P by letting Q replace the segment of P from u to v'. This ensures that the set R is no longer violated, since the path P now only exits R once.

Case 3: $|R \cap \{u, v\}| = 2$. Let u' be the first vertex visited by P in R_{in}. By Lemma 6.1*(b)*, there is a path Q from u' to v inside R. We modify P by letting Q replace the segment of P from u' to v. This ensures that the set R is no longer violated, since the path P now only enters and exits R at most once.

Case 4: $|R \cap \{u, v\}| = 2$ and $u \in S_{\text{in}}$ or $v \in S_{\text{out}}$. Suppose that $u \in S_{\text{in}}$ (the case $v \in S_{\text{out}}$ is analogous). Then, as $R \subseteq S$, $R \cap S_{\text{in}} \subseteq R_{\text{in}}$. So, by Lemma 6.1*(b)*, there is a path Q from u to v inside R. We replace P by Q and the set R is no longer violated.

At termination, the above algorithm returns a path satisfying all the desired conditions and thus the lemma. It remains to argue that the algorithm terminates in polynomial time. A laminar family contains at most $2n-1$ sets, so it is easy to efficiently identify a violated set R of maximum cardinality. The algorithm then, in polynomial time, modifies P by simple path computations so that the set R is no longer violated. Moreover, since the modifications are such that new edges are only added within the set R, they may only introduce new violations to sets contained in R – sets of smaller cardinality. It follows, since we always select a violated set of maximum cardinality, that any set R in $\mathcal{L}$ is selected in at most one iteration. Hence, the algorithm runs for at most $|\mathcal{L}| \leqslant 2n-1$ iterations (and so it terminates in polynomial time). □

The next lemma asserts that the strongly connected components of $S \in \mathcal{L}$ form a laminar family together with $\mathcal{L}$, a property that will be needed in Section 7.2 in order to apply Lemma 6.2.

Lemma 6.3. *For a set $S \in \mathcal{L}$, let $\mathcal{S}$ be the set of strongly connected components of S. Then $\mathcal{L} \cup \mathcal{S}$ is a laminar family.*

Proof. Let $\mathcal{S} = \{S_1, S_2, \ldots, S_\ell\}$, indexed in a topological order. Towards a contradiction, suppose there exists a component S_i and a set $R \in \mathcal{L}$ such that $R \setminus S_i$, $S_i \setminus R$, and $S \cap R$ are all non-empty. Furthermore, since $\mathcal{L}$ is a laminar family and $S_i \subseteq S \in \mathcal{L}$, we must have $R \subsetneq S$. We can thus partition R into the three sets

$$R_{<i} = R \cap (S_1 \cup \cdots \cup S_{i-1}), \qquad R_i = R \cap S_i, \qquad R_{>i} = R \cap (S_{i+1} \cup \cdots \cup S_\ell)\,.$$

In words, $R_{<i}$ is the part of R that intersects vertices of the strongly connected components that are ordered topologically before S_i. Similarly, $R_{>i}$ is the part of R that intersects vertices of the strongly connected components that are ordered topologically after S_i. Note that since R is not contained in S_i, we have that either $R_{<i}$ or $R_{>i}$ is non-empty. We suppose $R_{<i} \neq \emptyset$ (the case $R_{>i} \neq \emptyset$ is analogous).

As x is a feasible solution to LP$(G, \mathbf{0})$, we have $x(\delta^-(R_{<i})) \geqslant 1$. Moreover, since $\delta(R_i \cup R_{>i}, R_{<i}) = \emptyset$ due to the topological ordering, we have $\delta^-(R_{<i}) \subseteq \delta^-(R)$ and thus

$$1 = x(\delta^-(R)) \geqslant x(\delta^-(R_{<i})) + x(\delta(S_i \setminus R_i, R_i)) \geqslant 1 + x(\delta(S_i \setminus R_i, R_i))\,,$$

where the first equality follows since $R \in \mathcal{L}$ is a tight set. However, this is a contradiction because $x(\delta(S_i \setminus R_i, R_i)) > 0$; that holds since $S_i \setminus R_i = S_i \setminus R \neq \emptyset$ and $R_i \neq \emptyset$, S_i is a strongly connected component, and G only contains edges with strictly positive x-value. □

7 Contracting and Inducing on a Tight Set

In this section we generalize two natural graph-theoretic constructions that allow one to decompose the problem of finding a tour with respect to a vertex set S. The first relies on contracting S (see Definition 7.2 in Section 7.1) and the second relies on inducing on S (see Definition 7.6 in Section 7.2).

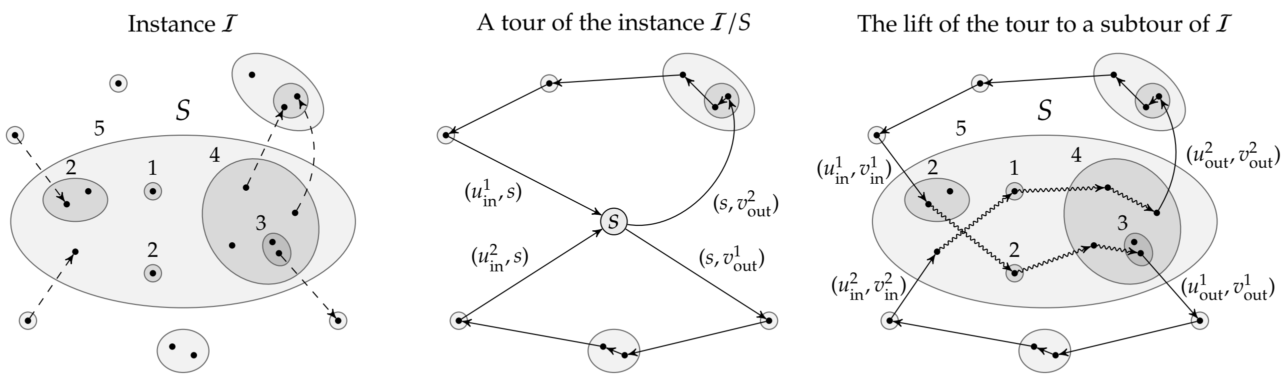

Instance $\mathcal{I}$
A tour of the instance $\mathcal{I}/S$
The lift of the tour to a subtour of $\mathcal{I}$
S
5
2
1
4
3
2
(u^1_{in}, s)
(u^2_{in}, s)
s
(s, v^2_{out})
(s, v^1_{out})
S
5
$(u^1_{\text{in}}, v^1_{\text{in}})$
2
1
4
$(u^2_{\text{out}}, v^2_{\text{out}})$
3
2
$(u^2_{\text{in}}, v^2_{\text{in}})$
$(u^1_{\text{out}}, v^1_{\text{out}})$

Figure 5: An example of the contraction of a tight set S and the lift of a tour. Only y-values of the sets $R \in \mathcal{L} : R \subseteq S$ are depicted. On the left, only edges that have one endpoint in S are shown. These are exactly the edges that are incident to s in the contracted instance. In the center, a tour of $\mathcal{I}/S$ is illustrated, and on the right we depict the lift of that tour.

7.1 Contracting a Tight Set

Consider an instance $\mathcal{I} = (G, \mathcal{L}, x, y)$. Before defining the contraction of a set $S \in \mathcal{L}$, we need to define the "distance" functions d_S and D_S. For $S \in \mathcal{L}$ and $u, v \in S$, define $d_S(u, v)$ to be the minimum weight of a path inside S from u to v (if no such path exists, $d_S(u, v) = \infty$). We also let

$$D_S(u,v) = \sum_{R \in \mathcal{L}:\ u \in R \subsetneq S} y_R + d_S(u,v) + \sum_{R \in \mathcal{L}:\ v \in R \subsetneq S} y_R\,,$$

which equals $d_S(u,v) + \sum_{R \in \mathcal{L}:\ R \subsetneq S} |R \cap \{u,v\}| \cdot y_R$ if $u \neq v$, and $\sum_{R \in \mathcal{L}:\ u \in R \subsetneq S} 2y_R$ if $u = v$. We remark that $D_S(u,u)$ might be strictly positive.

The intuition of the definition of D_S is as follows. After contracting S, all sets of the laminar family are still present in the contracted instance, except for the sets strictly contained in S. Now, after finding a tour in the contracted instance, we need to lift it back to a subtour in the original instance. This is done as depicted in Figure 5: for each visit of the tour to s (the vertex corresponding to the contraction of S) on the edges $(u^i_{\text{in}}, s), (s, v^i_{\text{out}})$, we obtain a subtour of the original instance by replacing $(u^i_{\text{in}}, s), (s, v^i_{\text{out}})$ by the corresponding edges (i.e., by their preimages) $(u^i_{\text{in}}, v^i_{\text{in}}), (u^i_{\text{out}}, v^i_{\text{out}})$ of G together with the minimum-weight path inside S from v^i_{in} to u^i_{out}. The value $D_S(v^i_{\text{in}}, u^i_{\text{out}})$ is defined to capture the weight increase incurred by this operation. For example, in Figure 5 we have

$$\underbrace{D_S(v^1_{\text{in}}, u^1_{\text{out}})}_{=22} = \underbrace{\sum_{R \in \mathcal{L}:\ v^1_{\text{in}} \in R \subsetneq S} y_R}_{=2} + \underbrace{d_S(v^1_{\text{in}}, u^1_{\text{out}})}_{=2+2\cdot 2+4+3} + \underbrace{\sum_{R \in \mathcal{L}:\ u^1_{\text{out}} \in R \subsetneq S} y_R}_{=3+4}\,.$$

Before formally defining the notions of contraction and lift, we state the following useful bound on $D_S(u, v)$.

Fact 7.1. *For any $u, v \in S$ with $u \in S_{in}$ or $v \in S_{out}$ we have*

$$D_S(u,v) \leqslant \text{value}(S).$$

Proof. Lemma 6.1*(b)* says that there is a path from u to v inside S. Select P to be the path from u to v as guaranteed by Lemma 6.2. Since $u \in S_{\text{in}}$ or $v \in S_{\text{out}}$, we have

$$d_S(u,v) \leqslant w(P) \leqslant \sum_{R \in \mathcal{L}:\ R \subsetneq S} (2 - |R \cap \{u,v\}|) \cdot y_R$$

and thus

$$D_S(u,v) = d_S(u,v) + \sum_{R \in \mathcal{L}: R \subsetneq S} |R \cap \{u,v\}| \cdot y_R \leqslant \sum_{R \in \mathcal{L}:\ R \subsetneq S} 2 \cdot y_R = \text{value}(S)\,.$$

□

We now define the notion of contracting a tight set for an ATSP instance. In short, the *contraction* is the instance obtained by performing the classic graph contraction of S, modifying $\mathcal{L}$ to remove the sets contained in S, and increasing the y-value of the new singleton $\{s\}$ corresponding to S so as to become $y_S + 1/2 \max_{u \in S_{\text{in}}, v \in S_{\text{out}}} D_S(u,v)$. This increase is done in order to pay for the maximum possible weight increase incurred when lifting a tour in the contraction back to a subtour in the original instance (as depicted in Figure 5, defined in Definition 7.4, and analyzed in Lemma 7.5).

Definition 7.2 (Contracting a tight set)**.** The instance $(G', \mathcal{L}', x', y')$ obtained from $\mathcal{I} = (G, \mathcal{L}, x, y)$ by *contracting* $S \in \mathcal{L}$, denoted by $\mathcal{I}/S$, is defined as follows:

- The graph G' equals G/S, i.e., the graph obtained from G by contracting S. Let s denote the new vertex of G' that corresponds to the set S.
- For each edge $e' \in E(G')$, $x'(e')$ equals $x(e)$, where $e \in E(G)$ is the preimage of e' in G.[7]
- The laminar family $\mathcal{L}'$ contains all remaining sets of $\mathcal{L}$:

$$\mathcal{L}' = \{(R \setminus S) \cup \{s\} : R \in \mathcal{L}, S \subseteq R\} \cup \{R : R \in \mathcal{L}, S \cap R = \emptyset\}\,.$$

- The vector y' equals y (via the natural mapping) on all sets but $\{s\}$. For $\{s\}$ we define

$$y'_s = y_S + \frac{1}{2} \max_{u \in S_{\text{in}}, v \in S_{\text{out}}} D_S(u,v)\,.$$

We remark that $\mathcal{I}/S$ as defined above is indeed an instance: $\mathcal{L}'$ is a laminar family of tight sets (since for each $R \in \mathcal{L}'$ we have $x'(\delta(R')) = x(\delta(R))$, where R is the preimage of R' in $\mathcal{L}$ via the natural mapping), $y'_R \geqslant 0$ is defined only for $R \in \mathcal{L}'$, and x' is a feasible solution to LP$(G', \mathbf{0})$ that is strictly positive on all edges.

The way we defined the new dual weight y'_s implies the natural property that the value of the linear programming solution does not increase after contracting a tight set:

[7] Recall that for notational convenience we allow parallel edges in G/S and therefore the preimage is uniquely defined.

Fact 7.3. $\text{value}(\mathcal{I}/S) = \text{value}(\mathcal{I}) - \left(\text{value}_{\mathcal{I}}(S) - \max_{u\in S_{in}, v\in S_{out}} D_S(u,v)\right) \leqslant \text{value}(\mathcal{I})$.

Proof. By definition,

$$\begin{aligned}
\text{value}(\mathcal{I}/S) = 2\cdot \sum_{R\in\mathcal{L}'} y'_R &= 2\cdot y'_s + 2\cdot \sum_{R\in\mathcal{L}:\ R\not\subseteq S} y_R \\
&= \max_{u\in S_{\text{in}}, v\in S_{\text{out}}} D_S(u,v) + 2\cdot y_S + 2\cdot \sum_{R\in\mathcal{L}:\ R\not\subseteq S} y_R \\
&= \max_{u\in S_{\text{in}}, v\in S_{\text{out}}} D_S(u,v) + 2\cdot \sum_{R\in\mathcal{L}} y_R - 2\cdot \sum_{R\in\mathcal{L}:\ R\subsetneq S} y_R \\
&= \max_{u\in S_{\text{in}}, v\in S_{\text{out}}} D_S(u,v) + \text{value}(\mathcal{I}) - \text{value}_{\mathcal{I}}(S)
\end{aligned}$$

and so the equality of the statement holds. Finally, the inequality of the statement follows from Fact 7.1, which implies that $\max_{u\in S_{\text{in}}, v\in S_{\text{out}}} D_S(u,v) \leqslant \text{value}_{\mathcal{I}}(S)$. □

Having defined the contraction of a tight set $S \in \mathcal{L}$, we define the aforementioned operation of lifting a tour of the contracted instance $\mathcal{I}/S$ to a subtour in the original instance $\mathcal{I}$. When considering a tour (or a subtour), we order the edges according to an arbitrary but fixed Eulerian walk. This allows us to talk about consecutive edges.

Definition 7.4. For a tour T of $\mathcal{I}/S$, we define its *lift* to be the subtour of $\mathcal{I}$ obtained from T by replacing each consecutive pair $(u_{\text{in}}, s), (s, v_{\text{out}})$ of incoming and outgoing edges incident to s by their preimages $(u_{\text{in}}, v_{\text{in}})$ and $(u_{\text{out}}, v_{\text{out}})$ in G, together with a minimum-weight path from v_{in} to u_{out} inside S.[8]

See Figure 5 for an illustration. It follows that the lift is a subtour (i.e., an Eulerian multiset of edges that forms a single component), because we added paths between consecutive edges in the tour of $\mathcal{I}/S$. However, the lift is usually not a tour of the instance $\mathcal{I}$, as it is not guaranteed to visit all the vertices in S. To extend the lift to a tour, we use the concept of inducing on the tight set S, which we introduce in Section 7.2.

We complete this section by bounding the weight of the lift of T.

Lemma 7.5. *Let T be a tour of the instance $\mathcal{I}/S$. Then the lift F of T satisfies $w_{\mathcal{I}}(F) \leqslant w_{\mathcal{I}/S}(T)$.*

Proof. Consider the tour T and let $(u^{(1)}_{\text{in}}, s), (s, v^{(1)}_{\text{out}}), \ldots, (u^{(k)}_{\text{in}}, s), (s, v^{(k)}_{\text{out}})$ be the edges that T uses to visit the vertex s (which corresponds to the contracted set S). That is, $(u^{(i)}_{\text{in}}, s)$ and $(s, v^{(i)}_{\text{out}})$ are the incoming and outgoing edge of the i-th visit of T to s. By the definition of contraction, we can write the weight of T as

$$\begin{aligned}
w_{\mathcal{I}/S}(T) &= \sum_{R\in\mathcal{L}:\ R\not\subseteq S} \alpha_R y_R + 2k\cdot y'_s \\
&= \sum_{R\in\mathcal{L}:\ R\not\subseteq S} \alpha_R y_R + k\cdot \left(2\cdot y_S + \max_{u\in S_{\text{in}}, v\in S_{\text{out}}} D_S(u,v)\right),
\end{aligned}$$

[8]We remark that it is not crucial that the minimum-weight path from v_{in} to u_{out} is selected to be inside S; a minimum-weight path without this restriction would also work. We have chosen this definition as we find it more intuitive and it simplifies some arguments.

where $\alpha_R = |\delta(R) \cap T|$. We now compare this weight to that of the lift F. Let $(u^{(i)}_{\text{in}}, v^{(i)}_{\text{in}})$ and $(u^{(i)}_{\text{out}}, v^{(i)}_{\text{out}})$ be the edges of G that are the preimages of $(u^{(i)}_{\text{in}}, s)$ and $(s, v^{(i)}_{\text{out}})$. The lift F is obtained from T by replacing $(u^{(i)}_{\text{in}}, s), (s, v^{(i)}_{\text{out}})$ by $(u^{(i)}_{\text{in}}, v^{(i)}_{\text{in}}), (u^{(i)}_{\text{out}}, v^{(i)}_{\text{out}})$ and adding a minimum-weight path inside S from $v^{(i)}_{\text{in}}$ to $u^{(i)}_{\text{out}}$. So F crosses every $R \in \mathcal{L} : R \not\subseteq S$ the same number of times α_R as T. To bound the weight incurred by crossing the tight sets "inside" S, note that the i-th visit to the set S incurs a weight from crossing sets $R \in \mathcal{L} : R \subseteq S$ that equals

$$2y_S + \sum_{R \in \mathcal{L}:\, v^{(i)}_{\text{in}} \in R \subsetneq S} y_R + d_S(v^{(i)}_{\text{in}}, u^{(i)}_{\text{out}}) + \sum_{R \in \mathcal{L}:\, u^{(i)}_{\text{out}} \in R \subsetneq S} y_R = 2y_S + D_S(v^{(i)}_{\text{in}}, u^{(i)}_{\text{out}})\,.$$

Hence

$$\begin{aligned} w_{\mathcal{I}}(F) &= \sum_{R \in \mathcal{L}:\, R \not\subseteq S} \alpha_R y_R + \sum_{i=1}^{k} \left(2 \cdot y_S + D_S(v^{(i)}_{\text{in}}, u^{(i)}_{\text{out}})\right) \\ &\leqslant \sum_{R \in \mathcal{L}:\, R \not\subseteq S} \alpha_R y_R + \sum_{i=1}^{k} \left(2 \cdot y_S + \max_{u \in S_{\text{in}}, v \in S_{\text{out}}} D_S(u, v)\right) \\ &= w_{\mathcal{I}/S}(T)\,. \end{aligned}$$

□

7.2 Inducing on a Tight Set

In this section we introduce our notion of induced instances. This concept will be used for completing a lift of a tour of a contracted instance into a tour of the original instance (see Definition 7.8 of "contractible" below). Inducing on a tight set S is similar to contracting its complement $V \setminus S$ into a single vertex $\bar{s}$ (see Definition 7.2), though the resulting laminar family and dual values are somewhat different: namely, we let $y'_{\bar{s}} = \text{value}(S)/2$ and we remove S (as well as all supersets of S) from $\mathcal{L}'$. The intuitive reason for the setting of $y'_{\bar{s}}$ is that each visit to $\bar{s}$ should pay for the most expensive shortest paths in the strongly connected components of S (see Figure 6 and the proof of Lemma 7.9).

We remark that the notion of inducing on S for ATSP instances differs compared to the graph obtained by inducing on S (in the usual graph-theoretic sense), as here we also have the vertex $\bar{s}$ corresponding to the contraction of the vertices not in S. This is needed to make sure that we obtain an ATSP instance (in particular, that we obtain a feasible solution x' to the linear programming relaxation).

Definition 7.6. The instance $(G', \mathcal{L}', x', y')$ obtained from $\mathcal{I} = (G, \mathcal{L}, x, y)$ by *inducing* on a tight set $S \in \mathcal{L}$, denoted by $\mathcal{I}[S]$, is defined as follows:

– The graph G' equals $G/\bar{S}$, i.e., the graph obtained from G by contracting $\bar{S} = V \setminus S$. Let $\bar{s}$ denote the new vertex of G' that corresponds to the set $\bar{S}$.

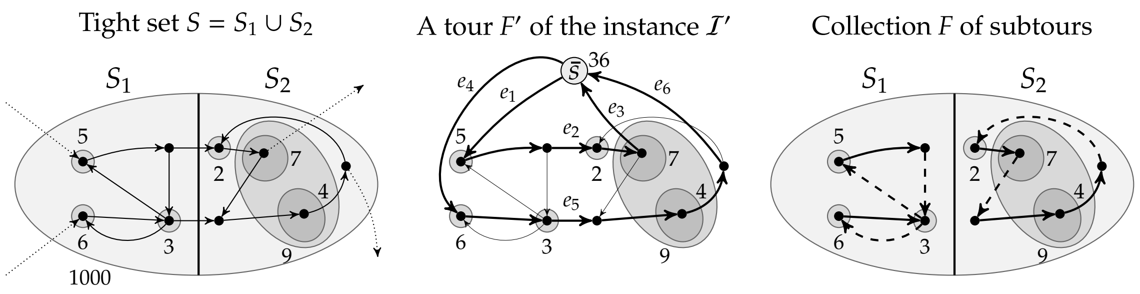

Tight set $S = S_1 \cup S_2$
S_1
S_2
5
6
3
2
7
4
9
1000
A tour F' of the instance $\mathcal{I}'$
$\bar{s}$
36
e_1
e_2
e_3
e_4
e_5
e_6
Collection F of subtours

Figure 6: In the left figure we depict a tight set $S \in \mathcal{L}$ with two strongly connected components S_1 and S_2. The induced instance (center figure) is obtained by contracting $\bar{S} = V \setminus S$ into a vertex $\bar{s}$ and removing the tight set S from $\mathcal{L}$. The solid edges are paths and edges of a tour in the induced instance. In Lemma 7.9 we obtain a collection of subtours in the original instance (right figure) by adding the dashed paths, resulting in a tour of each strongly connected component.

– For each edge $e' \in E(G')$, $x'(e')$ equals $x(e)$, where $e \in E(G)$ is the preimage of e' in G.[9]

– The laminar family $\mathcal{L}'$ contains $\{\bar{s}\}$ and all sets that are strict subsets of S:

$$\mathcal{L}' = \{R \in \mathcal{L} : R \subsetneq S\} \cup \{\{\bar{s}\}\}.$$

– The vector y' equals y on the sets common to $\mathcal{L}'$ and $\mathcal{L}$. For the new set $\{\bar{s}\}$ we define $y'_{\bar{s}} = \text{value}(S)/2$.

We remark that $\mathcal{I}[S]$ in an instance: $\mathcal{L}'$ is a laminar family of tight sets, $y'_R \geqslant 0$ is defined only for $R \in \mathcal{L}'$, and x' is a feasible solution to $\text{LP}(G', \mathbf{0})$ that is strictly positive on all edges.

As for the value of $\mathcal{I}[S]$, it is comprised of the y-values of sets strictly inside S, which contribute value(S), and that of $\{\bar{s}\}$, which also contributes $2y'_{\bar{s}} = \text{value}(S)$. Thus we have

Fact 7.7. value($\mathcal{I}[S]$) = 2 value(S).

As alluded to above, we will use the instance $\mathcal{I}[S]$ to find a collection F of subtours in the original instance $\mathcal{I}$ such that F plus a lift of a tour in $\mathcal{I}/S$ form a tour of the instance $\mathcal{I}$. We say that such a set F makes S *contractible*:

Definition 7.8. We say that $S \in \mathcal{L}$ is *contractible* with respect to a collection $F \subseteq E$ of subtours (i.e., F is an Eulerian multiset of edges) if the lift of any tour of $\mathcal{I}/S$ plus the edge set F is a tour of $\mathcal{I}$.

As an example, if F were a subtour visiting every vertex of S, then S would be contractible with respect to F. The following lemma shows that, in general, it is sufficient to find a tour of $\mathcal{I}[S]$ in order to make S contractible (see also Figure 6).

[9]We again recall that parallel edges are allowed in $G/\bar{S}$, and thus the preimage of an edge is uniquely defined.

Lemma 7.9. *Given a tour T of $\mathcal{I}[S]$, we can in polynomial time find a collection $F \subseteq E$ of subtours such that S is contractible with respect to F and $w_{\mathcal{I}}(F) \leqslant w_{\mathcal{I}[S]}(T)$.*

Proof. Let $S_1, \ldots, S_\ell$ be the strongly connected components of S indexed using a topological ordering. We will use T to obtain a low-weight tour F_i inside each S_i, and define F to be the union of these tours. Then S is contractible with respect to F. Indeed, the lift of any tour of $\mathcal{I}/S$ must contain a path traversing S, and any such path visits every connected component by Lemma 6.1*(a)*.

Let us fix one component S_i. We obtain the tour F_i of S_i by reproducing the movements of T inside S_i (recall that we think of T as a cyclically ordered Eulerian walk). More precisely, we retain those edges of T that are inside S_i and, every time T exits S_i on an edge $(u_{\text{out}}, v_{\text{out}}) \in \delta^+(S_i)$ and then returns to S_i on an edge $(u_{\text{in}}, v_{\text{in}}) \in \delta^-(S_i)$, we also insert a minimum-weight path from u_{out} to v_{in} inside S_i. Such a path exists because S_i is strongly connected. (This step corresponds to adding the dashed paths in Figure 6.) Then we set $F = F_1 \cup \ldots \cup F_\ell$.

It remains to show that F has low weight, i.e., that $w_{\mathcal{I}}(F) \leqslant w_{\mathcal{I}[S]}(T)$. For this, let k be the number of times the tour T visits the auxiliary vertex $\bar{s}$. The weight incurred by every such visit is at least $2y'_{\bar{s}} = \text{value}(S)$ (since the set $\{\bar{s}\}$ is crossed twice in each visit). Thus we have

$$w_{\mathcal{I}[S]}(T) \geqslant k \cdot \text{value}(S) + \sum_{i=1}^{\ell} w_{\mathcal{I}[S]}(T \cap E(S_i)) = k \cdot \text{value}(S) + \sum_{i=1}^{\ell} w_{\mathcal{I}}(T \cap E(S_i))\,.$$

On the other hand, each tour F_i consists of all those edges of T that are inside S_i, as well as k shortest paths between some pairs of vertices in S_i. Indeed, if T makes k visits to the auxiliary vertex $\bar{s}$, then we add exactly k paths inside S_i due to the path-like structure of the strongly connected components of a tight set S (Lemma 6.1).

By Lemma 6.3, $\mathcal{L} \cup \{S_i\}$ is a laminar family. Consequently, Lemma 6.2 is applicable to obtain that a shortest path between two vertices inside S_i has weight at most value(S_i). Recall that F_i consists of all those edges of T that are inside S_i, as well as k shortest paths between some pairs of vertices in S_i. Therefore, the weight of F is

$$\begin{aligned} w_{\mathcal{I}}(F) &= \sum_{i=1}^{\ell} w_{\mathcal{I}}(F_i) \\ &\leqslant \sum_{i=1}^{\ell} [k \cdot \text{value}(S_i) + w_{\mathcal{I}}(T \cap E(S_i))] \\ &\leqslant k \cdot \text{value}(S) + \sum_{i=1}^{\ell} w_{\mathcal{I}}(T \cap E(S_i)) \\ &\leqslant w_{\mathcal{I}[S]}(T)\,, \end{aligned}$$

as required. □

8 Reduction to Irreducible Instances

In this section we reduce the problem of approximating ATSP on general (laminarly-weighted) instances to that of approximating ATSP on irreducible instances. Specifically, Theorem 8.3 says that any approximation algorithm for irreducible instances can be turned into an algorithm for general instances while losing only a constant factor in the approximation guarantee.

We now define the notions of *reducible sets* and *irreducible instances*. The intuition behind them is as follows. The operations of contracting and inducing on a tight set S introduced in the last section naturally lead to the following recursive algorithm:

1. Select a tight set $S \in \mathcal{L}$.
2. Find a tour T_S in the induced instance $\mathcal{I}[S]$. Via Lemma 7.9, T_S yields a set F_S that makes S contractible.
3. Recursively find a tour T in the contraction $\mathcal{I}/S$.
4. Output F_S plus the lift of T.

For this scheme to yield a good approximation guarantee, we need to ensure that we can find a good approximate tour T_S in $\mathcal{I}[S]$ and that contracting the set S results in a "significant" decrease in the value of the LP solution. If it does, we refer to the set S as *reducible*:

Definition 8.1. We say that a set $S \in \mathcal{L}$ is *reducible* if

$$\max_{u \in S_{\text{in}}, v \in S_{\text{out}}} D_S(u,v) < \delta \cdot \text{value}(S)\,,$$

otherwise we say that S is *irreducible*. We also say that the instance $\mathcal{I}$ is *irreducible* if no set $S \in \mathcal{L}$ is reducible.

We will use the value $\delta = 0.78$; however, we keep it as a parameter $\delta \in (1/2, 1)$ to exhibit the dependence of the approximation ratio on this value.

Note that singleton sets are never reducible. Moreover, we have the following observation:

Fact 8.2. *Consider an instance $\mathcal{I} = (G, \mathcal{L}, x, y)$ and a set $S \in \mathcal{L}$. If every set $R \in \mathcal{L} : R \subsetneq S$ is irreducible, then $\mathcal{I}[S]$ is irreducible. In particular, if $\mathcal{I}$ is an irreducible instance, then $\mathcal{I}[S]$ is irreducible for every $S \in \mathcal{L}$.*

Proof. Let $\mathcal{I}[S] = (G', \mathcal{L}', x', y')$. By definition, $\mathcal{L}' = \{R \in \mathcal{L} : R \subsetneq S\} \cup \{\{\bar{s}\}\}$. Clearly, the singleton set $\{\bar{s}\}$ is irreducible. Now consider a set $R \in \mathcal{L} : R \subsetneq S$; we need to show that R is irreducible in $\mathcal{I}[S]$. Note that R is also present in $\mathcal{I}$ and that the sets $\{Q \in \mathcal{L} : Q \subsetneq R\}$ and $\{Q \in \mathcal{L}' : Q \subsetneq R\}$ are identical. This implies that the distance function D_R is identical in the instances $\mathcal{I}$ and $\mathcal{I}[S]$. Moreover, the sets R_{in} and R_{out} are also the same in the two instances. Therefore, as R is irreducible in $\mathcal{I}$ by assumption, we have that R is also irreducible in $\mathcal{I}[S]$. $\square$

The above fact implies that if we select $S \in \mathcal{L}$ to be a *minimal* reducible set, then the instance $\mathcal{I}[S]$ is irreducible. Hence, we only need to be able to find an approximate tour T_S for *irreducible* instances (in Step 2 of the above recursive algorithm). This is the idea behind the following theorem, and its proof is based on formally analyzing the aforementioned approach.

Theorem 8.3. *Let $\mathcal{A}$ be a polynomial-time ρ-approximation algorithm (with respect to the Held–Karp lower bound) for irreducible instances. Then there is a polynomial-time $\frac{2\rho}{1-\delta}$-approximation algorithm (with respect to the Held–Karp lower bound) for general instances.*

Proof. Consider a general instance $\mathcal{I} = (G, \mathcal{L}, x, y)$. If it is irreducible, we can simply return the result of a single call to $\mathcal{A}$. So assume that $\mathcal{I}$ is not irreducible, i.e., that $\mathcal{L}$ contains a reducible set. Let $S \in \mathcal{L}$ be a minimal (inclusion-wise) reducible set, i.e., one such that all subsets $R \in \mathcal{L} : R \subsetneq S$ are irreducible.

We will work with the induced instance $\mathcal{I}[S]$. Recall that $\text{value}(\mathcal{I}[S]) = 2\,\text{value}(S)$ (Fact 7.7). Moreover, $\mathcal{I}[S]$ is irreducible by Fact 8.2. We can therefore use $\mathcal{A}$ to find a tour T_S of $\mathcal{I}[S]$. Since $\mathcal{A}$ is a ρ-approximation algorithm, we have

$$w_{\mathcal{I}[S]}(T_S) \leqslant \rho \cdot \text{value}(\mathcal{I}[S]) = 2\rho\,\text{value}(S)\,.$$

Next, we invoke the algorithm of Lemma 7.9 to obtain a collection $F_S \subseteq E$ of subtours such that S is contractible with respect to F_S and

$$w_{\mathcal{I}}(F_S) \leqslant w_{\mathcal{I}[S]}(T_S) \leqslant 2\rho\,\text{value}(S)\,. \tag{8.1}$$

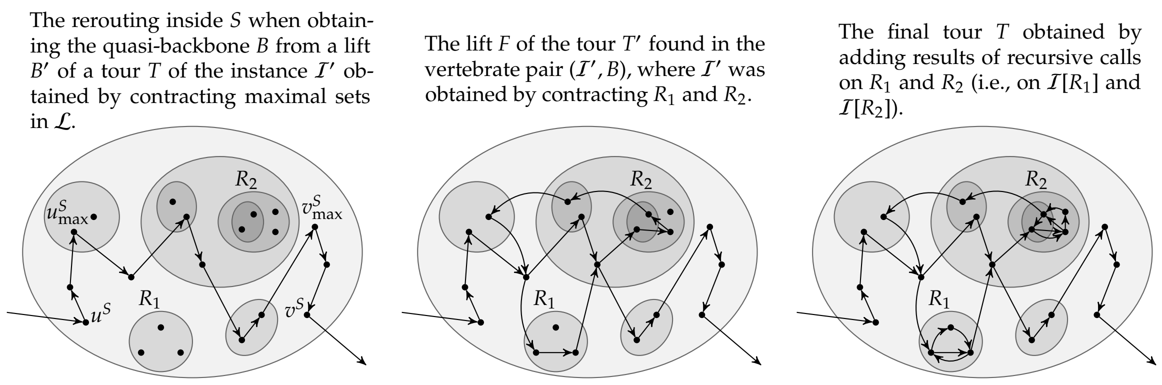
Now we recursively solve the contraction $\mathcal{I}/S$. (This is a smaller instance than $\mathcal{I}$, because $|\mathcal{I}/S| = |\mathcal{I}| - |S| + 1$ and $|S| \geqslant 2$ since a singleton set S would not have been reducible.) Let T be the tour obtained from the recursive call, and let F be the lift of T to $\mathcal{I}$. We finally return $F_S \cup F$. This is a tour of $\mathcal{I}$, as S is contractible with respect to F_S.

The running time of this algorithm is polynomial since each recursive call consists at most of: one call to $\mathcal{A}$, the algorithm of Lemma 7.9, simple graph operations and one recursive call (for a smaller instance).

Finally, let us show that this is a $\frac{2\rho}{1-\delta}$-approximation algorithm by induction on the instance size. We have

$$\begin{aligned} w_{\mathcal{I}}(F \cup F_S) &= w_{\mathcal{I}}(F) + w_{\mathcal{I}}(F_S) \\ &\leqslant w_{\mathcal{I}/S}(T) + w_{\mathcal{I}}(F_S) \\ &\leqslant \frac{2\rho}{1-\delta}\,\text{value}(\mathcal{I}/S) + 2\rho\,\text{value}(S) \\ &< \frac{2\rho}{1-\delta}\,[\text{value}(\mathcal{I}) - (1-\delta)\,\text{value}(S)] + 2\rho\,\text{value}(S) \\ &= \frac{2\rho}{1-\delta}\,\text{value}(\mathcal{I}), \end{aligned}$$

where the first inequality is by Lemma 7.5, the second follows since T is a $\frac{2\rho}{1-\delta}$-approximate solution for $\mathcal{I}/S$ and by (8.1), and the strict inequality is by Fact 7.3 and the reducibility of S. This shows that $F \cup F_S$ is a $\frac{2\rho}{1-\delta}$-approximate solution for $\mathcal{I}$. □


The rerouting inside S when obtaining the quasi-backbone B from a lift B' of a tour T of the instance $\mathcal{I}'$ obtained by contracting maximal sets in $\mathcal{L}$.

The lift F of the tour T' found in the vertebrate pair $(\mathcal{I}', B)$, where $\mathcal{I}'$ was obtained by contracting R_1 and R_2.

The final tour T obtained by adding results of recursive calls on R_1 and R_2 (i.e., on $\mathcal{I}[R_1]$ and $\mathcal{I}[R_2]$).

Figure 7: An illustration of the steps in the proofs of Lemma 9.3 (left) and Theorem 9.4 (center and right). Only one maximal set $S \in \mathcal{L}$ is shown.

9 Backbones and Reduction to Vertebrate Pairs

In this section we further reduce the task of approximating ATSP to that of finding a tour in instances with a *backbone*. For an example of such an instance see the right part of Figure 1 on page 6.

Definition 9.1. We say that an instance $\mathcal{I} = (G, \mathcal{L}, x, y)$ and a subtour B form a *vertebrate pair* if every $S \in \mathcal{L}$ with $|S| \geqslant 2$ is visited by B, i.e., $S \cap V(B) \neq \emptyset$. The set B is referred to as the *backbone* of the vertebrate pair.

The main result of this section, Theorem 9.4, takes as input an algorithm for vertebrate pairs that returns a tour with a weight bound depending on the backbone, and shows that this implies a constant-factor approximation algorithm for irreducible instances. Combining this with Theorem 8.3 allows us to reduce the problem of approximating ATSP on general instances to that of approximating ATSP on vertebrate pairs.

The proof of Theorem 9.4 is done in two steps. First, in Section 9.1, we give an efficient algorithm for finding a *quasi-backbone* B of an irreducible instance – a subtour that visits a large (weighted) fraction of the sets in $\mathcal{L}$. We use the term *quasi-backbone* as B might not visit *all* non-singleton sets, as would be required for a backbone. Then, in Section 9.2, we give the reduction to vertebrate pairs via a recursive algorithm (similar to the proof of Theorem 8.3 in the previous section).

9.1 Finding a Quasi-Backbone

We give an efficient algorithm for calculating a low-weight *quasi-backbone* of an irreducible instance.

Definition 9.2. For an instance $\mathcal{I} = (G, \mathcal{L}, x, y)$, we call a subtour B a *quasi-backbone* if

$$2 \sum_{S \in \mathcal{L}^*} y_S \leqslant (1 - \delta)\, \mathrm{value}(\mathcal{I})\,,$$

where $\mathcal{L}^* = \{S \in \mathcal{L} : S \cap V(B) = \emptyset\}$ contains those laminar sets that B does *not* visit.

Recall that $\delta = 0.78$ is the parameter in Definition 8.1 (of irreducible instances). Also note that a backbone is not necessarily a quasi-backbone, as it may not satisfy the above inequality if much y-value is on singleton sets. Recall that $\alpha_{\mathrm{S}} = 18 + \varepsilon$ denotes the approximation guarantee for singleton instances as in Corollary 5.2.

Lemma 9.3. *There is a polynomial-time algorithm that, given an irreducible instance $\mathcal{I} = (G, \mathcal{L}, x, y)$, constructs a quasi-backbone B such that $w(B) \leqslant (\alpha_{\mathrm{S}} + 3)\,\mathrm{value}(\mathcal{I})$.*

Proof. Let $\mathcal{L}_{\max}$ be the family of all maximal sets in $\mathcal{L}$. We define $\mathcal{I}'$ to be the instance obtained from $\mathcal{I}$ by contracting all sets in $\mathcal{L}_{\max}$; recall the definition of the contraction operation from Section 7.1. By Fact 7.3, the LP value does not increase, i.e., $\mathrm{value}(\mathcal{I}') \leqslant \mathrm{value}(\mathcal{I})$. In $\mathcal{I}'$, all sets in the laminar family are singletons, therefore the new instance is a singleton instance and we can use the α_{S}-approximation algorithm (Corollary 5.2) to find a tour T in $\mathcal{I}'$ with $w_{\mathcal{I}'}(T) \leqslant \alpha_{\mathrm{S}}\,\mathrm{value}(\mathcal{I}') \leqslant \alpha_{\mathrm{S}}\,\mathrm{value}(\mathcal{I})$.

Now, to obtain a subtour B of the original instance $\mathcal{I}$, we consider the lift B' of T back to $\mathcal{I}$ (see Definition 7.4). The lift B' is a subtour of low weight. Indeed, $w_{\mathcal{I}}(B') \leqslant w_{\mathcal{I}'}(T) \leqslant \alpha_{\mathrm{S}}\,\mathrm{value}(\mathcal{I})$ by Lemma 7.5. It also visits every maximal set $S \in \mathcal{L}_{\max}$. However, it might not yet satisfy the inequality of Definition 9.2. We therefore slightly modify the subtour B' to obtain B as follows. For each set $S \in \mathcal{L}_{\max}$:

1. Suppose the first visit[10] to S in the subtour B' arrives at a vertex $u^S \in S_{\mathrm{in}}$ and departs from a vertex $v^S \in S_{\mathrm{out}}$.
2. Replace the segment of B' from u^S to v^S by the union of:
   - a shortest path from u^S to $u^S_{\max}$,
   - a path from $u^S_{\max}$ to $v^S_{\max}$ inside S as given by Lemma 6.2,
   - and a shortest path from $v^S_{\max}$ to v^S,

   where $u^S_{\max} \in S_{\mathrm{in}}$ and $v^S_{\max} \in S_{\mathrm{out}}$ are selected to maximize $D_S(u^S_{\max}, v^S_{\max})$.

See the left part of Figure 7 for an illustration. The existence of the second path above is guaranteed by Lemma 6.1*(b)* since $u^S_{\max} \in S_{\mathrm{in}}$. It is clear that the obtained multiset B is a subtour (since B' is a subtour) and that the algorithm for finding B runs in polynomial time. It remains to bound the weight of B and to show that B satisfies the property of a quasi-backbone, i.e., the inequality of Definition 9.2.

For the former, note that the weight of B is at most the weight of the lift B' plus the weight of the three paths added for each set $S \in \mathcal{L}_{\max}$. For such a set $S \in \mathcal{L}_{\max}$, the weight of the path from u^S to $u^S_{\max}$ is at most $\mathrm{value}(S)$ since there is a path from $u^S \in S_{\mathrm{in}}$ to $u^S_{\max}$ *inside* S by Lemma 6.1*(b)* and such a path can be selected to have weight at most $\mathrm{value}(S)$ by Lemma 6.2. By the same argument, we have that the weight of the path from $v^S_{\max}$ to $v^S \in S_{\mathrm{out}}$ is at most $\mathrm{value}(S)$. Finally, by applying Lemma 6.2 again, we have that the weight of the path added from $u^S_{\max}$ to $v^S_{\max}$ is also bounded by $\mathrm{value}(S)$. It follows that

$$w(B) \leqslant w(B') + 3 \cdot \sum_{S \in \mathcal{L}_{\max}} \mathrm{value}(S) \leqslant w(B') + 3\,\mathrm{value}(\mathcal{I}) \leqslant (\alpha_{\mathrm{S}} + 3)\,\mathrm{value}(\mathcal{I}),$$

[10]Recall that the edges of the subtour B' are ordered by an Eulerian walk.

as required. (In the second inequality we used that the sets $S \in \mathcal{L}_{\max}$ are disjoint.)

We proceed to prove that B satisfies the inequality of Definition 9.2. Recall that $\mathcal{L}^* = \{S \in \mathcal{L} : S \cap V(B) = \emptyset\}$ contains those sets in $\mathcal{L}$ that B does not visit. As B visits every $S \in \mathcal{L}_{\max}$ (i.e., $\mathcal{L}_{\max} \cap \mathcal{L}^* = \emptyset$), it is enough to show the following:

Claim. For every $S \in \mathcal{L}_{\max}$ we have

$$\sum_{R\in\mathcal{L}^*:\, R\subsetneq S} 2y_R \leqslant (1-\delta)\,\text{value}(S)\,.$$

Once we have this claim, the property of a quasi-backbone indeed follows:

$$2\sum_{R\in\mathcal{L}^*} y_R = \sum_{S\in\mathcal{L}_{\max}} \sum_{R\in\mathcal{L}^*:\, R\subsetneq S} 2y_R \leqslant \sum_{S\in\mathcal{L}_{\max}} (1-\delta)\,\text{value}(S) \leqslant (1-\delta)\,\text{value}(\mathcal{I})\,.$$

Proof of Claim. The intuition behind the claim is that, when forming B, we have added a path P from $u^S_{\max}$ to $v^S_{\max}$. Since S is irreducible, this path P has a large weight. However, it is chosen so that it crosses each set in $\mathcal{L}$ at most twice. Thus it must cross most (weighted by value) sets of $\mathcal{L}$ contained in S.

Now we proceed with the formal proof. As $u^S_{\max} \in S_{\text{in}}$, the path P inside S from $u^S_{\max}$ to $v^S_{\max}$ that we have obtained from Lemma 6.2 crosses every tight set $R \in \mathcal{L}$ at most $2 - |R \cap \{u^S_{\max}, v^S_{\max}\}|$ times. Moreover, P (a subset of B) does not cross any set $R \in \mathcal{L}^*$. Therefore

$$d_S(u^S_{\max}, v^S_{\max}) \leqslant w(P) \leqslant \sum_{R\in\mathcal{L}\setminus\mathcal{L}^*:\, R\subsetneq S} \left(2 - |R\cap\{u^S_{\max}, v^S_{\max}\}|\right)\cdot y_R\,.$$

Furthermore, we have that the quasi-backbone B visits all sets in $\mathcal{L}_{\max}$ and visits both vertices $u^S_{\max}$ and $v^S_{\max}$. Therefore it must visit all sets $R \in \mathcal{L}$ for which $R \cap \{u^S_{\max}, v^S_{\max}\}$ is non-empty; i.e., for all $R \in \mathcal{L}^*$ we have $|R \cap \{u^S_{\max}, v^S_{\max}\}| = 0$. It follows that that the quasi-backbone visits most (weighted by value) laminar sets. If $u^S_{\max} = v^S_{\max}$, then we have

$$\sum_{R\in\mathcal{L}\setminus\mathcal{L}^*:\, R\subsetneq S} 2y_R = D_S(u^S_{\max}, v^S_{\max}),$$

and if $u^S_{\max} \neq v^S_{\max}$, then

$$\begin{aligned}\sum_{R\in\mathcal{L}\setminus\mathcal{L}^*:\, R\subsetneq S} 2y_R &= \sum_{R\in\mathcal{L}\setminus\mathcal{L}^*:\, R\subsetneq S} (2 - |R\cap\{u^S_{\max}, v^S_{\max}\}|)\cdot y_R + \sum_{R\in\mathcal{L}\setminus\mathcal{L}^*:\, R\subsetneq S} |R\cap\{u^S_{\max}, v^S_{\max}\}|\cdot y_R \\ &\geqslant d_S(u^S_{\max}, v^S_{\max}) + \sum_{R\in\mathcal{L}:\, R\subsetneq S} |R\cap\{u^S_{\max}, v^S_{\max}\}|\cdot y_R \\ &= D_S(u^S_{\max}, v^S_{\max})\end{aligned}$$

In both cases, we have $D_S(u^S_{\max}, v^S_{\max}) \geqslant \delta\,\text{value}(S)$ by the choice of $u^S_{\max}, v^S_{\max}$ and by the irreducibility of S. The claim now follows:

$$\sum_{R\in\mathcal{L}^*:\, R\subsetneq S} 2y_R = \text{value}(S) - \sum_{R\in\mathcal{L}\setminus\mathcal{L}^*:\, R\subsetneq S} 2y_R \leqslant \text{value}(S) - \delta\,\text{value}(S) = (1-\delta)\,\text{value}(S)\,.$$

◇

The proof of the above claim completes the proof of Lemma 9.3. □

9.2 Obtaining a Vertebrate Pair via Recursive Calls

We now prove the main result of Section 9. Recall the notation $\mathrm{lb}_{\mathcal{I}}(\bar{B})$ introduced in (5.1) on page 16.

Theorem 9.4. *Let $\mathcal{A}$ be a polynomial-time algorithm that, given a vertebrate pair $(\mathcal{I}', B)$, returns a tour of $\mathcal{I}'$ with weight at most*

$$\kappa\,\mathrm{value}(\mathcal{I}') + \eta\,\mathrm{lb}_{\mathcal{I}'}(\bar{B}) + w_{\mathcal{I}'}(B)$$

for some $\kappa, \eta \geqslant 0$. Then there is a polynomial-time ρ-approximation algorithm (with respect to the Held–Karp relaxation) for ATSP for irreducible instances, where

$$\rho = \frac{\kappa + \eta(1-\delta) + \alpha_S + 3}{2\delta - 1}.$$

The essence of the theorem is that if we have an algorithm for vertebrate pairs where the approximation factor is bounded by a constant factor of the value of the instance and the weight of the backbone, then this translates to a constant-factor approximation for ATSP in arbitrary irreducible instances (with no backbone given). The proof of this theorem is somewhat similar to that of Theorem 8.3, in that the algorithm presented here will call itself recursively on smaller instances, as well as invoking the black-box algorithm $\mathcal{A}$ (once per recursive call). The complicated dependence on the parameters is due to the recursive arguments. We will optimize the parameters κ and η in Section 11.

Proof. We briefly discuss the intuition first. Consider an irreducible instance $\mathcal{I} = (G, \mathcal{L}, x, y)$. By Lemma 9.3, we can find a quasi-backbone B – a subtour such that $2\sum_{S\in\mathcal{L}^*} y_S \leqslant (1-\delta)\,\mathrm{value}(\mathcal{I})$, where as before $\mathcal{L}^* = \{S \in \mathcal{L} : S \cap V(B) = \emptyset\}$ contains the laminar sets that the quasi-backbone does *not* visit. This is a small fraction of the entire optimum value($\mathcal{I}$), so we can afford to run an expensive procedure (say, a 2ρ-approximation) on the unvisited sets (using recursive calls) so as to make them contractible. Once we contract all these sets, B will become a backbone in the contracted instance and we will have thus obtained a vertebrate pair, on which the algorithm $\mathcal{A}$ can be applied.[11] See Figure 7 for an illustration.

We now formally describe the ρ-approximation algorithm $\mathcal{A}_{\mathrm{irr}}$ for irreducible instances. Given an irreducible instance $\mathcal{I} = (G, \mathcal{L}, x, y)$, it proceeds as follows:

1. Invoke the algorithm of Lemma 9.3 to obtain a quasi-backbone B with $w_{\mathcal{I}}(B) \leqslant (\alpha_S + 3)\,\mathrm{value}(\mathcal{I})$. Denote by $\mathcal{L}^*_{\max}$ the family of maximal (inclusion-wise) *non-singleton* sets in $\mathcal{L}^* = \{S \in \mathcal{L} : S \cap V(B) = \emptyset\}$. (For example, in Figure 7, R_1 and R_2 are two such sets.)

2. For each $S \in \mathcal{L}^*_{\max}$, recursively call $\mathcal{A}_{\mathrm{irr}}$ to find a tour T_S in the instance $\mathcal{I}[S]$ (which is irreducible by Fact 8.2). Then use T_S and the algorithm of Lemma 7.9 to find a collection F_S of subtours such that S is contractible with respect to F_S and $w_{\mathcal{I}}(F_S) \leqslant w_{\mathcal{I}[S]}(T_S)$.

[11] Note that we never actually find a backbone of the original, uncontracted instance.

3. Let $\mathcal{I}' = (G', \mathcal{L}', x', y')$ be the instance obtained from $\mathcal{I}$ by contracting all the maximal sets $S \in \mathcal{L}^*_{\max}$; let V' denote the contracted ground set. We have that $(\mathcal{I}', B)$ is a vertebrate pair by construction: we have contracted all sets in $\mathcal{L}$ that were not visited by B into single vertices, and so B is a backbone of $\mathcal{I}'$. Note that

$$\mathrm{lb}_{\mathcal{I}'}(\bar{B}) = 2 \sum_{v \in V' \setminus V(B)} y'_v \leqslant 2 \sum_{S \in \mathcal{L}^*} y_S \leqslant (1-\delta)\,\mathrm{value}(\mathcal{I}).$$

The first inequality follows by the definition of contraction, using Fact 7.1. We can invoke the algorithm $\mathcal{A}$ on the vertebrate pair $(\mathcal{I}', B)$; by the hypothesis of the theorem, it returns a tour T' of $\mathcal{I}'$ such that

$$w_{\mathcal{I}'}(T') \leqslant \kappa\,\mathrm{value}(\mathcal{I}') + \eta(1-\delta)\,\mathrm{value}(\mathcal{I}) + w_{\mathcal{I}'}(B). \tag{9.1}$$

4. Finally, return the tour T consisting of the lift F of T' to $\mathcal{I}$ together with $\bigcup_{S \in \mathcal{L}^*_{\max}} F_S$. (See the center and right parts of Figure 7 for an illustration.)

We remark that T is indeed a tour of $\mathcal{I}$ since all sets $S \in \mathcal{L}^*_{\max}$ are contractible with respect to $\bigcup_{S \in \mathcal{L}^*_{\max}} F_S$.

Having described the algorithm, it remains to show that $\mathcal{A}_{\mathrm{irr}}$ runs in polynomial time and that it has an approximation guarantee of ρ.

For the former, we bound the total number of recursive calls that $\mathcal{A}_{\mathrm{irr}}$ makes. We claim that the total number of recursive calls on input $\mathcal{I} = (G, \mathcal{L}, x, y)$ is at most the cardinality of $\mathcal{L}_{\geqslant 2} = \{S \in \mathcal{L} : |S| \geqslant 2\}$. The proof is by induction on $|\mathcal{L}_{\geqslant 2}|$. For the base case, i.e., when $|\mathcal{L}_{\geqslant 2}| = 0$, there are no recursive calls since there are no non-singleton sets in $\mathcal{L}^* \subseteq \mathcal{L}$ and so $\mathcal{L}^*_{\max} = \emptyset$. For the inductive step, suppose that $\mathcal{L}^*_{\max} = \{S_1, S_2, \ldots, S_\ell\} \subseteq \mathcal{L}^*$ and so there are ℓ recursive calls in this iteration – on the instances $\mathcal{I}[S_1], \mathcal{I}[S_2], \ldots, \mathcal{I}[S_\ell]$. If we let $\mathcal{L}^i_{\geqslant 2}$ denote the non-singleton laminar sets of $\mathcal{I}[S_i]$ then, by the definition of inducing on a tight set, for every $R \in \mathcal{L}^i_{\geqslant 2}$ we have $R \subsetneq S_i$ and $R \in \mathcal{L}_{\geqslant 2}$. It follows by the induction hypothesis that the total number of recursive calls that $\mathcal{A}_{\mathrm{irr}}$ makes is

$$\ell + \sum_{i=1}^{\ell} |\mathcal{L}^i_{\geqslant 2}| \leqslant \ell + |\mathcal{L}_{\geqslant 2}| - \ell = |\mathcal{L}_{\geqslant 2}|\,,$$

where the inequality holds because the sets $\mathcal{L}^i_{\geqslant 2}$ are disjoint and $\mathcal{L}^1_{\geqslant 2} \cup \mathcal{L}^2_{\geqslant 2} \cup \cdots \cup \mathcal{L}^\ell_{\geqslant 2} \subseteq \mathcal{L}_{\geqslant 2} \setminus \{S_1, S_2, \ldots, S_\ell\}$. Hence, the total number of recursive calls $\mathcal{A}_{\mathrm{irr}}$ makes is $|\mathcal{L}_{\geqslant 2}| \leqslant |\mathcal{L}|$, which is at most linear in $|V|$. The fact that $\mathcal{A}_{\mathrm{irr}}$ runs in polynomial time now follows because each call runs in polynomial time. Indeed, the algorithm of Lemma 9.3, the algorithm of Lemma 7.9, and $\mathcal{A}$ all run in polynomial time.

We now complete the proof of the theorem by showing that $\mathcal{A}_{\mathrm{irr}}$ is a ρ-approximation algorithm. From (9.1) and by Lemma 7.5 we have that the weight $w_{\mathcal{I}}(F)$ of the lift F of T' is at most

$$w_{\mathcal{I}'}(T') \leqslant \kappa\,\mathrm{value}(\mathcal{I}') + \eta(1-\delta)\,\mathrm{value}(\mathcal{I}) + w_{\mathcal{I}'}(B) \leqslant (\kappa + \eta(1-\delta) + \alpha_{\mathrm{S}} + 3)\,\mathrm{value}(\mathcal{I}),$$

where the second inequality follows by Fact 7.3 and since $w_{\mathcal{I}'}(B) = w_{\mathcal{I}}(B)$ ($\mathcal{I}'$ arises by contracting only sets not visited by B, which preserves the weight of B) and $w_{\mathcal{I}}(B) \leqslant (\alpha_{\mathrm{S}} + 3)\,\mathrm{value}(\mathcal{I})$.

Now, to show that $w(T) = w(F) + w\left(\bigcup_{S\in\mathcal{L}^*_{\max}} F_S\right) \leqslant \rho\,\text{value}(\mathcal{I})$, we proceed by induction on the total number of recursive calls. In the base case, when no recursive calls are made, we have $w(T) = w(F) \leqslant w_{\mathcal{I}'}(T') \leqslant (\kappa + \eta(1-\delta) + \alpha_{\mathrm{S}} + 3)\,\text{value}(\mathcal{I}) \leqslant \rho\,\text{value}(\mathcal{I})$. For the inductive step, the induction hypothesis yields that for each $S \in \mathcal{L}^*_{\max}$ we have

$$w(F_S) \leqslant w_{\mathcal{I}[S]}(T_S) \leqslant \rho\,\text{value}(\mathcal{I}[S]) = 2\rho\,\text{value}(S)\,,$$

where the equality is by Fact 7.7. Hence

$$\begin{aligned} w\left(\bigcup_{S\in\mathcal{L}^*_{\max}} F_S\right) &= \sum_{S\in\mathcal{L}^*_{\max}} w(F_S) \leqslant \sum_{S\in\mathcal{L}^*_{\max}} 2\rho\,\text{value}(S) \\ &= \sum_{S\in\mathcal{L}^*_{\max}} 2\rho \sum_{R\in\mathcal{L}^*:\, R\subsetneq S} 2y_R \leqslant 2\rho \sum_{R\in\mathcal{L}^*} 2y_R \leqslant 2\rho(1-\delta)\,\text{value}(\mathcal{I})\,. \end{aligned}$$

The second equality uses that $\text{value}(S) = \sum_{R\in\mathcal{L}:\, R\subsetneq S} 2y_R = \sum_{R\in\mathcal{L}^*:\, R\subsetneq S} 2y_R$ for all $S \in \mathcal{L}^*_{\max}$: as $S \cap V(B) = \emptyset$, any $R \in \mathcal{L}$ with $R \subsetneq S$ also has $R \cap V(B) = \emptyset$ and thus $R \in \mathcal{L}^*$. The last inequality holds because B is a quasi-backbone of $\mathcal{I}$ (see Definition 9.2). Summing up the weight of the lift F of T' and of $\bigcup_{S\in\mathcal{L}^*_{\max}} F_S$ we get

$$w(T) \leqslant (\kappa + \eta(1-\delta) + \alpha_{\mathrm{S}} + 3 + 2\rho(1-\delta))\,\text{value}(\mathcal{I}) = \rho\,\text{value}(\mathcal{I})\,,$$

by the selection of ρ to equal $(\kappa + \eta(1-\delta) + \alpha_{\mathrm{S}} + 3)/(2\delta - 1)$. This concludes the inductive step and the proof of the theorem. □

Part III

Solving Subtour Partition Cover

In this part we solve Subtour Partition Cover on vertebrate pairs. Recall the definition of Subtour Partition Cover given in Section 3: we are given an instance $\mathcal{I} = (G, \mathcal{L}, x, y)$ with a subtour B in G, and a partition $(V_1, V_2, \dots, V_k)$ of $V \setminus V(B)$, where each V_i is strongly connected. Our goal is to find a collection F of subtours of E crossing each of the sets V_i, and satisfying certain "local" and "global" cost bounds. The main technical concept in the argument is that of *witness flows*. On a high level, we want every subtour T in our solution to Subtour Partition Cover to be forced to intersect the backbone B if T crosses a non-singleton set in the laminar family $\mathcal{L}$. Every subtour that does not cross any such set locally behaves as in a singleton instance with regard to its cost, and it is easy to account for those subtours. On the other hand, we are able to take care of the cost of all subtours that do cross some such set (and thus also intersect B), together with B, using a global cost argument. The witness flow is a tool that allows us to enforce this crucial property in our solution to Subtour Partition Cover. It is inspired by a general method of ensuring connectivity in integer/linear programming formulations for graph problems, which requires the existence of a flow (supported on the LP solution) between the pairs of vertices that should be connected.

We note that witness flows used in this paper can be seen as a more concise variant of a previous argument using split graphs in the conference version [STV18a]. Split graphs have been first used in a similar role in [STV18b].

By the reductions in the previous parts, this is sufficient for obtaining a constant-factor approximation algorithm. We combine all the ingredients and calculate the obtained ratio in Section 11.

10 Algorithm for Vertebrate Pairs

In this section we consider a vertebrate pair $(\mathcal{I}, B)$ and prove the following theorem and corollary. This provides the algorithm required in Theorem 9.4.

Theorem 10.1. *There exists a* $(4, 2\,\text{value}(\mathcal{I}) + \text{lb}_{\mathcal{I}}(\bar{B}))$*-light algorithm for Subtour Partition Cover for vertebrate pairs* $(\mathcal{I}, B)$*.*

(Refer to Definition 3.1 of a light algorithm on page 13.) Combined with Theorem 5.1, we immediately obtain the following.

Corollary 10.2. *For every* $\varepsilon > 0$ *there is a polynomial-time algorithm that, given a vertebrate pair* $(\mathcal{I}, B)$*, returns a tour* T *of* $\mathcal{I}$ *with* $w(T) \leqslant 2\,\text{value}(\mathcal{I}) + (37 + 36\varepsilon)\,\text{lb}_{\mathcal{I}}(\bar{B}) + w(B)$*.*

Throughout this section we will assume that $B \neq \emptyset$. In the special case when $B = \emptyset$, it must be the case that $\mathcal{L}_{\geqslant 2} = \emptyset$ and thus the instance is singleton; in that case, we simply apply the strictly better $(2, 0)$-light algorithm of Theorem 4.1.

We now formulate our main technical lemma. Let $\mathcal{L}_{\geqslant 2}$ denote the family of non-singleton sets in $\mathcal{L}$.

Lemma 10.3. *There is a polynomial-time algorithm that solves the following problem. Let* $(\mathcal{I}, B)$ *be a vertebrate pair, and let* $U_1, \dots, U_\ell \subseteq V \setminus V(B)$ *be disjoint non-empty vertex sets such that the subgraphs* $G[U_1], \dots, G[U_\ell]$ *are strongly connected and for every* $S \in \mathcal{L}_{\geqslant 2}$ *and* $i = 1, \dots, \ell$ *we have either* $U_i \cap S = \emptyset$ *or* $U_i \subseteq S$*. Then the algorithm finds a collection of subtours* $F \subseteq E$ *such that:*

(a) $w(F) \leqslant 2\,\text{value}(\mathcal{I}) + \text{lb}_{\mathcal{I}}(\bar{B})$,

(b) $|\delta^-_F(U_i)| \geqslant 1$ *for every* $i = 1, \dots, \ell$,

(c) $|\delta^-_F(v)| \leqslant 4$ *whenever* $x(\delta^-(v)) = 1$,

(d) *any subtour in* F *that crosses a set in* $\mathcal{L}_{\geqslant 2}$ *visits a vertex of the backbone.*

Notice that the requirements on the disjoint sets $U_1, \dots, U_\ell$ imply that $\mathcal{L}_{\geqslant 2} \cup \{U_1, \dots, U_\ell\}$ is a laminar family in which the sets $U_1, \dots, U_\ell$ are minimal (see the left part of Figure 8). We also remark that property *(d)* will be important for analyzing the lightness of our tour. Indeed, it will imply that any subtour in our solution $F^\star$ to Subtour Partition Cover that is disjoint from the backbone does not cross a set in $\mathcal{L}_{\geqslant 2}$. Thus any edge (u, v) in such a subtour will have weight equal to $y_u + y_v$. Intuitively, this (almost) reduces the problem to the singleton case.

The proof will be given in Section 10.1, using the concept of *witness flows* that allow us to enforce the crucial property *(d)*.

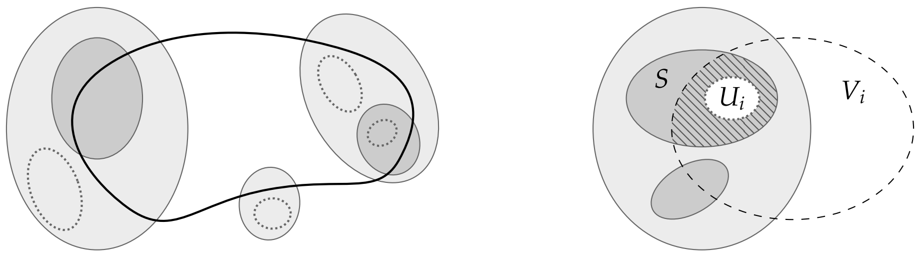


Figure 8: On the left, the "dotted" sets $U_1, \dots, U_\ell$ of Lemma 10.3 are depicted. On the right, we show how the set U_i is obtained by the algorithm for Subtour Partition Cover in the proof of Theorem 10.1: V_i is intersected with a minimal non-singleton set S to obtain V'_i (the striped area). Then, U_i is a source component in the decomposition of V'_i into strongly connected components. This implies that there are no edges from $V'_i \setminus U_i$ to U_i and so any edge in $\delta(V_i \setminus U_i, U_i)$ must come from outside of V'_i and thus cross the tight set S.

Proof of Theorem 10.1. Let $(V_1, V_2, \dots, V_k)$ be the input partition of $V \setminus V(B)$ in the Subtour Partition Cover problem. We will apply Lemma 10.3 for a collection $(U_1, U_2, \dots, U_k)$ of disjoint subsets with $U_i \subseteq V_i$, defined as follows. For $i = 1, \dots, k$, let V'_i be the intersection of V_i with a minimal set $S \in \mathcal{L}_{\geqslant 2} \cup \{V\}$ with $S \cap V_i \neq \emptyset$. Then consider a decomposition of V'_i into strongly connected components (with respect to $G[V'_i]$). Let $U_i \subseteq V'_i$ be the vertex set of a source component in this decomposition. That is, there is no edge from $V'_i \setminus U_i$ to U_i in G (see also the right part of Figure 8). By construction, the sets $U_1, \dots, U_k$ satisfy the conditions of Lemma 10.3; in particular, since V'_i (and thus U_i) is a subset of the minimal set S chosen above, it follows that $\mathcal{L}_{\geqslant 2} \cup \{U_i\}$ is a laminar family, and there are no subsets $S' \subsetneq U_i$, $S' \in \mathcal{L}_{\geqslant 2}$. We let F be the Eulerian multiset guaranteed by Lemma 10.3.

The rest of the proof is dedicated to showing that F satisfies the requirement of an $(4, 2\,\mathrm{value}(\mathcal{I}) + \mathrm{lb}_{\mathcal{I}}(\bar{B}))$-light algorithm. Let us start with the connectivity requirement.

Claim 10.4. We have $|\delta^-_F(V_i)| \geqslant 1$ for $i = 1, 2, \dots, k$.

Proof. By property *(b)* of Lemma 10.3, there exists an edge $e \in \delta^-_F(U_i)$. Then either $e \in \delta^-_F(V_i)$ (in which case we are done), or $e \in \delta_F(V_i \setminus U_i, U_i)$. Assume the latter case.

Using that U_i was a source component in the decomposition of V'_i into strongly connected components, e must enter a set in $\mathcal{L}_{\geqslant 2}$. Indeed, recall that U_i was selected so that there is no edge from $V'_i \setminus U_i$ to U_i. Since $e \in \delta_F(V_i \setminus U_i, U_i)$ and $\delta(V'_i \setminus U_i, U_i) = \emptyset$, we must have $e \in \delta(V_i \setminus V'_i, U_i) \subseteq \delta(V_i \setminus V'_i, V'_i)$. However, V'_i was obtained by intersecting V_i with a minimal set $S \in \mathcal{L}_{\geqslant 2} \cup \{V\}$ with $S \cap V_i \neq \emptyset$. Thus we must have $e \in \delta^-_F(S)$ (and $S \neq V$). Now, property *(d)* of Lemma 10.3 guarantees that the connected component (i.e., the subtour) of F containing e must visit $V(B)$. This subtour thus visits both V_i (the head of e is in $U_i \subseteq V_i$) and $V(B)$, which is disjoint from V_i. As such, the subtour must cross V_i, i.e., we have $|\delta^-_F(V_i)| \geqslant 1$ as required. □

Next, let us consider subtours in F that are disjoint from B.

Claim 10.5. Let T be a subtour in F with $V(T) \cap V(B) = \emptyset$. Then $w(T) \leqslant 4\,\mathrm{lb}(T)$.

Proof. Recall that the lower bound is

$$\mathrm{lb}(T) = 2 \sum_{v \in V(T)} y_v \,.$$

To bound the weight of T, note that by property *(d)* of Lemma 10.3, the edges of T do not cross any tight set in $\mathcal{L}_{\geqslant 2}$. Therefore any edge (u, v) in T has weight $y_u + y_v$ and so

$$w(T) = \sum_{e \in T} w(e) = \sum_{v \in V(T)} |\delta_F(v)| y_v \leqslant 8 \sum_{v \in V(T)} y_v = 4\,\mathrm{lb}(T)\,,$$

where for the inequality we used that y_v is only strictly positive if $x(\delta^-(v)) = 1$ (see Definition 2.3), in which case $|\delta_F(v)| = 2|\delta_F^-(v)| \leqslant 8$ using property *(c)* of Lemma 10.3. □

Finally, let $F_B \subseteq F$ be the collection of subtours in F that intersect B. Then, $w(F_B) \leqslant w(F) \leqslant 2\,\mathrm{value}(\mathcal{I}) + \mathrm{lb}_{\mathcal{I}}(\bar{B})$ by property *(a)* of Lemma 10.3. This completes the proof that F is a $(4, 2\,\mathrm{value}(\mathcal{I}) + \mathrm{lb}_{\mathcal{I}}(\bar{B}))$-light edge set. □

The rest of this section is devoted to the proof of the main technical Lemma 10.3.

10.1 Witness flows

Recall that $\mathcal{L}_{\geqslant 2}$ denotes the family of non-singleton sets in $\mathcal{L}$. Let us use an indexing $\mathcal{L}_{\geqslant 2} \cup \{V\} = \{S_1, S_2, \ldots, S_\ell\}$ such that $2 \leqslant |S_1| \leqslant |S_2| \leqslant \cdots \leqslant |S_\ell| = |V|$. For a vertex $v \in V$ let

$$\mathrm{level}(v) = \min\{i : v \in S_i\}$$

be the index of the first (smallest) set that contains v. We use these levels to define a partial order $\prec$ on the vertices: let $v \prec v'$ if $\mathrm{level}(v) < \mathrm{level}(v')$. This partial order is used to classify the edges as follows. An edge $(u, v) \in E$ is a

– *forward* edge if $v \prec u$,

– *backward* edge if $u \prec v$,

and otherwise it is a *neutral* edge. Let E_f, E_b and E_n denote the sets of forward, backward, and neutral edges respectively.

Definition 10.6. Let $z : E \to \mathbb{R}$ be a circulation. We say that $f : E \to \mathbb{R}$ is a *witness flow* for z if

(a) $0 \leqslant f \leqslant z$,

(b) $f(\delta^+(v)) \geqslant f(\delta^-(v))$ for every $v \in V \setminus V(B)$,

(c) $f(e) = 0$ for each backward edge $e \in E_b$,

(d) $f(e) = z(e)$ for each forward edge $e \in E_f$.

We say that a circulation z is *witnessed* if there exists a witness flow for z.

The following lemma reveals the importance of witness flows. By a component of a circulation z we mean a connected component of the edge set $\mathrm{supp}(z) = \{e \in E : z_e > 0\}$.

Lemma 10.7. *Let z be a witnessed circulation. Any component C of z that crosses a set in $\mathcal{L}_{\geqslant 2}$ must intersect B.*

Proof. Let f be a witness flow for z. Take i to be the smallest value such that $S_i \cap V(C) \neq \emptyset$. Then we must also have $V(C) \setminus S_i \neq \emptyset$, as otherwise $\mathcal{L}_{\geqslant 2} \cup \{V(C)\}$ would be laminar, contradicting the choice of C. So, C must have an edge entering S_i; moreover, all edges of C entering S_i are forward edges, and all edges of C exiting S_i are backward edges. Let $U = S_i \cap V(C)$. Then $f(\delta^-(U)) > 0$ and $f(\delta^+(U)) = 0$. If we had $U \cap V(B) = \emptyset$, then *(b)* would imply that

$$0 > f(\delta^+(U)) - f(\delta^-(U)) = \sum_{v \in U} f(\delta^+(v)) - f(\delta^-(v)) \geqslant 0\,,$$

a contradiction. □

The edge set F in Lemma 10.3 will be obtained from a witnessed integer circulation; in particular, we will use the witness flow to derive property *(d)* in Lemma 10.3. We start by showing that the Held-Karp solution x is a witnessed circulation.

Lemma 10.8. *The Held-Karp solution x is a witnessed circulation.*

Before giving the full proof, let us motivate the existence of a witness flow f in a simple example scenario where there is only one non-singleton set $S \in \mathcal{L}_{\geqslant 2}$. Then we have $E_f = \delta^-(S)$ and $E_b = \delta^+(S)$, i.e., the forward/backward edges are exactly the incoming/outgoing edges of S. The subtour elimination constraints imply (via the min-cut max-flow theorem) that x supports a unit flow between any pair of vertices. Let f be such a flow from any vertex outside S to a vertex $v \in S \cap V(B)$ (such a v exists by the backbone property). It is easy to see that f satisfies the conditions of the claim. Indeed, since S is a tight set, f saturates all incoming (forward) edges. It also does not leave S, i.e., use any backward edges. The proof of the general case uses an argument based on LP duality to argue the existence of f.

Proof of Lemma 10.8. We find a witness flow f in polynomial time by solving the following linear program:

$$\begin{array}{lll} \text{maximize} & \displaystyle\sum_{e \in E_f} f(e) & \\ \text{subject to} & f(\delta^+(v)) \geqslant f(\delta^-(v)) & \text{for } v \in V \setminus V(B), \\ & f(e) = 0 & \text{for } e \in E_b, \\ & 0 \leqslant f \leqslant x. & \end{array}$$

By the constraints of the linear program, we have that f satisfies *(a)*, *(b)*, and *(c)*. It remains to verify *(d)*, or equivalently, to show that the optimum value of this program equals $x(E_f)$.

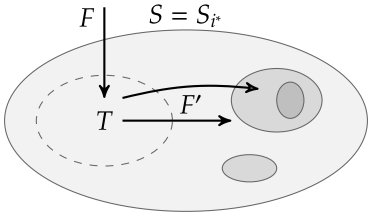
This will be shown using the dual linear program. The variables $(\pi_v)_{v \in V}$ correspond to the first set of constraints, and $(z(e))_{e \in E_f \cup E_n}$ to the capacity constraints on forward and neutral edges. No such variables are needed for backward edges. For notational simplicity, we introduce π_v also for $v \in V(B)$, and set $\pi_v = 0$ in this case. The dual program can be written as follows.

$$\begin{array}{lrl}
\text{minimize} & \displaystyle\sum_{e \in E_f \cup E_n} x(e)z(e) & \\
\text{subject to} & \pi_v - \pi_u + z(u,v) \geqslant 1 & \text{for } (u,v) \in E_f, \\
& \pi_v - \pi_u + z(u,v) \geqslant 0 & \text{for } (u,v) \in E_n, \\
& \pi_v = 0 & \text{for } v \in V(B), \\
& \pi, z \geqslant 0. &
\end{array}$$

Note that setting $\pi = \mathbf{0}$, $z(e) = 1$ if $e \in E_f$, and $z(e) = 0$ if $e \in E_n$ is a feasible solution with objective value $x(E_f)$. We complete the proof by showing that this is an optimal dual solution.

Let us select a dual optimal solution (π, z) that minimizes $\pi(V)$. We show that this minimum value is 0, that is, there exists a dual optimal solution with $\pi = \mathbf{0}$. This immediately implies that the above solution is a dual optimal one, because given $\pi = \mathbf{0}$ we get a constraint $z(u,v) \geqslant 1$ for all $(u,v) \in E_f$, making the objective value at least $x(E_f)$.

Towards a contradiction, assume that π is not everywhere zero. Let us select a vertex $t \in V$ such that $\pi_t > 0$, and $\mathrm{level}(t) = i^*$ is minimal among all such vertices. Let $S = S_{i^*}$, and define $T = \{u \in S : \pi_u > 0\}$. Since $S \cap V(B) \neq \emptyset$, and $\pi_u = 0$ for all $u \in V(B)$, we see that T is a proper subset of S. Let $F = \delta(V \setminus S, T)$ and $F' = \delta(T, S \setminus T)$. A depiction of the sets is as follows:



Let us show that the edges between T and $S \setminus T$ can be only of certain types.

Claim. $F' = \delta(T, S \setminus T) \subseteq E_f \cup E_n$ and $\delta(S \setminus T, T) \subseteq E_b \cup E_n$.

Proof of Claim. By the choice of i^*, for any $S_i \subsetneq S$ we have that $\pi_v = 0$ for all $v \in S_i$; thus $S_i \cap T = \emptyset$, and so for every $u \in T$ we have $\mathrm{level}(u) = i^*$. Therefore, for every $(u,v) \in F'$, we must have $\mathrm{level}(u) = i^* \geqslant \mathrm{level}(v)$, and for every $(u,v) \in \delta(S \setminus T, T)$, $\mathrm{level}(v) = i^* \geqslant \mathrm{level}(u)$. ◇

Let us construct another dual solution (π', z') as follows. Let $\varepsilon = \min\{\pi_u : u \in T\}$, and let

$$\pi'_u = \begin{cases} \pi_u - \varepsilon & \text{if } u \in T, \\ \pi_u & \text{otherwise,} \end{cases} \quad \text{and} \quad z'(e) = \begin{cases} z(e) + \varepsilon & \text{if } e \in F \cap (E_f \cup E_n), \\ z(e) - \varepsilon & \text{if } e \in F', \\ z(e) & \text{otherwise.} \end{cases}$$

We show that (π', z') is another optimal solution. Then, $\pi'(V) < \pi(V)$ gives a contradiction to the choice of (π, z). The proof proceeds in two steps: first we show feasibility and then optimality.

Feasibility. We have $\pi' \geqslant 0$ by the choice of ε. To show $z' \geqslant 0$, note that for $e = (u,v) \in F'$, by the Claim and the dual constraints for $e \in E_f \cup E_n$ we have $0 \leqslant \pi_v - \pi_u + z(u,v) \leqslant z(u,v) - \varepsilon$, where for the second inequality we used that $\pi_u \geqslant \varepsilon$ and $\pi_v = 0$. Thus, $z(e) \geqslant \varepsilon$ and $z'(e) = z(e) - \varepsilon \geqslant 0$ for every $e \in F'$. For an edge $e \notin F'$, we have $z'(e) \geqslant z(e) \geqslant 0$ and so we can conclude that $z' \geqslant 0$. We also have $\pi'_v = 0$ for $v \in V(B)$ since $T \cap V(B) = \emptyset$.

It remains to verify the constraints for $(u,v) \in E_f \cup E_n$. For every $(u,v) \in (F \cup F') \cap (E_f \cup E_n)$, as well as for edges not in $\delta(T)$, we have $\pi'_v - \pi'_u + z'(u,v) = \pi_v - \pi_u + z(u,v)$, and therefore the constraint remains valid. Thus we may have $\pi'_v - \pi'_u + z'(u,v) \neq \pi_v - \pi_u + z(u,v)$ in only two cases: either **(i)** if $(u,v) \in \delta(T, V \setminus S)$, or **(ii)** if $(u,v) \in \delta(S \setminus T, T)$.

In case **(i)**, the constraint on (u,v) remains valid since $\pi'_v - \pi'_u + z'(u,v) = \pi_v - \pi_u + z(u,v) + \varepsilon$. In case **(ii)**, $\pi'_u = 0$, $\pi'_v \geqslant 0$, and $z'(u,v) \geqslant 0$. Further, we have shown in the above Claim that $(u,v) \in E_b \cup E_n$. There is no constraint for $(u,v) \in E_b$, and $\pi'_v - \pi'_u + z'(u,v) \geqslant 0$ holds if $(u,v) \in E_n$.

Optimality. When changing (π, z) to (π', z'), the objective value increases by $\varepsilon(x(F \cap (E_f \cup E_n)) - x(F' \cap (E_f \cup E_n))) = \varepsilon(x(F \setminus E_b) - x(F'))$; we will show that this is non-positive. Recall that S is either a tight set or V, so that $x(\delta^-(S)) \leqslant 1$. Since $T \subsetneq S$, we have $1 \leqslant x(\delta^-(S \setminus T)) = x(\delta^-(S)) - x(F) + x(F') \leqslant 1 - x(F) + x(F')$, and therefore $x(F') \geqslant x(F) \geqslant x(F \setminus E_b)$. Thus, the objective value does not increase, therefore (π', z') must be optimal.

The existence of the optimal solution (π', z') with $\pi'(V) < \pi(V)$ contradicts the choice of (π, z), which completes the proof of Lemma 10.8. $\square$

Let us state one more lemma that enables rounding fractional witness flows to integer ones. Recall that in the proof of Theorem 4.1 for singleton instances a key step was to round a fractional circulation to an integer one, a simple corollary of the integrality of the network flow polyhedron. We will now need a stronger statement as we need to round a circulation z along with a witness flow f consistently. We formulate the following general statement (which does not assume that we have a vertebrate pair or that f is a witness flow).

Lemma 10.9. *For a digraph $G = (V, E)$ and edge weights $w : E \to \mathbb{R}$, consider $z, f : E \to \mathbb{R}_+$ such that z is a circulation and $f \leqslant z$. Then there exist integer-valued vectors $\bar{z}, \bar{f} : E \to \mathbb{Z}_+$ with $w^\top \bar{z} \leqslant w^\top z$ satisfying the following properties:*

(a) *$\bar{z}$ is a circulation,*

(b) *$\bar{f}(\delta^+(v)) \geqslant \bar{f}(\delta^-(v))$ whenever $f(\delta^+(v)) \geqslant f(\delta^-(v))$,*

(c) *$\lfloor f(\delta^-(v)) \rfloor \leqslant \bar{f}(\delta^-(v)) \leqslant \lceil f(\delta^-(v)) \rceil$ for every node $v \in V$,*

(d) *$\lfloor g(\delta^-(v)) \rfloor \leqslant \bar{g}(\delta^-(v)) \leqslant \lceil g(\delta^-(v)) \rceil$ for every node $v \in V$, where $g = z - f$ and $\bar{g} = \bar{z} - \bar{f}$.*

(e) *$\bar{f} \leqslant \bar{z}$; $\bar{f}_e = \bar{z}_e$ whenever $f_e = z_e$, and $\bar{f}_e = 0$ whenever $f_e = 0$.*

The proof relies on the total unimodularity of network matrices. Let (V, E) be a directed graph and $T = (V, E_T)$ a directed tree on the same node set. These define a *network matrix* $B \in \mathbb{Z}^{|E_T| \times |E|}$ as follows. For every $e = (u, v) \in E$, let P_e be the unique undirected $u - v$ path in the tree T. Then we set $B_{e_T,e} = 1$ if e_T occurs in forward direction in P_e, $B_{e_T,e} = -1$ if e_T occurs in the backward direction in P_e, and $B_{e_T,e} = 0$ if e_T does not occur in P_e. We will use the following result (see e.g. [Sch98, (34) in Sec 19.3]):

Theorem 10.10 ([Tut65]). *Every network matrix is totally unimodular.*

Together with [Sch98, Theorem 19.1 and (4) in Sec 19.1], it follows that any LP of the form $\min\{c^\top x : b^{\text{lower}} \leqslant Bx \leqslant b^{\text{upper}}, \ell \leqslant x \leqslant u\}$ has an integer optimal solution for any integer vectors $b^{\text{lower}}, b^{\text{upper}} \in \mathbb{Z}^{|E_T|}$ and $\ell, u \in \mathbb{Z}^{|E|}$. A fundamental example of network matrices is the incidence matrix of a directed graph $G = (V, E)$. Consider the digraph $(V \cup \{r\}, E)$ obtained by adding a new vertex r and the tree $T = (V \cup \{r\}, \{(r, u) : u \in V\})$ that forms a star. The associated network matrix is identical to the incidence matrix of G. The proof below extends this simple construction to capture all degree requirements.

Proof of Lemma 10.9. Let us introduce new variables $\bar{g} = \bar{z} - \bar{f}$ (the notation already used in property (d)). The integrality of $\bar{z}$ is equivalent to the integrality of $\bar{f}$ and $\bar{g}$.

Let us construct a network matrix $B \in \mathbb{Z}^{4|V| \times 2|E|}$ as follows. We define a tree $T = (V', E_T)$, where V' contains a root node r, and for each $v \in V$, we include four nodes $v^g, v^f, v^{g'}, v^{f'}$ and four edges (r, v^g), (v^g, v^f), $(v^f, v^{f'})$ and $(v^g, v^{g'})$ in E_T. Let us define the directed graph (V', E'), where $E' = E'_f \cup E'_g$ is obtained as follows: for each $(u, v) \in E$, we add an edge $(u^f, v^{f'})$ to E'_f and an edge $(u^g, v^{g'})$ to E'_g. Let B be the network matrix corresponding to (V', E') and T (see Figure 9 for an example). That is, B is an $|E_T| \times |E'|$ matrix such that

– the column corresponding to $(u^f, v^{f'}) \in E'_f$ has a -1 entry in the rows corresponding to the edges (r, u^g) and (u^g, u^f), and $+1$ entries corresponding to the edges (r, v^g), (v^g, v^f), and $(v^f, v^{f'})$; all other entries are 0;

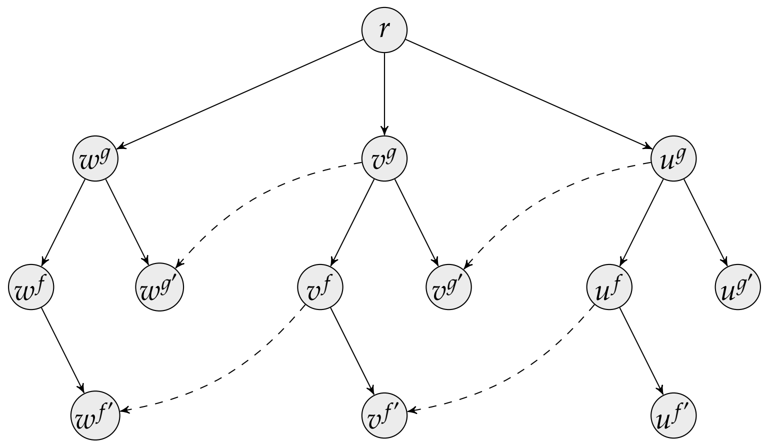
– the column corresponding to $(u^g, v^{g'}) \in E'_g$ has a -1 entry in the row corresponding to the edge (r, u^g), and $+1$ entries corresponding to the edges (r, v^g) and $(v^g, v^{g'})$; all other entries are 0.

Then B is a totally unimodular matrix according to Theorem 10.10. Using the variables $\bar{f}$ and $\bar{g}$, the system (a)-(e) can be equivalently written in the form

$$b^{\text{lower}} \leqslant B(\bar{f}, \bar{g}) \leqslant b^{\text{upper}}$$
$$d^{\text{lower}} \leqslant (\bar{f}, \bar{g}) \leqslant d^{\text{upper}},$$

where $b^{\text{lower}}, b^{\text{upper}}, d^{\text{lower}}, d^{\text{upper}}$ are vectors with integer or infinite entries. (Here, $(\bar{f}, \bar{g}) \in \mathbb{R}^{|E|+|E|}$ is a column vector, and $B(\bar{f}, \bar{g})$ the matrix-vector product.) Namely, the variable $\bar{f}(u, v)$ corresponds to the column for $(u^f, v^{f'}) \in E'_f$, and the variable $\bar{g}(u, v)$ corresponds to the column for $(u^g, v^{g'}) \in E'_g$. See also Figure 9.

– The rows for $(r, v^g) \in E_T$ describe the flow conservation constraints as in (a) with $b^{\text{lower}}(r, v^g) = b^{\text{upper}}(r, v^g) = 0$.



part of v's:	inc. f-flow $(u^f, v^{f'})$	inc. g-flow $(u^g, v^{g'})$	out. f-flow $(v^f, w^{f'})$	out. g-flow $(v^g, w^{g'})$	corresponds to v's constraint:
(r, v^g)	1	1	-1	-1	(a)
(v^g, v^f)	1	0	-1	0	(b)
$(v^f, v^{f'})$	1	0	0	0	(c)
$(v^g, v^{g'})$	0	1	0	0	(d)
(r, u^g)	-1	-1	0	0	
(u^g, u^f)	-1	0	0	0	
$(u^f, u^{f'})$	0	0	0	0	
$(u^g, u^{g'})$	0	0	0	0	

Figure 9: The network matrix in the proof of Lemma 10.9.
Top: fragment of the tree $T = (V', E_T)$ corresponding to three vertices u, v, w and two edges (u, v), (v, w) (whose images are dashed).
Bottom: fragment of the network matrix B corresponding to vertices u, v and edges (u, v), (v, w), illustrating how rows of B encode the degree requirements on vertex v.

– The rows for $(v^g, v^f) \in E_T$ describe the constraints as in (b), with $b^{\text{lower}}(r, v^f) = -\infty$ and $b^{\text{upper}}(r, v^f) = 0$ for those $v \in V$ with $f(\delta^+(v)) \geqslant f(\delta^-(v))$, and $b^{\text{upper}}(r, v^f) = +\infty$ for other $v \in V$.

– The rows for $(v^f, v^{f'}) \in E_T$ describe the constraints as in (c) with $b^{\text{lower}}(v^f, v^{f'}) = \lfloor f(\delta^-(v)) \rfloor$ and $b^{\text{upper}}(v^f, v^{f'}) = \lceil f(\delta^-(v)) \rceil$.

– The rows for $(v^g, v^{g'}) \in E_T$ describe the constraints as in (d) with $b^{\text{lower}}(v^g, v^{g'}) = \lfloor g(\delta^-(v)) \rfloor$ and $b^{\text{upper}}(v^f, v^{f'}) = \lceil g(\delta^-(v)) \rceil$ (recall that $g = z - f$).

– The capacity constraints $d^{\text{lower}} \leqslant (\bar{f}, \bar{g}) \leqslant d^{\text{upper}}$ are used to describe (e).

With $g = z - f$, the vector (f, g) is a feasible solution to the system. Consider the cost function $(w, w) \in \mathbb{R}^{2|E|}$. Using total unimodularity, there must be an integer solution $(\bar{f}, \bar{g})$ with $(w, w)^\top(\bar{f}, \bar{g}) \leqslant (w, w)^\top(f, g) = w^\top z$. Thus, $\bar{z} = \bar{f} + \bar{g}$ and $\bar{f}$ satisfy the statement of the lemma. $\square$

Equipped with the above lemmas, we are ready to prove Lemma 10.3.

Proof of Lemma 10.3. We are given a vertebrate pair $(\mathcal{I}, B)$ and disjoint non-empty vertex sets $U_1, \ldots, U_\ell \subseteq V \setminus V(B)$ such that the subgraphs $G[U_1], \ldots, G[U_\ell]$ are strongly connected and for every $S \in \mathcal{L}_{\geqslant 2}$ and $i = 1, \ldots, \ell$, we have either $U_i \cap S = \emptyset$ or $U_i \subseteq S$. For the Held-Karp solution x, let f be the witness flow guaranteed by Lemma 10.8.

We can assume that each edge e has either $f(e) = 0$ or $f(e) = x(e)$ without loss of generality, by breaking e up into two parallel copies and dividing its x and f values between them appropriately. We say that an edge e is *marked* if $f(e) = x(e)$ and *unmarked* otherwise.

Let us now introduce a convenient decomposition of x that we will obtain and utilize in our algorithm. By a *2-cycle* we mean a closed walk that visits every vertex at most twice and contains every edge at most once. A 2-cycle $C \subseteq E$ is *consistent* if, for any two consecutive edges $(u, v), (v, v') \in C$ with $v \notin V(B)$, if (u, v) is marked then (v, v') is also marked.

Claim 10.11. We can in polynomial time decompose x into consistent 2-cycles. That is, there is a polynomial-time algorithm that outputs consistent 2-cycles $C_1, C_2, \ldots, C_\ell$ and multipliers $\lambda_1, \lambda_2, \ldots, \lambda_\ell \geqslant 0$ such that $x = \sum_{i=1}^{\ell} \lambda_i \mathbb{1}_{C_i}$, where $\mathbb{1}_C \in \{0, 1\}^E$ denotes the indicator vector of a 2-cycle C.

Proof. The proof uses a variant of the standard cycle decomposition argument. We identify 2-cycles by constructing walks on edges in the support of x. The algorithm identifies and removes 2-cycles one by one. After the removal of every cycle, we maintain property *(b)* of witness flows: for every $v \in V \setminus V(B)$ the outgoing flow amount on marked edges is greater or equal to the incoming flow amount on marked edges.

We find walks using the following procedure. We start on an arbitrary (marked or unmarked) edge. If the walk uses a marked edge then let us select the next edge as a marked edge whenever possible, and after an unmarked edge let us continue on an unmarked edge if possible. We terminate according to the following rules:

1. If we visit a node $v \in V(B)$ the second time, then we terminate the current walk. We select C as the segment of the walk between the first and second visits to v. (If we started the walk from v, we also count this as a visit.)

2. When visiting a node $v \in V \setminus V(B)$ for the second time, we terminate if the outgoing edge of the first visit and the incoming edge of the second visit are of the same type (marked or unmarked). We also terminate if every outgoing edge of v is marked. We select C as the segment of the walk between the first and second visits to v.

3. If we visit a node in $v \in V \setminus V(B)$ the third time, then we always terminate. Let (v, u_1) and (v, u_2) be the edges where the walk left v for the first and second time, and let (z_1, v) and (z_2, v) be the edges where we arrived to v the second and third times. If (v, u_2) and (z_2, v) are both unmarked or both marked, then we let C be the segment of the walk between (v, u_2) and (z_2, v). Otherwise, we let C be the segment of the walk between (v, u_1) and (z_2, v) (visiting v twice). (These edges are of the same type, and so are (z_1, v) and (v, u_2).)

	graph	integral	obtained from the previous by
x	G	no	input to Lemma 10.3
x, f	G	no	finding a witness flow (Lemma 10.8)
x', f'	G'	no	introducing a_i, redirecting edges, subtracting x_i and f_i
$\bar{z}', \bar{f}'$	G'	yes	rounding (Lemma 10.9 applied to $z = 2x'$ and $2f'$)
$\bar{z}, \bar{f}$	G	yes	un-redirecting edges, mapping back to G
z^*, f^*	G	yes	adding walks P_i (to $\bar{f}$ only if $\bar{f}'(\delta^-(a_i)) = 1$)

Figure 10: This table summarizes the various circulations and flows that appear in our algorithm, in order.

Given C, we let λ denote the minimum flow value on any edge of C. We decrease the value of x on every edge of C by λ (deleting an edge if its x-value becomes zero), and also the value of f on every marked edge of C. We add C to the decomposition with coefficient λ and proceed to finding the next 2-cycle.

Assume property *(b)* of witnessed flows holds when we start the walk. Thus, whenever the walk enters a node $v \in V \setminus V(B)$ on a marked edge, it can continue on a marked edge (or terminate, if it is the second visit, and the first outgoing edge was marked). The above rules guarantee that C will be a consistent 2-cycle: if we close C in a node $v \in V \setminus V(B)$, then if the incoming edge is marked, the outgoing edge will also be marked. Moreover, if the walk is not terminated, then it can continue on a yet-unused edge.

It remains to show that property *(b)* is maintained after removing C. Assume we decrease the outgoing flow at a node $v \in V \setminus V(B)$ on a marked edge. First, notice that there can be at most one outgoing marked edge from v on C. If there was also an incoming marked edge on C, then the marked flow decreases by the same amount on these two edges. If all incoming edges were unmarked, then our rules implies that all outgoing edges at v are marked. In this case, *(b)* is trivially maintained (as x is, and remains, a circulation).

The proof can be immediately turned into a polynomial-time algorithm. Notice that the number of edges decreases by at least one at the removal of every cycle. □

We will construct an auxiliary graph $G' = (V', E')$ similarly as in the proof of Theorem 4.1. For convenience, Figure 10 gives an overview of the different steps, graphs and flows used by our algorithm. We select edge sets X_i^-, X_i^+ and flows x_i and f_i as follows:

- For every U_i we can select (possibly by subdividing edges as above) a subset of incoming edges $X_i^- \subseteq \delta^-(U_i)$ with $x(X_i^-) = 1/2$ such that either all edges $e \in X_i^-$ are marked or all are unmarked. This is possible since $x(\delta^-(U_i)) \geqslant 1$ (and all edges are either marked or unmarked).
- Take the decomposition of x into 2-cycles as guaranteed by Claim 10.11, and follow the incoming edges in X_i^- in the decomposition. We let X_i^+ be the set of edges on which these walks first leave U_i after entering on an edge in X_i^-. We let x_i denote the respective x-flow on the segments of these walks connecting the

heads of edges in X_i^- and the tails of edges in X_i^+. On the same edges, we define f_i by $f_i(e) = x_i(e)$ for every marked edge and $f_i(e) = 0$ for every unmarked edge.

Note that we have $0 \leqslant f - f_i \leqslant x - x_i$. We further claim that $(f - f_i)(\delta^+(v)) \geqslant (f - f_i)(\delta^-(v) \setminus X_i^-)$ for every $v \in V \setminus V(B)$. To see this, consider the consistent 2-cycles in the decomposition. In each of these 2-cycles an incoming marked edge must be followed by an outgoing marked edge when considering a vertex $v \in V \setminus V(B)$. This gives a (fractional) pairing of incoming and outgoing edges of v such that each incoming marked edge is paired with an outgoing marked edge. By our selection of X_i^- and x_i using the consistent 2-cycles, we have that the marked incoming edges that were not used by x_i or X_i^- are still paired with marked outgoing edges that were not used by x_i. We thus have $(f - f_i)(\delta^+(v)) \geqslant (f - f_i)(\delta^-(v) \setminus X_i^-)$ for every $v \in V \setminus V(B)$ as claimed.

We now transform G into a new graph G', x into a new circulation x', and f into a new f', as follows. For every $i = 1, \dots, \ell$, we introduce a new auxiliary vertex a_i and redirect all edges in X_i^- to point to a_i, and those in X_i^+ to point from a_i. We subtract the flow x_i from x and f_i from f inside U_i; hence the resulting vector x' will be a circulation in G', and $f' \leqslant x'$. We now have $f'(\delta^+(v)) \geqslant f'(\delta^-(v))$ for all $v \in V \setminus V(B)$ (using that $(f - f_i)(\delta^+(v)) \geqslant (f - f_i)(\delta^-(v) \setminus X_i^-)$ for every $v \in V \setminus V(B)$ as explained above). We also have $x'(\delta^-(a_i)) = x'(\delta^+(a_i)) = 1/2$, and either $f'(\delta^-(a_i)) = 0$ (in the case when $f(X_i^-) = 0$) or $f'(\delta^-(a_i)) = f'(\delta^+(a_i)) = 1/2$ (in the case when $f(X_i^-) = 1/2$: since in this case all edges in X_i^+ must also be marked due to the facts that the 2-cycles are consistent and $U_i \cap V(B) = \emptyset$). We define the weights $w'(e) = w(e)$ if e was not modified, and for every redirected edge e, we set $w'(e)$ as the weight of the edge it was redirected from. Thus, the total weight may only decrease: $\sum_{e \in E'} w'(e)x'(e) \leqslant \sum_{e \in E} w(e)x(e) = \mathrm{value}(\mathcal{I})$ (it decreases if the flows x_i are nonzero on edges with positive weight).

Let us now apply Lemma 10.9 in the graph G' with weights w' and $z = 2x'$ and $2f'$ in place of f. We thus obtain the integer vectors $\bar{z}'$ and $\bar{f}'$ with $w'^\top \bar{z}' \leqslant w'^\top z$. Note in particular that properties (c), (d) imply that $\bar{z}'(\delta^-(a_i)) = 1$ for every auxiliary vertex a_i; this is because $z(\delta^-(a_i)) = 1$ and $2f'(\delta^-(a_i)) \in \{0, 1\}$.

Now we map $\bar{z}'$ and $\bar{f}'$ back to the vectors $\bar{z}$ and $\bar{f}$ in the original graph G. Namely, if e is incident to an auxiliary vertex a_i, then we reverse the redirection of this edge, and move $\bar{z}$ and $\bar{f}$ to the original graph.

The vector $\bar{z}$ may not be a circulation. Specifically, since the in- and out-degree of a_i were exactly 1 in $\bar{z}'$, in each component U_i there is a pair of vertices u_i, v_i which are the head and tail, respectively, of the mapped-back edges adjacent to a_i. These are the only vertices whose in-degree may differ from their out-degree. (They differ unless $u_i = v_i$.) To repair this, for each $i = 1, \dots, \ell$ with $u_i \neq v_i$, we route a path P_i from u_i to v_i in U_i; this is always possible as we assumed that U_i is strongly connected.

We obtain a circulation z^* from $\bar{z}$ by increasing the value by 1 on every edge of every such path P_i. Further, we obtain f^* from $\bar{f}$ as follows. If $\bar{f}'(\delta^-(a_i)) = 0$, then we let f^* be identical to $\bar{f}$ inside U_i. If $\bar{f}'(\delta^-(a_i)) = 1$, then we increase the value of $\bar{f}$ by 1 on every edge of P_i.

Claim 10.12. We have $w^\top z^* \leqslant 2\,\mathrm{value}(\mathcal{I}) + \mathrm{lb}_{\mathcal{I}}(\bar{B})$, and f^* is a witness flow for z^*.

Proof. By the construction, $w^\top \bar{z} = w'^\top \bar{z}' \leqslant w'^\top z = 2w'^\top x' \leqslant 2w^\top x = 2\,\mathrm{value}(\mathcal{I})$. We obtained z^* from $\bar{z}$ by increasing it on a set of disjoint paths P_i in $V \setminus V(B)$, and these paths do not cross any set in $\mathcal{L}_{\geqslant 2}$. Thus, their total cost is bounded by $\mathrm{lb}_{\mathcal{I}}(\bar{B}) = 2\sum_{v\in V\setminus V(B)} y_v$.

We next verify that f^* is a witness flow for z^*. Property *(a)*, i.e., that $f^* \leqslant z^*$ is clear from the construction. We proceed to verify properties *(c)* and *(d)*, which state that $f^*(e) = 0$ for each backward edge e and $f^*(e) = z^*(e)$ for each forward edge, respectively. This holds for the initial witness flow f. We are now going to use that all edges in $G[U_i]$ are neutral since for every $S \in \mathcal{L}_{\geqslant 2}$ either $U_i \cap S = \emptyset$ or $U_i \subseteq S$. It follows that f' and f only differ in neutral edges and hence f' also satisfies properties *(c)* and *(d)*. By Lemma 10.9, we thus have that $\bar{f}'$ also satisfies these properties. Finally, f^* is obtained from $\bar{f}'$ by reversing the redirection of the edges incident to the auxiliary vertices and potentially increasing the flow value on neutral edges. As these operations maintain the properties, we have that f^* satisfies properties *(c)* and *(d)*.

Finally, for *(b)*, recall that we had $f'(\delta^+(v)) \geqslant f'(\delta^-(v))$ for every $v \in V \setminus V(B)$, and hence the same holds for $\bar{f}'$ by Lemma 10.9. For $\bar{f}$, this condition can be violated only at some nodes u_i for some values of $i = 1, 2, \ldots, \ell$ for which $\bar{f}'(\delta^-(a_i)) = \bar{f}'(\delta^+(a_i)) = 1$ and $u_i \neq v_i$. We obtain f^* from $\bar{f}$ by increasing the flow value on the path P_i from u_i to v_i for such i; this increases $f^*(\delta^+(u_i))$ so that $f^*(\delta^+(u_i)) \geqslant f^*(\delta^-(u_i))$, while not violating $f^*(\delta^+(v_i)) \geqslant f^*(\delta^-(v_i))$ since $\bar{f}'(\delta^-(a_i)) = \bar{f}'(\delta^+(a_i)) = 1$ implies $\bar{f}(\delta^+(v_i)) \geqslant \bar{f}(\delta^-(v_i))+1$. □

Let F be the Eulerian edge multiset obtained by taking z^*_e copies of edge e. The proof concludes by showing that F satisfies all requirements of Lemma 10.3. The cost bound *(a)* follows by Claim 10.12 since $w(F) = w^\top z^*$. The connectivity requirement *(b)*, namely that $|\delta^-_F(U_i)| \geqslant 1$ for every $i = 1, \ldots, \ell$, is immediate by the construction.

Let us now show *(c)*, that is, $|\delta^-_F(v)| = z^*(\delta^-(v)) \leqslant 4$ for every $v \in V$ such that $x(\delta^-(v)) = 1$. Note that we have $x'(\delta^-(v)) \leqslant 1$. Denote $g' = x' - f'$ and $\bar{g}' = \bar{x}' - \bar{f}'$. By properties (c), (d) of Lemma 10.9 we have

$$\bar{z}'(\delta^-(v)) = \bar{f}'(\delta^-(v)) + \bar{g}'(\delta^-(v)) \leqslant \lceil 2f'(\delta^-(v))\rceil + \lceil 2g'(\delta^-(v))\rceil \leqslant 3\,,$$

as the maximum value of the function $\lceil 2p\rceil + \lceil 2q\rceil$ subject to $p + q \leqslant 1$ is 3. Adding the paths P_i and reversing the redirection of edges incident to auxiliary vertices a_i may further increase $z^*(\delta^-(v))$ over $\bar{z}'(\delta^-(v))$ by 1.

Property *(d)* requires that every subtour in F that crosses a tight set in $\mathcal{L}_{\geqslant 2}$ visit a vertex of the backbone. This follows from Lemma 10.7 since z^* is a witnessed circulation. □

11 Completing the Puzzle: Proof of Theorem 1.1

We now combine the techniques and algorithms of the previous sections to obtain a constant-factor approximation algorithm for ATSP. In multiple steps, we have reduced ATSP to finding tours for vertebrate pairs. Every reduction step was polynomial-time and increased the approximation ratio by a constant factor. Hence, together they give a constant-factor approximation algorithm for ATSP.

We now give an overview of these reductions and set the parameters. Throughout, $\varepsilon > 0$ will be a fixed small value. We set $\delta = 0.78$. All approximation guarantees are with respect to the optimum value of the Held-Karp relaxation $\mathrm{LP}(G, w)$. The reduction proceeds using the following algorithmic subroutines.

– Corollary 5.2 provides a polynomial-time α_S-approximation $\mathcal{A}_S$ for singleton instances, with $\alpha_S = 18 + \varepsilon$. This will be used to find a (quasi-)backbone for irreducible instances (Lemma 9.3).

– Algorithm $\mathcal{A}_{\text{ver}}$, which, for a vertebrate pair $(\mathcal{I}, B)$, finds a tour of cost $\kappa\,\text{value}(\mathcal{I}) + \eta\,\text{lb}_B(V) + w(B)$, where $\kappa = 2$ and $\eta = 37 + 36\varepsilon$ (Corollary 10.2). Algorithm $\mathcal{A}_{\text{ver}}$ uses the reduction of Theorem 5.1 from Subtour Partition Cover to ATSP.

– Algorithm $\mathcal{A}_{\text{irr}}$ which, provided $\mathcal{A}_{\text{ver}}$ as above, obtains a polynomial-time ρ-approximation algorithm for irreducible instances, where $\rho = (\kappa + \eta(1-\delta) + \alpha_S + 3)/(2\delta - 1) < 55.61$ for sufficiently small $\varepsilon > 0$ (Theorem 9.4).

– Algorithm $\mathcal{A}_{\text{lam}}$, which converts the ρ-approximation algorithm $\mathcal{A}_{\text{irr}}$ to a $2\rho/(1-\delta)$-approximation algorithm for an arbitrary laminarly-weighted instance $\mathcal{I}$. Here, $2\rho/(1-\delta) < 506$ (Theorem 8.3).

– Our final Algorithm $\mathcal{A}_{\text{ATSP}}$, which reduces an arbitrary input weighted digraph (G, w) to a laminarly-weighted instance, keeping the same approximation ratio (Theorem 2.4).

All in all we have thus obtained a polynomial-time algorithm for ATSP that returns a tour of value at most 506 times the Held-Karp lower bound.

Integrality gap. Theorem 1.1 of course implies an upper bound of 506 on the integrality gap of the Held-Karp relaxation. However, if we do not require a polynomial-time algorithm, then the loss factor of $9(1+\varepsilon)$ in the reduction of Theorem 5.1 can be decreased to 5. Therefore non-constructively we can have $\alpha'_S = 10$ and $\eta' = 1 + 5 \cdot 4$ (instead of $\eta = 1 + 9 \cdot (1+\varepsilon) \cdot 4$), which yields $\rho' < 35.04$ and a final integrality gap of at most 319.

Theorem 11.1. *The integrality gap of the asymmetric Held-Karp relaxation is at most* 319.

Asymmetric Traveling Salesman *Path* Problem. In the Asymmetric Traveling Salesman Path Problem (ATSPP), in addition to the usual ATSP input, we are also given two special vertices $s, t \in V$ and wish to find a walk that visits all vertices, starts from s, and ends at t. Feige and Singh [FS07] proved that if there is a β-approximation algorithm for ATSP, then there is a $((2+\varepsilon)\beta)$-approximation algorithm for ATSPP for any $\varepsilon > 0$. Together with Theorem 1.1, this implies:

Corollary 11.2. *There is a polynomial-time algorithm for ATSPP that returns a tour of value at most* 1012 *times the integral optimum.*

Note that the approximation ratio here is not bounded in terms of the Held-Karp lower bound[12]. Köhne, Traub and Vygen [KTV19] give a similar reduction as Feige and Singh, but for integrality gaps, with a loss of $\beta \mapsto 4\beta - 3$. Together with Theorem 11.1, this implies:

[12]For ATSPP, this is defined as the optimal value of a relaxation that is similar to LP(G, w), with the differences that $x(\delta^+(s)) - x(\delta^-(s)) = 1$, $x(\delta^-(t)) - x(\delta^+(t)) = 1$, and the cuts S with $s \in S$ are only required to have at least one outgoing edge (but possibly no incoming edge).

Corollary 11.3. *The integrality gap of the asymmetric Held-Karp relaxation for ATSPP is at most* 1273*.*

Finally, since their reduction is constructive, together with Theorem 1.1 we get:

Corollary 11.4. *There is a polynomial-time algorithm for ATSPP that returns a tour of value at most* 2021 *times the Held-Karp lower bound.*

12 Conclusion

In this paper we gave the first constant-factor approximation algorithm for ATSP. The result was obtained in three steps. First, we gave a generic reduction from ATSP to Subtour Partition Cover. Then, we showed how to simplify general ATSP instances to very structured instances (vertebrate pairs) by only incurring a constant-factor loss in the approximation guarantee. Finally, these instances were solved using the connection to Subtour Partition Cover.

Subsequent to our work, Traub and Vygen [TV20] have attained a substantially better approximation guarantee $22 + \varepsilon$, with a matching integrality gap 22. The main improvement in the approximation factor is due to eliminating the reduction to irreducible instances, and using a variant of Subtour Partition Cover for arbitrary instances in a careful recursive framework. It remains open whether one can get an approximation algorithm (nearly) matching the integrality gap of the Held–Karp relaxation, and whether the current lower bound on the gap is tight.

Open Question 12.1. Is the integrality gap of the standard LP relaxation upper-bounded by 2?

We remark that the above question is also open for the simpler case of unweighted instances, where the best known upper bound is 13 [Sve15].

As mentioned in the introduction, Asadpour et al. [AGM$^+$17] introduced a different approach for ATSP based on so-called thin spanning trees. Our algorithm does not imply a better construction of such trees and the $O(\text{poly}\log\log n)$-thin trees of [AG15] remain the best such (non-constructive) result. Whether trees of better thinness exist is an interesting question. Also, as shown in [AKS10], the construction of $O(1)$-thin trees would lead to a constant-factor approximation algorithm for the bottleneck ATSP problem. There, we are given a complete digraph with edge weights satisfying the triangle inequality, and we wish to find a Hamiltonian cycle that minimizes the maximum edge weight. A tight 2-approximation algorithm for bottleneck *symmetric* TSP was given already in [Fle74, Lau81, PR84], but no constant-factor approximation is known for bottleneck ATSP.

Open Question 12.2. Is there a $O(1)$-approximation algorithm for bottleneck ATSP?

We believe that this is an interesting open question in itself, and progress on it may shed light on the existence of $O(1)$-thin trees.

Acknowledgments

The authors are grateful to Johan Håstad and Hyung-Chan An for inspiring discussions and valuable comments that influenced this work, and to Jens Vygen for useful feedback on the manuscript. We are grateful to András Frank for pointing us to the reference [Kar96]. We are also grateful to the Simons Foundation which has sponsored several workshops and programs where numerous inspiring discussions have taken place.

References

[AG15] Nima Anari and Shayan Oveis Gharan. Effective-resistance-reducing flows, spectrally thin trees, and asymmetric TSP. In *Proceedings of the 56th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 20–39, 2015.

[AGM$^+$17] Arash Asadpour, Michel X. Goemans, Aleksander Madry, Shayan Oveis Gharan, and Amin Saberi. An $O(\log n/\log\log n)$-approximation algorithm for the asymmetric traveling salesman problem. *Operations Research*, 65(4):1043–1061, 2017.

[AKS10] Hyung-Chan An, Robert D. Kleinberg, and David B. Shmoys. Approximation algorithms for the bottleneck asymmetric traveling salesman problem. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX)*, pages 1–11, 2010.

[AKS15] Hyung-Chan An, Robert D. Kleinberg, and David B. Shmoys. Improving Christofides' algorithm for the s-t path TSP. *J. ACM*, 62(5):34:1–34:28, 2015.

[Art82] TS Arthanari. On the traveling salesman problem. In *XI Symposium on Mathematical Programming held at Bonn, West Germany*, 1982.

[Blä08] Markus Bläser. A new approximation algorithm for the asymmetric TSP with triangle inequality. *ACM Transactions on Algorithms*, 4(4), 2008.

[Car96] Robert Carr. Separating over classes of tsp inequalities defined by 0 node-lifting in polynomial time. In William H. Cunningham, S. Thomas McCormick, and Maurice Queyranne, editors, *Integer Programming and Combinatorial Optimization*, pages 460–474, Berlin, Heidelberg, 1996. Springer Berlin Heidelberg.

[CFN85] Gérard Cornuéjols, Jean Fonlupt, and Denis Naddef. The traveling salesman problem on a graph and some related integer polyhedra. *Mathematical Programming*, 33(1):1–27, 1985.

[CGK06] Moses Charikar, Michel X. Goemans, and Howard J. Karloff. On the integrality ratio for the asymmetric traveling salesman problem. *Math. Oper. Res.*, 31(2):245–252, 2006.

[Chr76] Nicos Christofides. Worst-case analysis of a new heuristic for the travelling salesman problem. Technical report, Graduate School of Industrial Administration, CMU, 1976.

[FGM82] Alan M. Frieze, Giulia Galbiati, and Francesco Maffioli. On the worst-case performance of some algorithms for the asymmetric traveling salesman problem. *Networks*, 12(1):23–39, 1982.

[Fle74] Herbert Fleischner. The square of every two-connected graph is Hamiltonian. *Journal of Combinatorial Theory, Series B*, 16(1):29 – 34, 1974.

[FS07] Uriel Feige and Mohit Singh. Improved approximation ratios for traveling salesperson tours and paths in directed graphs. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Technique (APPROX)*, pages 104–118, 2007.

[GLS12] Martin Grötschel, László Lovász, and Alexander Schrijver. *Geometric algorithms and combinatorial optimization*, volume 2. Springer Science & Business Media, 2012.

[GS11] Shayan Oveis Gharan and Amin Saberi. The asymmetric traveling salesman problem on graphs with bounded genus. In *Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 967–975, 2011.

[GSS11] Shayan Oveis Gharan, Amin Saberi, and Mohit Singh. A randomized rounding approach to the traveling salesman problem. In *Proceedings of the 52nd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 550–559, 2011.

[Kar96] Alexander Karzanov. How to tidy up a symmmetric set-system by use of uncrossing operations. *Theoretical Computer Science*, 157:215–225, 1996.

[KLS13] Marek Karpinski, Michael Lampis, and Richard Schmied. New inapproximability bounds for TSP. In *Proceedings of the 24th International Symposium on Algorithms and Computation (ISAAC)*, pages 568–578, 2013.

[KLSS05] Haim Kaplan, Moshe Lewenstein, Nira Shafrir, and Maxim Sviridenko. Approximation algorithms for asymmetric TSP by decomposing directed regular multigraphs. *J. ACM*, 52(4):602–626, 2005.

[KTV19] Anna Köhne, Vera Traub, and Jens Vygen. The asymmetric traveling salesman path LP has constant integrality ratio. *Mathematical Programming*, 2019.

[Lau81] H. T. Lau. Finding EPS-graphs. *Monatshefte für Mathematik*, 92(1):37–40, Mar 1981.

[MS16] Tobias Mömke and Ola Svensson. Removing and adding edges for the traveling salesman problem. *J. ACM*, 63(1):2:1–2:28, 2016.

[Muc12] Marcin Mucha. 13/9-approximation for graphic TSP. In *29th International Symposium on Theoretical Aspects of Computer Science, STACS 2012*, pages 30–41, 2012.

[PR84] R.Gary Parker and Ronald L Rardin. Guaranteed performance heuristics for the bottleneck travelling salesman problem. *Operations Research Letters*, 2(6):269 – 272, 1984.

[Sch98] Alexander Schrijver. *Theory of linear and integer programming*. John Wiley & Sons, New York, 1998.

[Sch03] Alexander Schrijver. *Combinatorial Optimization - Polyhedra and Efficiency*. Springer-Verlag, Berlin, 2003.

[Ser78] Anatoliy I. Serdyukov. On some extremal tours in graphs (in russian). *Upravlyaemye systemy*, 17:76–79, 1978.

[STV18a] Ola Svensson, Jakub Tarnawski, and László A. Végh. A constant-factor approximation algorithm for the asymmetric traveling salesman problem. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 204–213, 2018.

[STV18b] Ola Svensson, Jakub Tarnawski, and László A. Végh. Constant factor approximation for ATSP with two edge weights. *Mathematical Programming*, 172(1-2):371–397, 2018.

[SV14] András Sebő and Jens Vygen. Shorter tours by nicer ears: 7/5-approximation for the graph-TSP, 3/2 for the path version, and 4/3 for two-edge-connected subgraphs. *Combinatorica*, 34(5):597–629, 2014.

[Sve15] Ola Svensson. Approximating ATSP by relaxing connectivity. In *Proceedings of the 56th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 1–19, 2015.

[Tut65] William Thomas Tutte. Lectures on matroids. *J. Research of the National Bureau of Standards (B)*, 69:1–47, 1965.

[TV20] Vera Traub and Jens Vygen. An improved approximation algorithm for atsp. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 1–13, 2020.

[vBS20] René van Bevern and Viktoriia A Slugina. A historical note on the 3/2-approximation algorithm for the metric traveling salesman problem. *Historia Mathematica*, 2020.

[VY99] Santosh Vempala and Mihalis Yannakakis. A convex relaxation for the asymmetric TSP. In *Proceedings of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 975–976, 1999.